한국정보보호학회
Korea Institute of Information Security & Cryptology

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

일자 2018년 12월 8일(토) **장소** 세종대학교(군자관, 충무관)

주최 한국정보보호학회 Korea Institute of Information Security & Cryptology 주관 세종대학교 SEJONG UNIVERSITY

후원 국가정보원 National Intelligence Service Korea, 행정안전부, KISA 한국인터넷진흥원, ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, NSR 국가보안기술연구소, LIG 넥스원, 나눔과 협업 정보과 공익 보안프로젝트 WWW.BOANPROJECT.COM, kakao

Proceedings

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

목 차

• 구두발표 트랙 A 1

• 구두발표 트랙 B 82

• 구두발표 트랙 C 164

• 구두발표 트랙 D 248

• 구두발표 트랙 E 334

• 포스터발표 트랙 E 380

• 포스터발표 트랙 F 513

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

구두발표 트랙 A

CUDA를 활용한 차분 확률의 통계적 분석*

조은지*, 김성겸*, 박연지*, 이용성*, 홍득조**, 성재철***, 홍석희*
*고려대학교 정보보호학과, **전북대학교 IT정보공학과, ***서울시립대 수학과

The Statistical Analysis of the Differential Probability using CUDA.

Eunji Jo*, Seong-Gyeom Kim*, YeonJi Park*, Yongseong Lee*,
Deukjo Hong**, Jaechul Sung***, Seokhie Hong*
*Graduate School of Information Security, Korea University.
**Department of Information Technology, Chonbuk National University.
***Department of Mathematics, University of Seoul.

요 약

본 논문은 병렬 프로그래밍 CUDA를 활용하여 경량 블록암호 PRESENT와 GIFT를 대상으로 Markov Cipher 가정하의 차분 경로의 기대 확률과 통계적으로 추정한 실제 차분 확률을 비교하여 다름을 보인다. 또한 키에 따라 달라지는 실제 차분 경로를 만족하는 평문쌍의 분포를 히스토그램을 이용하여 분석한다. 이 분포를 통해 Markov Cipher 가정하에 구한 몇몇 차분 경로의 기대 확률이 실제 차분 확률과 다를 수 있음을 실험적으로 보인다. 이를 통해 같은 확률의 차분 특성이지만 더 적은 데이터를 이용하여 공격할 수 있는 차분 특성이 있음을 알 수 있다. 마지막으로 키에 따라서 달라지는 차분 경로의 분포가 정규분포가 아닌 부분적으로 정규분포를 따르는 경우가 있음을 제시한다. 이런 경우 특정키를 이용하면 차분 공격에 내성을 가질 수도 있음을 보였다.

I. 서론

병렬 프로그래밍이란 사용 가능한 프로세서 자원들로 동시에 연산을 수행하여 연산 시간을 줄이는 것이다. 특히 그래픽 처리 장치(GPU)를 사용하는 병렬 프로그래밍 GPGPU(General-Purpose computing on Graphics Processing Units) 기술을 이용한 연구들이 활발히 진행되고 있다. GPGPU 기술을 활용한 예로는 전자 화폐 채굴과 2017년 SHA-1에 대한 충돌쌍 공격(collision attack)을 현실적으로 수행한 것이 있다[7]. 이처럼 GPGPU 기술을 암호 분야에 접목하여 블록암호의 차분 특성의 기대 확률을 추정하는 데 이용될 수 있다.

차분 분석(Differential Cryptanalysis)은 블록암호의 안전성 분석에 사용되는 대표적인 분석 방법이다[1]. 차분 분석에 사용되는 차분 특성(Differential Characteristic)의 기대 확률은 흔히 중간 차분(Intermediate Differential)을 고려한 차분 경로(Differential Trail)의 확률과 고려하지 않는 디퍼런셜(Differential)의 확률로 나뉜다.

각 라운드의 출력 차분 결과값이 Markov Chain을 형성하여 라운드 독립성을 만족하는 암호는 Markov Cipher라고 한다[2]. 보통 디퍼런셜의 확률 계산이 어려우므로, 블록암호를 Markov Cipher라고 가정하여 중간 차분을 고려한 확률을 구한다. 대부분 암호는 라운드 독립성을 만족하기 어려운 키스케줄을 사용하는 데도 Markov Cipher를 만족한다는 가정하에 안전성을 분석하지만, 실제 Markov Cipher를 만족하는지는 입증할 수 없다. 즉 Markov Cipher라고 가정하여 구한 차분 경로의 기대 확률과 실제 차분 경로의 확률은 다를 수 있다.

본 논문에서는 Markov Cipher 가정하의 차분 경로의 기대 확률과 실제 확률을 비교하여 Markov Cipher 가정의 타당성을 검증한다. 실험을 통해 Markov Cipher 가정하에 동일 확률을 가지는 차분 경로 간의 실제 확률이 다를 수 있음을 보인다. 또한, 싱글키에서의 차분 경로의 확률이 정규분포에 근사한다고 알려졌지만[8], 경량 블록암호에서도 동일하게 적용되는지를 확인한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 차분 분석 등 배경 지식을 소개하고, Ⅲ장에서는 검정을 위한 데이터 수집 방법을 소개한다. Ⅳ장에서는 제시한 방법으로 PRESENT[3]와 GIFT[4]를 대상으로 한 실험 결과를 제시하며, 마지막 Ⅴ장에서 결론을 맺는다.

II. 배경 지식

2.1 차분 분석

차분 분석은 1990년 Eli Biham과 Adi Shamir에 의해

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

제안되었으며, 평문쌍의 입력 차분이 비선형성으로 인해 $\frac{1}{2}$보다 더 높은 확률로 특정 출력 차분을 발생시키는 디퍼런셜을 확률을 통해 분석한다. 하지만 디퍼런셜의 경우에 가능한 모든 경로를 찾을 수 없으므로 대부분 모든 라운드가 독립이라고 가정하여 가장 높은 확률을 갖는 차분 경로로 구성되는 차분 특성을 이용하여 분석한다.

2.2 차분 경로의 기대 확률의 계산

i라운드의 입출력 차분 D_i를 만족하는 차분 경로 $D_0, D_1, \ldots, D_r$가 주어졌을 때, 주어진 블록 암호가 Markov Cipher를 만족하면 r라운드 차분 경로의 기대 확률 E_{Markov}를 다음과 같이 구할 수 있다[1].

$$E_{Markov} = \prod_{i=1}^{r} pr(D_{i-1} \rightarrow D_i)$$

고정된 비밀키 k에 대한 차분 경로의 확률 $P_{real}(k)$는 다음과 같이 나타낼 수 있다. $S_{real}(N_p, k)$는 가능한 모든 평문쌍의 개수 N_p개 중 해당 차분 경로를 만족하는 평문쌍이다.

$$P_{real}(k) = \frac{S_{real}(N_p, k)}{N_p}$$

가능한 모든 키의 개수 N_k개의 비밀키를 갖는 실제 블록암호에 대한 차분 경로의 기대 확률 E_{real}은 다음과 같이 구할 수 있다.

$$E_{real} = \frac{\sum_{i=1}^{N_k} P_{real}(k_i)}{N_k}$$

블록 크기는 64비트, 키 크기가 128비트인 블록암호에서는 전수조사의 데이터 복잡도가 약 2^{192}이므로 $P_{real}(k)$의 기댓값 E_{real}은 구할 수 없다. 그러나 E_{real}에 대한 통계적 검정은 30개 이상의 비밀키와 평문쌍을 표본 추출하여 수행할 수 있다.

평문쌍 n_p개를 표본 추출하여 $P_{real}(k)$를 $\overline{P}_{real}(k)$로 점추정하고 다음과 같이 고정된 비밀키 k에 대한 차분 경로의 확률 $P_{real}(k)$를 추정할 수 있다.

$$P_{real}(k) \approx \overline{P}_{real}(k) = \frac{S_{real}(n_p, k)}{n_p}$$

n_k개의 비밀키로 구한 점추정값 $\overline{P}_{real}(k)$를 표본으로 사용하여 E_{real}의 값에 대한 검정을 수행할 수 있다.

2.3 통계적 가설 검정

통계적 가설 검정(Statistical Hypothesis Testing)이란 표본으로부터 얻은 정보를 이용하여 모수에 대한 주장의 옳고 그름을 판정하는 과정이다[5]. 모수에 대한 주장을 가설이라 하고, 가설에는 귀무가설과 대립가설이 있다. 귀무가설 H_0은 검정 대상이 되는 가설로, 유의수준에 따라 기각하는 것이 목표이다. 대립가설 H_1은 귀무가설에 상반되는 가설로 증거에 의하여 입증하고자 하는 가설이다.

본 논문에서는 Markov Cipher 가정을 통해 구한 차분 경로의 기대 확률 E_{Markov}가 실제 확률 E_{real}과 같은지 확인하기 위해 다음의 통계적 가설을 설정하였다.

$$H_0 : E_{real} = E_{Markov}$$
$$H_1 : E_{real} \neq E_{Markov}$$

또한 E_{Markov}의 값이 같은 서로 다른 차분 경로의 실제 확률 $E1_{real}$, $E2_{real}$이 서로 같은지 확인하기 위해 다음의 통계적 가설을 설정하였다.

$$H_0 : E1_{real} = E2_{real}$$
$$H_1 : E1_{real} \neq E2_{real}$$

2.4 CUDA

CUDA(Compute Unified Device Architecture)는 NDIVIA에서 그래픽 카드의 병렬 프로그램 개발 환경으로 GPGPU 기술의 활용을 극대화하기 위해 개발하였다[6]. GPGPU 기술은 CPU와 달리 간단한 문제를 반복해서 여러 번 수행하는 SPMD(Single Program, Multiple Data) 문제를 해결하는 데 적합한 기술이다.

III. 검정을 위한 데이터 수집 방법

3.1 데이터 수집 방법

주어진 차분 특성을 만족하는 평문쌍의 개수 $S(n_p, k_i)$를 수집하는 알고리즘은 다음과 같다.

Algorithm 1. 검정을 위한 표본 데이터 수집 방법

Input: n_p, n_k, $(D_0, D_1, \ldots, D_r)$
Output: $S(n_p, k_i)$ (단, $1 \le i \le n_k$)
Set counter $S(n_p, k_i)$ and initialize as 0
1. for i from 1 to n_k do
   1.1 임의의 k_i 선택 → n_k개 표본 추출
   1.2 키스케줄 연산으로 라운드키 생성
2. for i from 1 to n_p do
   임의의 평문쌍 p_{i1}, p_{i2} 생성 → n_p개 표본 추출
3. for i from 1 to n_k do
   for j from 1 to n_p do
   Encryption(p_{j1}, k_i), (p_{j2}, k_i)
   if Enc$(p_{j1}) \oplus$ Enc$(p_{j2}) = D_j$
   $S(n_p, k_i)$++ → 경로를 만족하는 평문쌍
4. return $S(n_p, k_i)$ (단, $1 \le i \le n_k$)

Algorithm 1은 n_k, n_p개를 랜덤으로 표본 추출하여 모든 비밀키와 평문쌍에 대해 차분 경로를 만족하는 평문쌍의 개수 $S(n_p, k_i)$를 계산한다.

3.2 실험 환경 설정

확률이 최소 2^{-20}인 차분 경로를 이용하므로 적어도 2^{20}개 이상의 평문쌍이 필요하고, 실험 결과를 최소 3시간 내에 얻기 위하여 실험에서 사용된 평문쌍과 키의 개수는 2^{24}개와 2^{11}개를 기준으로 진행하였다. 차분 경로의 확률은 CUDA를 활용하여 계산하였으며, GIFT와 PRESENT 각각 4라운드의 가장 높은 확률 $2^{-11.415}$, 2^{-12}의 차분 경로 7개에 대하여 실험을 진행하였다.

이 실험에서 사용된 프로세서는 CPU는 i7-8700K를 사용하였고, GPU는 GTX-1080 Ti를 사용하였다.

IV. 실험 결과

4.1 GPU와 CPU의 성능 비교

본 절에서는 GIFT와 PRESENT에 대상으로 Algorithm 1을 CPU와 GPU에 적용하여 효율성을 비교한다.

성능 비교 실험은 확률이 각각 2^{-18}, 2^{-20}인 입출력이 같은 4라운드 차분 경로(PRESENT-T_0와 GIFT-T_1)를 대상으로 진행했고, 결과는 [표 1, 2]와 같다. GPU를 이용했을 때, CPU보다 평균적으로 약 18배 빠른 것을 확인할 수 있다. 즉 GPU를 이용하면 더 효율적으로 분석 데이터를 현실적인 시간 내에 얻을 수 있다.

[표 1] PRESENT의 CPU와 GPU를 이용한 속도 비교표

대상	평문쌍 개수	키 개수	소요 시간	대상	평문쌍 개수	키 개수	소요 시간
CPU	2^{24}	2^{11}	142분	CPU	2^{25}	2^{12}	526분
GPU			7분	GPU			31분
CPU	2^{25}	2^{11}	253분				
GPU			15분				

[표 2] GIFT의 CPU와 GPU를 이용한 속도 비교표

대상	평문쌍 개수	키 개수	소요 시간	대상	평문쌍 개수	키 개수	소요 시간
CPU	2^{24}	2^{11}	138분	CPU	2^{25}	2^{12}	552분
GPU			7분	GPU			30분
CPU	2^{25}	2^{11}	277분	CPU	2^{27}	2^{12}	2227분
GPU			15분	GPU			127분

4.2 차분 경로의 기댓값 검정

본 절에서는 GIFT와 PRESENT를 대상으로 3장의 방법으로 수행된 실험 결과와 Markov Cipher 가정하에 얻은 기대 확률을 비교한다. 이를 위해 확률값에 사용한 평문쌍을 곱하여 통계적 가설 검정을 수행한다.

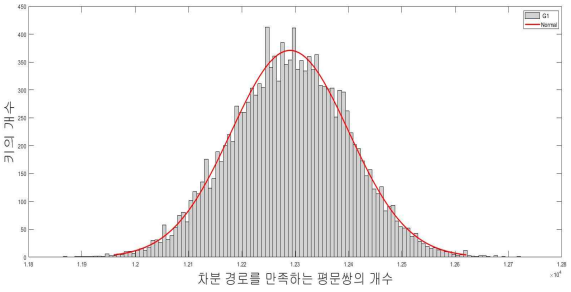
첫 번째는 GIFT에 대한 검정이다. 실험은 평문쌍 2^{25}개, 키 14336(약 2^{13})개, 이용한 GIFT의 차분 경로 7개는 $G_1,\ldots,G_7$로 표현한다. $\overline{P}_{real}(k)$는 E_{real}의 점추정값을 나타내고, $\overline{P}_{real}(k)\times n_p$의 평균값은 $\overline{E}_{real}$로 표현한다. [표 3]은 GIFT의 실험 결과이고, E_{markov}는 Markov Cipher라 가정하여 구한 차분 경로의 기대 확률을 나타낸다. G_i의 분포는 $B(2^{25},2^{-11.415})$을 따르고, 표본크기가 2^{25}개로 충분히 크므로, G_i의 분포는 $N(12288.3,0.92^2)$을 따른다고 할 수 있다. 확률변수 X가 정규분포 $N(m,\sigma^2)$을 따를 때, 표준정규확률변수 $Z=\dfrac{X-m}{\sigma}$으로 정규화하면 $Z\sim N(0,1)$을 따른다[5].

정리한 내용을 바탕으로 귀무가설과 대립가설을 $H_0:\overline{E}_{real}=12288.3, H_1:\overline{E}_{real}\neq 12288.3$이라 하고, 유의수준 $\alpha=0.01$에서 검정한다. 각 경로에 따른 Z를 구한 후, 표준정규분포의 관측값 $Z_{0.005}=2.58$을 기준으로 $|Z|\le Z_{0.005}$이면 기각할 수 없고, $|Z|>Z_{0.005}$이면 기각한다.

G_i를 만족하는 평문쌍의 개수에 대해 검정하면, $|Z|\le Z_{0.005}$인 G_1, G_2, G_4는 귀무가설 $H_0:\overline{E}_{real}=12288.3$을 유의수준 $\alpha=0.01$에서 기각할 수 없다. 그러나 $|Z|>Z_{0.005}$인 G_3, G_5, G_6, G_7은 유의수준 $\alpha=0.01$에서 기각한다. 즉 Markov Cipher라고 가정하여 구한 GIFT의 4라운드 차분 경로의 기대 확률과 실험을 통한 차분 경로의 확률이 모두 다 일치한다고 할 수 없다. [그림 1]은 G_1의 분포를 나타낸 그래프이다.

[표 3] GIFT의 실험 결과

경로	$E_{markov}\times n_p$	$\overline{P}_{real}(k)\times n_p$의 평균($\overline{E}_{real}$)	표준편차	Z
G_1	12288.3	12290.3	108.9	2.1
G_2		12287.9	111.8	−0.4
G_3		12268.3	111.0	−21.6
G_4		12288.6	108.6	0.3
G_5		12297.3	111.9	9.7
G_6		12278.6	111.4	−10.4
G_7		12297.7	110.8	10.1



[그림 1] GIFT의 G_1의 분포

두 번째 PRESENT도 GIFT와 동일한 조건하에 실험하였다. PRESENT의 차분 경로 7개는 $P_1,\ldots,P_7$로 표현하고, [표 4]는 PRESENT의 실험 결과이다. P_i의 분포는 $B(2^{25},2^{-12})$을 따르고, 표본크기가 2^{25}개로 충분히 크므로, P_i의 분포는 $N(8192,0.75^2)$를 따른다고 할 수 있다. 위 내용을 바탕으로 귀무가설과 대립가설을 $H_0:m=8192$, $H_1:m\neq 8192$라 하고 유의수준 $\alpha=0.01$에서 검정한다.

P_i를 만족하는 평문쌍의 개수에 대해 검정하면, 모든 P_i에 대해 귀무가설 $H_0:m=8192$를 유의수준 $\alpha=0.01$에서 기각할 수 없다. 즉 Markov Cipher라고 가정하여 구한 PRESENT의 4라운드 차분 경로의 기대 확률과 실험을 통한 차분 경로의 확률이 일치한다고 할 수 있다.

[표 4] PRESENT의 실험 결과

경로	$E_{markov}\times n_p$	$\overline{P}_{real}(k)\times n_p$의 평균($\overline{E}_{real}$)	표준편차	Z
P_1	8192	8192.6	90.5	0.9
P_2		8191.1	90.5	−1.1
P_3		8191.7	91.1	−0.3
P_4		8191.6	90.8	−0.5
P_5		8192.1	91.2	0.1
P_6		8191.8	91.0	−0.1
P_7		8191.5	90.3	−0.5

또한 Markov Cipher 가정하에 얻은 차분 경로의 기댓값이 동일한 2개의 차분 경로를 대상으로 차분 경로를 만족하는 실제 평문쌍의 개수가 다름을 보인다.

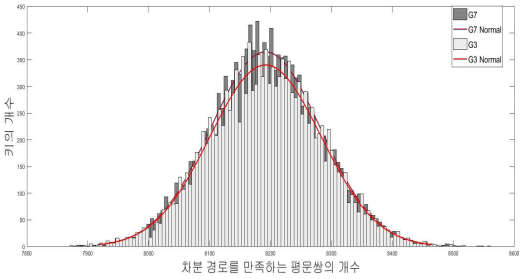
[표 3]의 G_3과 G_7을 바탕으로 귀무가설과 대립가설을 $H_0 : \overline{E}1_{real} = \overline{E}2_{real}, H_1 : \overline{E}1_{real} \neq \overline{E}2_{real}$라고 할 때, 유의수준 $\alpha = 0.01$에서 검정한다.

$$n_1 = 14336,\ \overline{X_1} = 12268.3,\ \sigma_1 = 111.0$$

$$n_2 = 14336,\ \overline{X_2} = 12297.7,\ \sigma_2 = 110.8$$

$$Z = \frac{(\overline{X}_1 - \overline{X}_2) - (\overline{E}1 - \overline{E}2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} = \frac{12268.3 - 12297.7}{\sqrt{\frac{111.0^2 + 110.8^2}{14336}}} = -22.4$$

$|Z| = 22.4 > Z_{0.005} = 2.58$이므로 H_0는 기각된다. 즉 G_3과 G_7은 동일한 기대 확률의 차분 경로지만, 실제 차분 경로의 확률이 다르다고 할 수 있다. 따라서 동일한 기대 확률의 차분 특성이지만, 실제로는 더 높은 확률의 평균값인 차분 특성을 이용한다면 공격 시 데이터 복잡도를 줄일 수 있다.

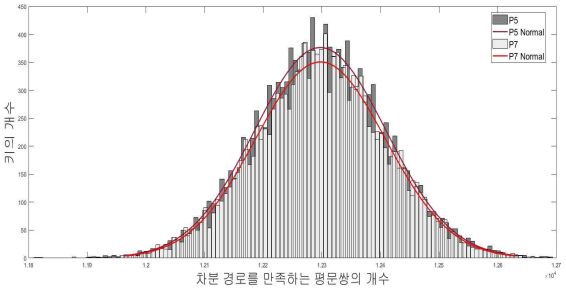


[그림 2] GIFT의 G_3과 G_7의 분포

4.3 차분 경로의 분포 그래프 비교

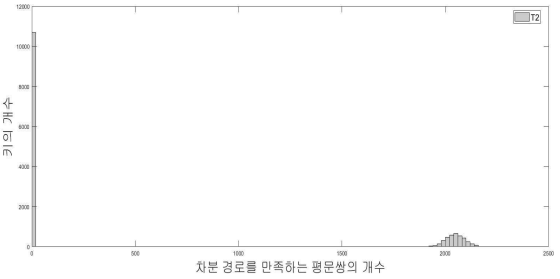
[8]의 논문에 따르면 블록암호의 차분 분포는 옳은 키를 가정할 때, 정규분포를 따른다고 한다. 그러나 키에 따라서 달라지는 차분 경로의 분포 그래프가 정규분포와 다른 분포의 형태가 있음을 보인다.

[그림 2, 3]은 PRESENT와 GIFT의 차분 경로의 분포로, 그래프의 형태가 정규분포를 따르는 것을 확인할 수 있다. [그림 3]은 PRESENT의 차분 경로 중 표준편차가 가장 큰 P_5와 가장 작은 P_7의 분포이다.



[그림 3] PRESENT의 P_5와 P_7의 분포

반면 GIFT 입출력이 같은 4라운드 차분 경로 T_2의 분포는 [그림 4]와 같다. 그래프의 모양이 정규분포를 따르지 않지만, 부분적으로는 정규분포를 따르는 것을 확인할 수 있다. T_2는 차분 경로를 만족하는 평문쌍의 개수가 0개인 키의 개수가 약 10000개 이상 존재한다. 즉 특정키에서는 차분 경로를 만족할 확률이 0이다. 따라서 부분적으로 정규분포를 따르는 형태의 차분 특성은 높은 확률을 갖더라도 특정키를 사용한다면 차분 공격에 내성을 가질 수 있다.



[그림 4] GIFT의 T_2의 분포

V.결론

본 논문에서는 CUDA를 통한 실험으로 Markov Cipher 가정하에 차분 경로의 기대 확률과 실제 차분 경로의 확률 비교 결과가 항상 일치하지 않는다는 사실을 확인하였다. 이를 통해 같은 확률의 차분 경로라 하더라도 실제로는 더 높은 확률로 특성을 만족하는 경로가 있음을 알 수 있었다. 이러한 차분 특성을 찾아 동일한 기대 확률이지만, 더 낮은 데이터 복잡도로 차분 공격을 할 수 있다. 또한, 키에 따라서 달라지는 실제 차분 경로의 분포가 반드시 정규분포를 따르지 않는다는 사실을 입증하였으며, 이런 경우 특정키를 이용하여 차분 공격에 내성을 가질 수도 있음을 보였다. 향후에는 이러한 특성을 바탕으로 기존 이론과 다른 값이 나온 원인을 파악하는 연구를 시행한다. 이를 위해 GIFT에 대한 다양한 실험을 통해 여러 가지 현상의 원인을 파악할 수 있을 것이다.

[참고문헌]

[1] Biham, Eli, and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." Journal of CRYPTOLOGY 4.1 (1991): 3-72.

[2] Lai, Xuejia, James L. Massey, and Sean Murphy. "Markov ciphers and differential cryptanalysis." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1991.

[3] Bogdanov, Andrey, et al. "PRESENT: An ultra-lightweight block cipher." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2007.

[4] Banik, Subhadeep, et al. "GIFT: a small PRESENT." International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, 2017.

[5] 김우철, et al. "현대통계학. 서울; 영지문화사." (2009).

[6] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture: Programming Guide (Version 7.0).

[7] Stevens, Marc, et al. "The first collision for full SHA-1." Annual International Cryptology Conference. Springer, Cham, 2017.

[8] Daemen, Joan, and Vincent Rijmen. "Probability distributions of correlation and differentials in block ciphers." Journal of Mathematical Cryptology JMC 1.3 (2007): 221-242.

MILP를 활용한 차분 공격 및 선형 공격관점에서의 PRINCE 블록 암호 분석※

전 용 진*, 이 종 혁**, 박 석 원**, 윤 형 준**, 최 예 림**, 김 경 란**, 박 태 진**, 조 은 영**, 김 종 성*,**

*국민대학교 금융정보보안학과

**국민대학교 정보보안암호수학과

요 약

차분 공격과 선형 공격은 DES(Data Encryption Standard)가 나온 이래로 가장 널리 퍼지고 효과적인 암호 공격 방법이다. 차분 경로와 선형 경로는 각각의 공격을 보조해주는 역할을 한다. PRINCE는 2012년 발표된 경량 블록암호이며, 4-bit Sbox를 사용한다. 이는 MILP(Mixed Integer Linear Programming)로 확률이 최대화된 차분 및 선형 경로를 실제로 볼 수 있게 해주는 요소가 된다. [1]에서는 Active Sbox를 활용하여 차분 및 선형 공격에 대한 확률을 어림짐작하였다. 본 논문에서는 PRINCE의 소개와 키를 고정하고 라운드 수를 줄인 PRINCE에 대한 차분 및 선형 경로 관점에서 분석한 과정과 결과를 서술한다.

Ⅰ. 서 론

PRINCE 블록 암호는 하드웨어의 경량화 구현을 목표로 만들어졌다[1]. 2012년 ASIACRYPT에서 발표되었으며, FX구조로 구성되어있다. 암호화 과정과 복호화 과정이 동일하나 Feistal 구조가 아닌 것이 특징이다. PRINCE는 연관키 공격이 활발하게 이루어져있으며 고정된 키에 관한 연구가 상대적으로 부족하다. 암호는 고정된 키를 사용하는 경우가 많으므로 이에 대한 많은 연구가 진행되어야 한다. 본 논문에서는 차분 공격과 선형 공격에 필요한 경로의 확률을 MILP를 사용하여 계산할 것이다.

MILP는 정수 변수들의 일차 결합으로 이루어진 식에 대한 최적화 알고리즘이다. 부등식 및 등식을 입력으로 받고 주어진 식을 만족하는 최적의 해를 출력한다. 본 논문에서는 MILP를 지원하는 도구인 CPLEX를 사용한다. CPLEX는 1988년 CPLEX Optimization Inc.에서 개발되었고 현재는 IBM에서 솔루션을 제공하고 있다.

※본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

차분 공격은 2개의 평문의 XOR값(이하 차분)과 각 평문으로 생긴 암호문의 차분을 사용하는 공격이다. 완벽한 난수라면 입력차분과 출력차분이 항상 독립적이겠지만 비선형 함수의 경우에는 될 수 없는 쌍이 존재한다. 이를 통하여 비선형 함수의 입력과 출력을 추정할 수 있다. 대표적으로 비선형 함수인 Sbox에 입력차분에 대한 출력차분을 표로 표현한 DDT(Distributed Differential Table)가 있다. 암호내에서 생성 가능한 차분들을 표현한 것을 차분경로라고 한다.

평문과 암호문과 키를 각각 P, C, K라 작성하고 표수가 2인 유한체상의 벡터로 보자. 임의의 마스킹 값 λ_P, λ_C, λ_K에 대해 식(1)을 만든다.

$$\lambda_P \cdot P \oplus \lambda_C \cdot C = \lambda_K \cdot K \qquad (1)$$

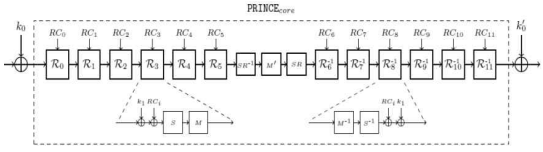
식(1)이 만족될 확률의 편차를 이용하여 공격하는 것을 선형공격이라 한다. 완벽한 난수라면 식(1)의 확률은 2^{-1}이 되어야하는데, 암호에서 사용하는 함수들에는 편차가 생기는 점들이 존재한다. 대표적으로 Sbox에 입력 마스킹에 대한 출력 마스킹을 표로 표현한 LAT(Linear Attribution Table)가 있다. 암호 전체에 대해서 바로 식(1)의 형태를 추측할 수 없으므로 암호화를 몇 개의 부분함수로 쪼개어 생각하는데, 이때 각 부분함수들의 중간 값들은 공격자 입장에서

확인할 수 없으므로 만들어진 식끼리 XOR하여 중간 값들이 사라지게끔 마스킹 값을 구축한다. 그 때 각 부분함수들의 마스킹 값을 표현한 것을 선형경로라 한다.

본 논문은 2장 PRINCE 블록 암호 소개, 3장 MILP 소개, 4장 차분공격, 5장 선형공격, 6장 결론으로 구성되어있다.

Ⅱ. PRINCE 블록 암호

PRINCE는 PRINCE core 함수 앞뒤에 라운드 키를 하나씩 XOR하는 구조를 갖는다. 본 논문에서는 키를 고정한 차분 공격 및 선형 공격의 관점에서 보고 있으므로 키 확장에 관한 내용은 서술하지 않는다.



(그림 1) 전체적인 PRINCE 구조

(그림 1)은 PRINCE의 구조를 도식화 한 것이다 〔1〕. 암호화의 앞, 뒤가 역함수를 사용한 대칭으로 이루어져있으며 라운드 키(RC)를 총 12개를 사용한다. 라운드 함수 R은 4-bit Sbox를 사용하는 S 함수와 행렬 곱을 사용하는 M함수, 상수 RC와 라운드 키의 XOR로 이루어져 있다.

PRINCE의 Sbox는 〔표 1〕과 같다. 각 값은 16진수이다. 이 Sbox는 DDT의 가장 큰 값이 4이며, LAT의 가장 편차가 큰 값은 ±4이다.

〔표 1〕 PRINCE Sbox

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S〔x〕	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4

M함수는 식(2)과 같다. SR은 AES에서 사용하는 Shift Row이며, 4-bit를 1-nibble이라 할 때, SR은 〔표 2〕와 같다.

$$M = SR \circ M' \tag{2}$$

〔표 2〕 SR(Shift Row)

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SR〔x〕	0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

M'은 2개의 16x16 행렬 $\widehat{M}^{(0)}$과 $\widehat{M}^{(1)}$을 사용하여 식(3)와 같이 이루어진다. 64-bit의 입력을 받으면 16-bit씩 쪼개어 16x1 행렬로 변환한 후 각각의 위치에 있는 행렬과 행렬곱을 실행하게 된다.

$$M' = (\widehat{M}^{(0)}, \widehat{M}^{(1)}, \widehat{M}^{(1)}, \widehat{M}^{(0)}) \tag{3}$$

각 행렬 $\widehat{M}^{(0)}$과 $\widehat{M}^{(1)}$은 4x4 소행렬 M_0, M_1, M_2, M_3을 사용하여 식(4)와 같이 나타내어진다.

$$\widehat{M}^{(0)} = \begin{pmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{pmatrix}, \widehat{M}^{(1)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{pmatrix} \tag{4}$$

4x4 행렬 M_0, M_1, M_2, M_3는 식(6), (7)과 같다.

$$M_0 = \begin{pmatrix} 0000 \\ 0100 \\ 0010 \\ 0001 \end{pmatrix}, M_1 = \begin{pmatrix} 1000 \\ 0000 \\ 0010 \\ 0001 \end{pmatrix} \tag{6}$$

$$M_2 = \begin{pmatrix} 1000 \\ 0100 \\ 0000 \\ 0001 \end{pmatrix}, M_3 = \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0000 \end{pmatrix} \tag{7}$$

〔1〕에서는 4개의 라운드 함수를 거치면 차분과 선형경로에 쓰이는 Active Sbox가 최소 16개가 된다는 것을 증명하였다. Active Sbox당 평균 차분 확률 및 선형 표준편차를 할당하여 차분 경로는 평균 2^{-96}의 확률, 선형 경로는 평균 2^{-49}의 확률로써, 대략적인 안정성을 계산하였다.

Ⅲ. MILP

MILP는 차분(혹은 선형)경로를 하나의 점으로 생각하여 점의 각 성분 간의 부등식을 사용하여 모든 경우에서 경로로써 가능한 점들을 걸러내는 방법이다. 하나의 암호에 대한 최적의 부등식을 한 번에 구할 수 없으므로 반복되는 구간의 부등식을 구해 전체 부등식을 구성한다. PRINCE는 Sbox와 M함수와 역함수로 이루어져 있는 데, 역함수는 각 함수의 부등식에 입력과 출력을 반대로 작성하면 된다. 따라서 Sbox의 부등식과 M'행렬의 부등식만 구하면 전체 부등식을 구성할 수 있다.

3.1. Sbox의 부등식 생성

MILP를 사용하기에 앞서, 확률을 셈하기 위하여 Sbox의 DDT를 순서쌍 (입력, 출력, p, q)로 출력한

다. 입력과 출력은 이진수로 표기한다. 각 입력과 출력에 대응되는 DDT의 값에 따라 p와 q를 정한다. DDT의 값이 2이면 p=1, q=0, DDT의 값이 4면 p=0, q=1이라 출력한다. 예외적으로 입력과 출력이 0일 때엔 p=0, q=0이라 출력한다. 전체 경로의 p의 값을 전부 더하면 확률이 2^{-3}인 Sbox의 개수이고, q의 값을 전부 더하면 확률이 2^{-2}인 Sbox의 개수이다. 따라서 다음의 식을 minimize하면 차분 경로의 최적 확률 경로가 된다.

$$3\sum_{\text{전체경로}} p + 2\sum_{\text{전체경로}} q \tag{8}$$

SAGE[1)]를 활용하여 부등식을 만든다. 상기의 방식으로 만든 모든 가능한 순서쌍을 SAGE에 polyhedron 함수를 사용하여 입력한다. 약 천개가량의 부등식이 출력되므로 부등식을 추려내는 작업이 필요하다. 먼저, 가장 많은 점들을 거르는 부등식을 찾아 출력한다. 걸러진 점들을 제외한 점들 중 가장 많은 점들을 거르는 부등식을 찾아 출력한다. 실존하는 경로(DDT에 존재하는 순서쌍)를 제외한 다른 모든 점들이 걸러질 때까지 이 과정을 반복한다.

LAT에 대해서도 확률부분을 제외하면 상기의 방법과 비슷하다. LAT에 작성되는 값들은 편차이므로 차분 확률과 비교하기 쉽게 하기 위해 상관관계를 계산한다. 편차가 ϵ일 때, 상관관계는 $4\epsilon^2$이 된다. 따라서 LAT의 값이 ± 2일 때엔 상관관계가 2^{-4}가 되고, LAT의 값이 ± 4일 때엔 상관관계가 2^{-2}가 된다. p와 q가 1이 되는 값을 각각 ± 2, ± 4라고 하면 다음의 식을 minimize하면 선형 경로의 최적 상관관계 경로가 된다.

$$4\sum_{\text{전체경로}} p + 2\sum_{\text{전체경로}} q \tag{9}$$

3.2. M함수의 등식 생성

M함수의 $\widehat{M}^{(0)}$행렬은 (그림 3)와 같다. $\widehat{M}^{(1)}$도 마찬가지로 표현할 수 있으므로 M함수의 출력의 각 성분은 입력의 세 성분의 XOR로 볼 수 있다.

1)범용적인 수학적 계산을 해주는 무료 도구이다.

$$\begin{pmatrix}
\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}
\end{pmatrix}
\begin{pmatrix}\begin{pmatrix}x_0\\x_1\\x_2\\x_3\end{pmatrix}\\\begin{pmatrix}x_4\\x_5\\x_6\\x_7\end{pmatrix}\\\begin{pmatrix}x_8\\x_9\\x_{10}\\x_{11}\end{pmatrix}\\\begin{pmatrix}x_{12}\\x_{13}\\x_{14}\\x_{15}\end{pmatrix}\end{pmatrix}
=
\begin{pmatrix}
\begin{pmatrix}x_4\oplus x_8\oplus x_{12}\\x_1\oplus x_9\oplus x_{13}\\x_2\oplus x_6\oplus x_{14}\\x_3\oplus x_7\oplus x_{11}\end{pmatrix}\\
\begin{pmatrix}x_0\oplus x_4\oplus x_8\\x_5\oplus x_9\oplus x_{13}\\x_2\oplus x_{10}\oplus x_{14}\\x_3\oplus x_7\oplus x_{15}\end{pmatrix}\\
\begin{pmatrix}x_0\oplus x_4\oplus x_{12}\\x_1\oplus x_5\oplus x_9\\x_6\oplus x_{10}\oplus x_{14}\\x_3\oplus x_{11}\oplus x_{15}\end{pmatrix}\\
\begin{pmatrix}x_0\oplus x_8\oplus x_{12}\\x_1\oplus x_5\oplus x_{13}\\x_2\oplus x_6\oplus x_{10}\\x_7\oplus x_{11}\oplus x_{15}\end{pmatrix}
\end{pmatrix}$$

(그림 3) $\widehat{M}^{(0)}$ 행렬

차분 값에 대해서 실제 경로의 XOR은 차분 값의 XOR이 된다. x_1, x_2, x_3을 XOR하여 x_4를 만들어 낸다고 가정하자. 그러면 임시 이진 변수 d_1, d_2에 대해 식(10)가 성립한다.

$$x_1 + x_2 + x_3 + x_4 - 2d_1 - 2d_2 = 0 \tag{10}$$

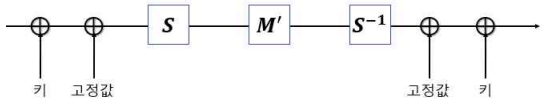
실제 경로의 XOR에 대해 연산의 모든 변수들의 선형 마스킹은 같게 된다[2]. 따라서 x_1, x_2가 하나의 XOR에 관여되었다고 하면 식(11)이 성립한다.

$$x_1 - x_2 = 0 \tag{11}$$

Ⅳ. 차분 공격

고정된 키에 대하여 4개의 라운드 함수를 거치면 차분 경로에서의 Active Sbox는 최소 16개가 된다는 것은 증명되었다[1]. PRINCE의 Sbox는 DDT의 가장 큰 값이 4이므로 하나의 Active Sbox에 대해 평균 확률이 2^{-3}이라 가정할 수 있다. PRINCE는 총 10개의 라운드함수를 거치므로 평균 확률이 2^{-96}이하가 된다. 그러나 만약 확률이 2^{-2}인 값으로만 최소 Active Sbox를 갖는 경로를 구성할 수 있게 된다면 PRINCE는 확률이 2^{-64}가 되고 라운드 수를 줄인 PRINCE는 그 경로에 대하여 취약하다.

PRINCE에는 라운드 키가 총 12개이다. 이를 12라운드라고 하고 라운드 수를 줄일 때는 앞과 뒤 라운드 함수를 각각 줄여 2라운드씩 감소하는 형태로 한다. 2라운드의 PRINCE는 (그림4)와 같다.



(그림 4) 2라운드 PRINCE

[표 3]은 2라운드와 4라운드에서 MILP로 측정한 차분경로의 최적 확률을 나타낸 표이다.

〔표 3〕 라운드 수에 따른 차분경로의 최적 확률

라운드수	2	4	6
확률	2^{-6}	2^{-24}(추정)	2^{-53}이상

4라운드는 2^{-11}이하 2^{-24}이상의 결과가 나왔으나 계산시 필요한 메모리의 양이 48기가가 넘어 더 이상의 계산은 불가능하였다. 4라운드의 차분 경로는 Active Sbox가 12개, 모든 Sbox의 확률이 2^{-2}가 되는 경로이다. 6라운드는 2^{-53}이상이라는 결과가 나왔고 4라운드와 마찬가지의 이유로 더 이상의 연산은 불가능 하였다.

Ⅴ. 선형 공격

〔표 4〕는 4장에서 한 것과 같이 라운드 수를 늘려가며 MILP로 측정한 선형 경로의 상관관계 표이다. 라운드 수가 5이면 PRINCE가 된다.

〔표 4〕 라운드 수에 따른 선형경로의 최적 상관관계

라운드수	2	4	6	8	10	12
상관관계	2^{-16}	2^{-128}	2^{-192}	2^{-256}	2^{-320}	2^{-384}

2라운드를 제외하곤 경로의 모든 값이 1로 출력이 된다. Sbox의 LAT에서 0xF -〉 0xF가 +4인 부분만이 경로에 사용되며 각 라운드의 모든 Sbox가 Active Sbox가 되었다. 이는 1라운드부터 상관관계가 2^{-64}을 넘으므로 PRINCE는 선형 공격이 불가능함을 보인다.

평문에 마스킹 값을 줘야하므로 첫 번째 라운드 함수 전에는 마스킹 값이 0이 아닌 nibble(AES와 PRINCE에서 사용하는 4-bit블록단위)이 최소 하나가 존재한다. 64-bit 기준에서 이 nibble의 위치를 A의 위치라 하고 A의 위치가 마스킹 값이 0이 아니면 A의 위치는 0이 아니다 라고 하자. Sbox를 거치면 A의 위치는 0이 아니게 된다. 따라서 M함수의 입력에서 A의 위치는 0이 아니다. XOR 연산에 참여하는 모든 변수의 마스킹 값이 같게 되므로 $\widehat{M}^{(0)}$과 $\widehat{M}^{(1)}$의 생김새에 의해서 M함수의 입력에서 A의 위치가 있는 16-bit블록단위에 있는 모든 nibble은 A의 위치의 마스킹 값과 같아지게 된다. 따라서 마스킹 값이 0이 아닌 nibble은 4개가 존재한다. 이는 평문에서도 똑같다. M'함수는 출력과 입력 마스킹 값을 보존하므로 마스킹 값은 변하지 않는다. 그 후 SR을 거치는데, 0이 아닌 4개의 각 nibble들이 한 개씩 다른 16-bit블록으로 퍼지게 된다. 다시 Sbox를 거친 후 M함수를 거치면 모든 16-bit블록이 마스킹 값이 0이 아닌 nibble을 가지게 되고 모든 nibble은 마스킹 값이 0이 아니게 된다. 이는 평문도 동일한데, Sbox는 nibble단위로 실행되므로 모든 Sbox가 상관관계가 존재하는 Active Sbox가 된다.

Ⅵ. 결론

본 논문에서는 PRINCE를 MILP를 사용하여 차분 경로의 최대 확률의 bound를 계산하고 선형 경로를 분석하였다. PRINCE는 SR함수와 XOR연산을 잘 활용하여 고정키에 대한 선형 공격에 대한 완벽한 안정성을 갖추었고 차분 공격에 대해서는 라운드 수를 늘려 안정성을 갖추었다.

최소한의 수식들을 사용하였지만 현재의 방법으로는 계산량에 한계가 있어 추정치밖에 얻을 수 없었다. 향후엔 MILP를 사용한 다른 공격법들을 개발하고 실질적인 공격을 연구할 것이다.

참 고 문 헌

〔1〕 Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın, "PRINCE – A Low-latency Block Cipher for Pervasive Computing Applications", *ASIACRYPT'12*, pp. 208-225, 1012

〔2〕 Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel, "Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming", *Information Security and Cryptology*, pp. 57-76, 2011.

확률 계산 방법에 따른 자동화 탐색의 효율성 분석*

이상협*, 김성겸*, 이용성*, 홍득조**, 성재철***, 홍석희*

*고려대학교 정보보호학과, **전북대학교 IT정보공학과, ***서울시립대 수학과

Efficiency analysis for automated searching with different probability calculation methods

Sanghyeop Lee*, Seong-Gyeom Kim*, Yongseong Lee*,

Deukjo Hong**, Jaechul Sung***, Seokhie Hong*

*Graduate School of Information Security, Korea University.

**Department of Information Technology, Chonbuk National University.

***Department of Mathematics, University of Seoul.

요 약

본 논문은 자동화 탐색을 이용해 차분 특성을 탐색할 때 차분 확률 계산 시 서로 다른 3종류의 확률 변수 설정 및 확률 계산법을 적용하여 효율성 및 장·단점을 분석한다. 이 방법을 4bit S-box를 사용하는 블록암호 PRESENT, Midori등에 적용하였다. 그 결과 탐색 속도 측면에서는 확률 값을 2진수로 표현하여 계산하는 방법이, 제약식 개수에서는 확률 비트에 각기 다른 가중치를 적용하는 방법이 효율적이었다. 각 확률 비트에 동일한 가중치를 적용한 방법은 덧셈과 S-box를 함께 사용하는 암호에 용이함을 확인하였다. 또한, 위 방법들에 각각 제약식을 간소화하는 Quine-McCluskey 알고리즘을 적용하여 탐색 시간을 단축시켰으며 이를 통해 CryptoSMT의 성능을 향상시켰다.

I. 서론

차분 분석(Differential Cryptanalysis)은 블록 암호 알고리즘을 분석하는 대표적인 분석 방법이다[1]. 블록 암호는 암호에 포함된 비선형 함수에 의해 입력 차분에 해당하는 출력 차분의 분포가 균일하지 않은 성질이 존재한다. 즉, 랜덤한 출력 차분 보다 더 높은 특정 출력 차분을 발생시키는 성질이 있다. 차분 분석은 이러한 성질을 갖는 입·출력 차분(디퍼렌셜, Differential)을 이용한 공격방법이다. 디퍼렌셜의 정확한 확률을 계산하는 것을 매우 어렵기 때문에 모든 라운드가 독립인 것으로 가정하여, 각 라운드의 입·출력 차분의 확률을 이용해 차분 특성(Differential Characteristic) 혹은 차분 경로(Differential Trail)의 확률을 계산한다.

비선형 함수의 출력 차분을 고려하면서 좋은 확률을 갖는 차분 특성을 수기로 구성하는 것이 어렵기 때문에 자동화 탐색 연구가 진행되었다[2,3]. 앞선 자동화 탐색 연구에서 임의의 문제를 수학적으로 모델링하여 해를 찾기 위한 도구들이 중요한 역할을 한다. 블록 암호 분석을 위해 두 가지 도구가 대표적으로 사용되는데 MILP(Mixed Integer Linear Programming) 기반과 SAT/SMT기반 도구이다. 이러한 각 도구는 수학적 문제를 해결하기 위해 개발되었기 때문에, 블록 암호 분석을 위한 여러 가지 연산이 추가적으로 필요하다. 그러한 연산들 중에서 차분 특성에 대한 확률을 계산하는 다양한 방법이 존재한다.

본 논문에서는 자동화 탐색에서 S-box에 대한 확률 계산 방법들을 소개하고 장점과 단점을 분석하여, 블록 암호에 사용되는 비선형 함수의 종류에 따라 최적의 확률 계산 방법을 선택할 수 있는 기준을 제안한다. 블록 암호에 대한 자동화 탐색은 부등식, 배열, 벡터 등의 입력 방식을 지원하는 SMT Solver로 만들어진 CryptoSMT[4]에 확률 계산 방법들을 적용하여 진행한다. 또, 제약식을 간소화(Logic optimization)하는 Quine-McCluskey 알고리즘을 사용하여 CryptoSMT의 탐색시간을 단축시켜 성능을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 SAT/SMT에 대해 설명하고, 차분 분석에 사용되는 차분분포표를 소개한다. 3장에서는 자동화 탐색에서 사용할 수 있는 확률 계산 방법들을 소개한 뒤 각 방법에 대한 장점 및 단점을 분석한다. 4장에서는 3장에서 소개한 확률 계산 방법들 중 동일한 가중치를 적용시키는 방법과 2진법을 사용하는 방법을 CryptoSMT에 적용하고, 표로 정리하여 차이점을 비교한다. 마지막으로 5장에서는 본 논문의 결론을 짓는다.

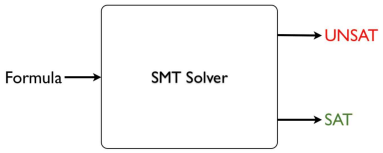
* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

II. 사전 지식

2.1 SAT/SMT

이진 충족 가능성 문제(Boolean Satisfiability Problem)는 이진 변수로 이루어진 논리식이 주어졌을 때, 논리식이 참이 되게 하는 해가 존재하는지 찾는 결정 문제이다. SAT Solver는 이진 변수의 개수, 이진 변수의 논리합(AND, logical disjunction)(절, Clause)의 개수, 절의 논리 곱(OR, logical conjunction)으로 표현된 논리식(CNF, Conjunctive Normal Form)을 입력받는다. 논리식이 참이 되게 하는 이진 변수 값이 존재할 경우 SAT (Satisfiability)를 반환하고 존재하지 않을 경우 UNSAT(Unsatisfiable)을 반환한다.

SAT Solver를 이용해 문제의 해를 구하기 위해서는 문제를 이진 변수를 이용해서 절의 논리곱으로 표현해야 한다. 하지만 문제를 이진수로 이루어진 절의 논리곱으로 표현하기 어렵기 때문에 SMT Solver가 개발되었다. SMT(Satisfiability Modulo Theories)[12]는 SAT의 확장으로 S-box의 입력/출력/확률 변수를 방정식으로 표현하는 등 다양한 변수의 1차 표현 방식을 입력으로 허용한다.



[그림 1] SMT Solver Architecture

SMT Solver를 사용할 때 가장 중요하게 고려되는 점은 제약식과 사용하는 변수의 개수가 있다. 제약식의 개수를 줄이는 방법으로는 논리식을 간소화(Logic optimization)하는 방법이 있다. 제약식을 간소화하는 알고리즘으로는 대표적으로 Espresso와 Quine-McCluskey 알고리즘이 존재한다. 본 논문에서는 탐색시간을 감소시키기 위해 Quine-McCluskey 알고리즘을 사용해 제약식을 간소화하여 탐색 시간을 감소시킨다.

2.2 차분 분석(Differential Cryptanalysis)

차분 분석은 입력 차분 값에 따른 출력 차분 값을 일정 확률로 예측할 수 있음을 이용하는 방법이다. 선형 구조에서는 입·출력사이의 차분의 변화를 1의 확률로 예측할 수 있고 비선형 구조는 일정한 확률로 차분의 변화를 예측 가능하다.

비선형 구조의 대표적인 예로는 S-box가 있다. S-box는 특정 비트의 값을 입력 받아 대응되는 특정 비트를 출력하는 비선형 함수이다. n비트가 입력되어 n비트가 출력되는 S-box에서 모든 입·출력 차분 값이 발생할 확률이 균일하다면 특정 입력 차분 값에 대하여 특정 출력 차분 값이 나올 확률은 $1/2^n$으로 동일해야 한다. 하지만 이러한 분포는 존재하지 않는다. 이런 성질을 이용하여 차분 분석이 진행되며 분석 대상의 모든 입력 차분 값과 가능한 모든 출력 차분 값을 표로 만들어 사용한다. 가능한 입·출력 차분의 분포를 표로 나타낸 것을 차분분포표(Difference Distribution Table, DDT)라 정의한다. $f: F_2^n \rightarrow F_2^m$인 S-box f가 존재하면 차분분포표 $D(f)$는 $2^n \times 2^m$인 행렬로 정의한다.

$$D(f)[\delta,\Delta] = \#\ \{x \in F_2^n, f(x+\delta)+f(x) = \Delta\}$$

행렬의 원소 값은 입력 차분이 δ인 입력 값 $x, x+\delta$를 각각 S-box에 대입한 출력 값 $f(x), f(x+\delta)$의 차분이 Δ를 만족하는 쌍의 개수이다.

Nyberg는 함수 $f: F_2^n \rightarrow F_2^n$에서 입력 차분과 출력 차분에 대하여 입력 차분이 0이 아닌 값 중 원소의 값이 가장 큰 값을 차분 균일성(Differential uniformity)이라고 정의하였다. 또한, f는 Δ_f-uniform이라 정의한다[5].

$$\Delta_f = \max\{D(f)[\delta,\Delta] : \delta \in F_{2^n}^*, \Delta \in F_{2^n}\}$$

III. 확률 계산 방법

수학적 문제들을 풀기 위한 도구로 개발된 각 Solver들은 차분 분석에 필요한 연산들이 존재하지 않는다. 따라서 사용자가 차분 특성을 찾을 수 있게 프로그램을 구성하여야 한다. 차분 특성을 찾기 위한 여러 가지 연산 중 본 장에서는 확률 계산 방법의 종류에 따라 장점과 단점을 분석한다.

3.1 각 확률 비트에 동일한 가중치 적용

CryptoSMT에서 확률 계산은 확률 변수 한 비트에 2^{-1}의 가중치를 두고, 확률 변수의 모든 확률을 곱하여 계산한다. 예를 들어 2^{-3}은 [0,1,1,1]로 표현되고 모든 비트를 더하는 것으로 확률의 지수 부분 표현이 가능하다. 4-uniform을 갖는 4bit S-box의 가능한 확률을 모두 표현하기 위해서 CryptoSMT는 총 4비트의 확률 변수를 사용하며, S-box 전체를 표현하기 위해서는 입력 4bit, 출력 4bit, 확률 4bit로 총 12bit가 필요하다.

Solver를 사용하기 위해서는 S-box의 차분분포표를 만들어 입·출력 차분에 대한 불가능한 경우를 모두 제약식으로 만들어 사용한다. 예를 들어 입력 차분이 $\Delta 1 \rightarrow \Delta 5$가 나올 확률이 $\frac{2}{16} = 2^{-3}$일 때, 가능한 경우가 [0,0,0,1/0,1,0,1/0,1,1,1](입력/출력/확률)라고 가정하자. 확률부분인 [0,1,1,1]을 제외한 모든 불가능한 경우는 [0,0,0,1/0,1,0,1/ 0,0,0,0],…, [0,0,0,1/0,1,0,1/0,1,1,0], [0,0,0,1/0,1,0,1/1,0,0,0],…, [0,0,0,1/0,1,0,1/1,1,1,1]로 모두 제약식으로 입력된다.

각 확률 비트에 동일한 가중치를 적용하는 방법의 장점은 4-uniform S-box를 포함한 확률이 2의 거듭제곱 꼴로 표현되는 비선형 연산에 대한 확률 계산을 할 수 있기 때문에 범용성이 좋다는 것이다. 하지만

단점은 $\frac{6}{16}=2^{-1.415}$와 같이 확률의 지수 부분이 소수로 표현되는 확률을 표현하기 위해서 더 많은 비트를 사용해야하고, 4비트의 정보를 확률로 나타내기 때문에 제약식이 많아 탐색시간이 증가한다는 것이다.

3.2 각 확률 비트에 모두 다른 가중치 적용

각 확률 비트에 모두 다른 가중치를 넣어 계산하는 방법은 S-box가 가지는 확률의 가짓수에 따라 사용하는 비트의 개수가 정해진다. 예를 들어 차분분포표의 확률이 총 3가지가 존재한고 가정한다. 그렇다면 확률 변수를 $[p_1,p_2,p_3]$로 3가지를 사용하고, 각각 p_1은 $\frac{2}{16}=2^{-3}$, p_2는 $\frac{4}{16}=2^{-2}$, p_3은 $\frac{6}{16}=2^{-1.415}$의 가중치를 준다. 모두 0인 경우는 확률 1이 되며, 최종적으로 확률 변수에 가중치를 곱하여 확률을 계산할 수 있다.

각 확률 비트에 모두 다른 가중치를 적용하는 방법의 장점은 확률이 갖는 경우의 수에 따라 변수의 개수가 달라진다는 것이다. 따라서 유동적으로 계산이 가능하여 최적화 변수의 개수를 가변적으로 정할 수 있고 $\frac{6}{16}=2^{-1.415}$과 같은 소수점 계산이 가능하다. 하지만 단점은 최종적으로 확률 계산할 때 덧셈 연산이 아닌 곱셈 연산을 사용하기 때문에 연산 부하가 생긴다는 것이다.

3.3 확률의 지수 부분 값을 2진수로 표현

확률의 지수 부분 값을 2진수로 표현하여 계산하는 방법은 확률의 지수를 2진법으로 변환하여 비트로 표현하는 것이다. 예를 들어 2^{-3}을 기존 CryptoSMT 방법은 [0,1,1,1]로 표현한 것을 [1,1]로 줄일 수 있다. 그러므로 4bit S-box를 사용하는 암호에서 12bit를 10bit로 변수의 개수를 줄여 전체적인 제약식의 개수를 크게 감소시킨다.

확률의 지수 부분 값을 2진수로 표현하는 방법의 장점은 4-uniform S-box에 대한 확률을 표현하는데 2bit의 정보를 사용함으로 제약식의 개수가 줄고 탐색 속도가 증가한다. 하지만 단점은 $2^{-1.415}$와 같이 지수 부분이 소수인 경우는 2진법으로 표현이 힘들기 때문에 확률을 계산하기 어렵다는 것이다.

IV. 적용 및 비교

본 장에서는 3장에서 소개한 방법을 적용하여 실험을 진행한다. 제약식의 개수를 비교할 때는 3종류의 방법을 모두 적용했다. 또한, 탐색 속도 비교는 동일한 가중치를 적용하는 방법과 확률의 지수 부분을 2진수로 표현하는 방법을 PRESENT와 Midori에 각각 적용하여 3라운드에서 8라운드까지 탐색을 진행한다.

[표 1] 4bit S-box의 제약식 개수

암 호	3.1절	3.2절	3.3절
GIFT[6]	-	1949	-
PRESENT[7]	3999	927	927
Midori[8]	3999	927	927
Skinny-64[9]	3999	927	927
Rectangle[10]	3999	927	927
Prince[11]	3999	927	927

4bit S-box를 사용하는 암호들에 적용 후 생성되는 각각의 제약식 개수는 [표 1]과 같다. [표 1]의 3.1절을 사용한 방법에서는 6-uniform S-box를 사용하여 지수 부분이 소수인 확률을 가지고 있는 GIFT를 제외한 암호들의 제약식의 개수가 모두 동일한 것을 알 수 있다. 3.2 절의 방법을 사용한 경우는 6-uniform S-box인 GIFT를 포함하여 4-uniform S-box를 사용하는 나머지 암호들도 모두 표현 가능한 것을 알 수 있다. 이때, 2가지의 확률 변수 $(2^{-2},2^{-3})$를 사용한 나머지 암호들보다 3가지의 확률 $(2^{-2},2^{-3},2^{-1.415})$을 나타내야 하는 GIFT의 경우 제약식의 개수가 더 많은 것을 확인할 수 있다. 3.3절의 방법을 사용한 경우 6-uniform S-box를 사용하는 GIFT를 제외한 나머지 암호들은 2가지의 확률 변수$(2^{-2},2^{-3})$를 사용하여 3.2절과 제약식의 개수가 동일한 것을 확인할 수 있다.

[표 2] CryptoSMT 차분 특성 탐색 시간 (초)

PRESENT			Midori		
Round	3.1절	3.3절	Round	3.1절	3.3절
3	242	52	3	433	76
4	712	139	4	1598	348
5	2169	496	5	2649	734
6	2455	809	6	4433	1357
7	2848	1016	7	7481	2348
8	3855	1155	8	9473	2997

CryptoSMT의 코드를 변경하여 차분 특성 탐색을 진행하여 차분 특성 탐색 속도를 [표 2]와 같이 정리하였다. [표 2]는 4-uniform S-box를 사용하는 대표적인 암호 중 PRESENT와 Midori를 대상으로 탐색을 진행하였다. 4가지 확률 변수를 사용하는 3.1절의 방식보다 2가지의 확률 변수를 사용하는 3.3절의 방식을 사용했을 때 차분 특성을 PRESENT의 경우 약 3.8배, Midori의 경우 약 3.9배 더 빠르게 차분 특성을 탐색할 수 있음을 확인하였다.

3.2절의 방법을 적용하여 비교하지 않은 이유는 다음과 같다. 3.1절과 3.3절의 경우 CryptoSMT의 확률 계산 부분에서 확률을 앞부분에 0을 붙여 8비트로 만들어진 확률을 더하여 계산한다. 따라서 정확한 차분 특성 탐색 속도를 비교하는 것이 가능하다. 그러나 3.2절은 곱셈을 이용해 확률을 계산하기 때문에 차분 특성 탐색 속도를 3.1절과 3.2절의 방법과 비교하는 것은 정확하지 않아 제외하였다.

[표 3] Quine-McCluskey 알고리즘 적용 후 탐색 시간 (초)

PRESENT			Midori		
Round	3.1절	3.3절	Round	3.1절	3.3절
3	25	5	3	41	8
4	38	10	4	135	48
5	84	34	5	354	175
6	146	77	6	1079	621
7	222	138	7	2036	1192
8	338	208	8	2904	1923

[표 3]은 PRESENT와 Midori의 3.1절의 방법과 3.3절의 방법 모두 Quine-McCluskey알고리즘을 적용하여 차분 특성 탐색 시간을 비교하였다. PRESENT의 경우 3.1절의 방법을 제약식 간소화 알고리즘을 사용한 후 약 15.8배 더 빠르게 차분 특성 탐색이 진행되는 것을 확인하였다. 3.3절의 방법은 약 10.3배 빠르게 탐색되는 것을 확인하였다. 마찬가지로 Midori는 3.1절의 경우 약 6.8배, 3.3절은 4.4배 빠르게 탐색이 진행되는 것을 확인하였다. 또 간소화 알고리즘을 적용한 후 3.1절과 3.3절은 PRESENT와 Midori 각각 약 2.7배, 2.4배정도 빠르게 탐색하는 것을 확인하였다.

V. 결론

본 논문에서는 자동화 탐색에 적용되는 확률 계산 방법들에 대한 분석을 진행하였다. 이를 통해 각 블록 암호가 사용하는 비선형 함수에 따라 확률 계산 방법을 적절하게 선택할 수 있다. 예를 들어, ARX 구조는 3.1절의 방법으로, GIFT와 같은 6-uniform은 3.2절의 방법을 사용하면 된다. 또 4-uniform S-box를 사용하는 암호에는 3.2절 또는 3.3절의 방법 중 선택하여 사용하면 보다 효율적인 차분 탐색이 가능하다. 그리고 PRESENT와 Midori에 확률 변수 개수에 따른 차분 특성 탐색 속도 비교 및 Quine-McCluskey 알고리즘 적용 이후의 탐색 속도를 비교하여 정리하였다. 그 결과 상황에 맞는 확률 변수의 개수를 설정하여 탐색 속도를 최적화 할 수 있다. 또한, 제약식을 간소화하는 방법으로 CryptoSMT의 성능을 향상시킬 수 있었다.

[참고문헌]

[1] Eli Biham and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." CRYPTO'90, Lecture Notes in Computer Science, vol. 537, pp.2-21, August, 1991

[2] Alex Biryukov and Vesselin Velichkov. "Automatic search for differential trials in ARX ciphers." CTRSA 2014, Lecture Notes in Computer Science, vol. 8366, pp. 227-250. Feb, 2014.

[3] Alex Biryukov and Ivica Nikolic. "Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia.", EUROCRYPT 2010, Lecyure Notes in Computer Science, vol. 6110, pp. 322-344, May, 2010.

[4] Stenfan Kölbl. "CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives." https://github.com/kste/cryptosmt, 2015.

[5] Kaisa Nyberg. "Differentially uniform mappings for cryptography." In Advances in Cryptology - EUROCRYPT'93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings, pages 55-64, 1993.

[6] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A Small Present," CHES, pp. 25 - 28, 2017.

[7] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450 - 466. Springer, Heidelberg, 2007.

[8] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: a block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 411 - 436. Springer, Heidelberg, 2015.

[9] Beierle, C., et al.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123 - 153. Springer, Heidelberg, 2016.

[10] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. 58(12), 1 - 15, 2015.

[11] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE - a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208 - 225. Springer, Heidelberg 2012.

[12] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satifsability modulo theories. Handbook of satisability, vol. 185, pp. 825-885, 2009.

Linux환경에서 SP800-90B 엔트로피 분석도구의 효율적인 호출 방안 연구

이태호*, 김현기**, 송행권**, 이옥연**

*국민대학교 정보보안암호수학과, **국민대학교 금융정보보안학과

A study of efficient calling measures of SP 800-90B Entropy analysis tool on Linux

Tae-Ho Lee*, Hyun-Ki Kim**, Haeng-Gwon Song**, Ok-Yeon Yi**

*Division of information security Cryptographic Mathematics, Kookmin University, **Division of Financial information security, Kookmin University.

요 약

안전한 암호 시스템을 구현하기 위해 필수불가결한 요소인 난수는 그 안전성을 평가하기 위해 진난수를 가공해 만드는 시드와 그 시드와 결정론적 알고리즘을 통해 만드는 의사난수에 대해 많은 평가 방법들이 제시된다. 그 중에서 미국의 NIST에서 제공하는 시드의 엔트로피를 평가하는 SP 800-90B(Second draft)를 기반으로 구현한 엔트로피 분석도구의 효율적인 호출 방안을 제시한다. 분석도구의 인수로 사용되는 스레드 수는 엔트로피 테스트 중에서도 IID-test 내부의 Permutation Test에서 사용된다. 스레드 수를 1개만 사용하면 분석하는 시간이 오래 걸려 효율이 떨어지는 면을 보완하기 위함이다. 가장 효율적인 분석속도를 제공하는 스레드 수를 찾고 이유를 분석한다.

I. 서론

난수는 안전한 암호 시스템을 구성하기 위해 필수불가결한 요소이다. 이는 난수발생기를 통하여 얻을 수 있으며, 암호학적으로 안전한 난수를 사용하기 위하여 예측불가능성, 비편향성, 비트간 독립성 등을 통계적 방법으로 해당 난수에 대한 테스트를 진행한다. 난수발생기는 물리적 잡음원을 입력으로 하는 진난수 발생기(TRNG, True Random Number Generator)와 시드와 알고리즘을 통해 의사난수를 발생시키는 의사난수 발생기(PRNG, Pseudo Random Number Generator)로 구성된다.

암호학적으로 안전한 난수발생기를 평가하는 방법은 PRNG를 통해 생성된 난수의 난수성 검정과 PRNG에 예측불가능성을 갖는 입력값으로 사용될 시드에 대한 엔트로피 테스트가 있다. 본 논문에서는 SP 800-90B에 언급된 엔트로피 평가 방법에 대해 기술하고, NIST에서 제공하는 엔트로피 분석도구를 이용하여 시드의 엔트로피 평가를 효율적으로 실행하는 방법을 제시한다.

본 논문은 5장으로 구성되어 있다. 2장에서는 기본적인 난수의 개념에 대해서 설명한다. 3장에서는 SP 800-90B 엔트로피 평가들에 대한 목적, 함수들을 설명하며, 그에 대한 소스를 제시한다. 4장에서는 실험에 대한 조건, 변인, 실험결과들을 제시하고 실험하는 과정과 결과를 분석한다. 마지막 결론에서는 가장 효율적인 스레드 수를 제시하고 스레드 수에 따른 효율을 다소 향상시킬 수 있는 방법을 제시하며 마무리한다.

II. 암호학에서의 난수의 중요성

암호를 만드는 데에 있어서 key, nonce 등의 핵심보안 매개변수는 암호 시스템 사용자 외에 아무도 예측할 수 없어야 한다. 사용하려는 난수가 암호학적으로 안전하다는 최소한의 보장을 받기 위해 표준에 제시되어있는 테스트들을

통과해야 한다.

2.1. 난수

난수는 현재 주어진 수를 통해 다음에 나올 수를 예측할 수 없는 성질을 가지는 무작위로 이루어진 수열을 말한다. 암호학에서의 난수는 비결정론적인 특성을 가지는 진난수와 결정론적인 특성을 가지는 의사난수로 구분되며 암호학적 난수 사용 시, 진난수를 암호학적 보안강도를 만족하는 수열 출력용으로 사용하지는 않는다. 진난수는 일반적으로 주변 환경의 물리적 잡음원을 획득하여 난수를 출력하는데, 이상적인 잡음원을 실제 시스템이나 환경에서 획득하는 것은 어려우며, 시간이 오래 걸리기 때문에 의사난수 생성기로 예측이 불가능하며 특정 패턴으로 치우치지 않는 암호학적으로 안전한 난수를 만들어야 한다.

2.2. /dev/random

UNIX계열 운영체제의 기본적인 난수 생성 방법에는 blocking 함수인 /dev/random과 non-blocking 함수인 /dev/urandom이 있다. 본 논문에서는 Linux 운영체제에 비해 생성이 용이한 Mac OS X의 /dev/random을 사용한다. 본 논문에서 밝히고자 하는 바는 엔트로피 분석 도구의 효율적인 호출 방안이므로, 해당 난수의 난수성은 논외로 한다.

III. SP 800-90B

본 논문에서는 SP 800-90B를 기반으로 만들어진 엔트로피 분석도구를 이용하여 엔트로피를 평가한다. 해당 분석도구의 목표는 실제 의사난수 생성기의 입력 값으로 필요한 엔트로피 수준을 충족시키기 위해 잡음원을 평가하는 것이다.

SP 800-90B 엔트로피 평가도구는 iid-test를 실행할 때, 사용할 스레드 수를 설정할 수 있다. 스레드 수는 실행 속도가 느려서 많은 실행을 진행하는 데 있어서 현실적으로 시간이 부족하기 때문에 여러 스레드 수를 이용해 시간을 단축시키는 데에 사용된다. Non-iid-test는 비교적 빠른 속도로 실행되어 스레드 1개로도 충분히 운용할 수 있는 반면에 iid-test는 스레드 1개를 사용하기에는 실행 속도가 오래 걸리기 때문에 여러 개의 스레드를 사용하여 운용하는 것을 추천하고 있다. 효율적인 운용 스레드 수를 알기 위해 iid-test(Permutation Test)의 스레드 수에 따른 속도 변화를 분석한다.

Permutation Test	Test 설명
Excursion	잡음원을 샘플 단위로 나누어, 샘플이 기댓값에서 벗어난 정도로 IID 여부 확인
Number of directional runs	연속된 샘플의 값을 1, 0, -1로 치환한 값의 런 개수로 IID 여부 확인
Length of directional runs	연속된 샘플의 값을 1, 0, -1로 치환한 값의 런 중 가장 긴 런의 길이로 IID 확인
Number of increases and decreases	연속된 샘플의 값을 1, 0 ,-1로 치환한 값에서 1과 - 1 중 많은 것의 개수로 IID 여부 확인
Number of runs based on the median	샘플의 중앙값을 기준으로 1과 -1로 치환된 값들의 런의 총 개수로 IID 여부 확인
Length of runs based on median	샘플의 중앙값을 기준으로 1과 -1로 치환된 값들의 런 중 가장 긴 런의 길이로 IID 여부 확인
Average collision	충돌의 평균값으로 충돌 빈도를 구해 IID 여부 확인
Maximum collision	Average collision과 같은 가정 하에 충돌의 최댓값을 비교
Periodicity	잡음원에 주기적인 성질이 얼마나 자주

	발생하는 지 확인
Covariance	기존의 잡음원과 셔플링된 잡음원의 공분산으로 IID 여부 확인
Compression	잡음원을 Bzip2으로 압축한 갑과의 차이를 이용해 IID 여부 확인

[표 1] Permutation Test 구성과 이용 통계량

IV. 실험

4.1. 실험 환경 및 방법

통제 변인	운영체제	LINUX 16.04 LTS
	지원 스레드 수	1 ~ 8
	파일 종류	/dev/random
	파일 크기	1MB
	실행 파일	SP 800-90B (2nd-draft)기반 IID-test 실행파일
조작 변인	사용 스레드 수	0(==1) ~ 8
종속 변인	사용스레드 수에 따른 실행 속도	-

[표 2] 실험 환경

실험은 총 2가지로서 IID-Test의 스레드 수별 속도 변화 분석을 진행하며 Permutation Test 내부에 속해 있는 각 Test들의 속도를 측정하고 차지하는 시간의 비율을 분석한다.

속도 측정 실험은 다음과 같이 진행한다. 각 Test를 나누어 실행하며, 이를 총 10번 반복하여 측정된 시간의 평균값을 통해 시간 비율을 분석한다.

4.2 실험 결과

내부 함수들의 Test 속도 비율을 보면 크게 Compression / Collision / Excursion, Directional, Consecutive_runs, Periodicity, Covariance 로 나눠지는데 이러한 구조는 스레드 4개에서 최고의 효율을 보이고 그 이후로는 효율적이지 않기 때문에 SP 800-90B에서는 4개의 스레드 수면 충분하다고 제시했다.

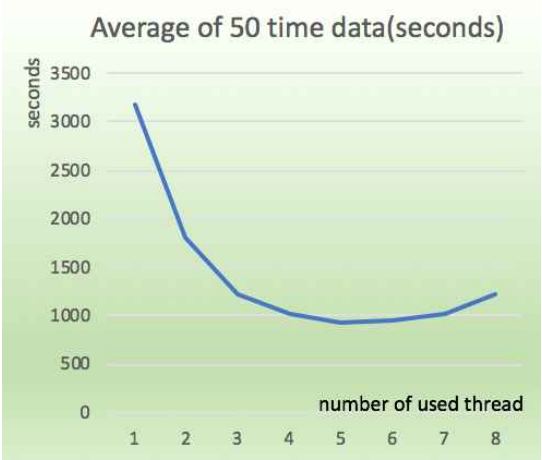
소스 코드 내에 있는 lock 함수, 즉 데이터 레이스를 막기 위한 함수에 의해 스레드 수가 늘어남에 따라 간혹 속도가 느려지는 경우도 존재한다.

소스 상 테스트	실제 수행 테스트
Excursion test	Excursion test
Directional test	Number of directional runs test Length of directional runs test Number of increases and decreases
Consecutive runs test	Number of runs based on the median test Length of runs based on the median test
Collision test	Average collision test Maximum collision test
Periodicity test	Periodicity test
Covariance test	Covariance test
Compression test	Compression test

[표 3] 함수 내부의 실제 수행 테스트

Test 명	스레드 당 소요 시간	소요 시간 비율
Excursion test	145(초)	$\frac{145}{3748} \times 100 = 3.87(\%)$
Directional test	138	$\frac{138}{3748} \times 100 = 3.68(\%)$
Consecutive runs test	144	$\frac{144}{3748} \times 100 = 3.84(\%)$
Collision test	549	$\frac{549}{3748} \times 100 = 14.65(\%)$
Periodicity test	98	$\frac{98}{3748} \times 100 = 2.61(\%)$
Covariance test	97	$\frac{97}{3748} \times 100 = 2.59(\%)$
Compression test	2577	$\frac{2577}{3748} \times 100 = 68.76(\%)$

[표 4] Permutation Test 내부 함수 속도 비율



[그림 1] 스레드 수 별 실행 시간 평균

V. 결론

스레드 수에 따라 속도 차이가 있으며, 4~5개의 스레드를 사용했을 때 최대의 효율을 보인다. 소스 코드는 운영체제가 달라져도 차이가 없기 때문에 최대 스레드 수, 사용 스레드 수가 같다면 운영체제의 처리 능력에 따라 속도 차이를 보일 뿐 걸리는 시간의 비율과 최대 효율을 보이는 스레드 수는 일정하다.

실험 결과에서 스레드 수가 늘어남에 따라 속도가 느려지는 현상을 보이는 이유인 lock함수를 제거하면 속도가 느려지는 현상을 없앨 수 있다. 그러나 lock함수를 제거하면 데이터레이스에 의한 계산 오류가 발생할 수 있기 때문에 Atomic Programming(원자적으로 프로그래밍하여 다른 스레드의 개입을 막는 방법), Refactoring(스레드 지역변수에서 원자적이지 않은 계산을 수행한 뒤, 데이터 레이스가 발생하는 전역변수 부분만 lock을 사용하는 방법) 등의 방법으로 데이터 레이스를 방지하면서 속도가 감소하는 현상을 바로잡고 스레드 수를 늘린다면 SP800-90B의 IID-Test 실행파일의 효율성 증대 및 속도 개선을 기대할 수 있을 것이다.

[사사 정보]

본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

[참고문헌]

[1] A. Rukhin, et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22 Revision 1a, Apr. 2010

[2] 최희봉 등 7명. "운영체제별 잡음원 수집 및 응용 지침", TTAS.KO-12.0235/R1. Jun. 2017

[3] M. Turan, et al., "Recommendation for the Entropy Sources Used for Random Bit Generation", (Second DRAFT) Special Publication 800-90B. Jan. 2016

[4] 박호중. 엔트로피 소스에 대한 통계적 난수성 검정 방법 연구. RISS 학위논문. 2016

완전동형암호화를 사용한 암호화된 데이터베이스 시스템†

김태윤*, 김형식*

*성균관대학교 전자전기컴퓨터공학과

Querying encrypted databases using fully homomorphic encryption

Taeyun Kim*, Hyoungshick Kim*

*Department of Computer Science and Engineering, Sungkyunkwan University.

요 약

암호화된 상태로 사용자의 데이터를 데이터베이스에 저장하는 것은 사용자의 데이터의 기밀성을 제공하지만 사용자가 실행하고자 하는 쿼리에는 답할 수 없게 된다. 본 논문에서는 완전동형암호화 라이브러리인 Fast Fully Homomorphic Encryption over the Torus (TFHE)를 사용하여 암호화된 값을 MySQL 프록시를 사용하여 MySQL 데이터베이스에 저장하는 방법을 소개한다. MySQL 프록시가 주어진 쿼리를 암호화된 데이터베이스에서 실행될 수 있는 쿼리들로 변환시켜주기 때문에 사용자는 평문 상태의 데이터를 저장하는 데이터베이스에서 쿼리하는 것과 마찬가지의 방법으로 암호화된 데이터베이스에 쿼리할 수 있다. 구현된 시스템은 DBMS 서버가 악의적이다 하더라도 사용자의 데이터를 유출시키지 않고 사용자에게 제공할 수 있다. 또한, 데이터를 완전동형암호화시켜 저장하기 때문에 사용자가 제공하는 어떠한 쿼리도 구현된 시스템을 확장하여 답할 수 있을 것으로 보인다. 구현된 시스템은 16 비트 정수를 대략 0.27초 안에 암호화하고 저장하며 약 2.045초 안에 16 비트 정수를 추출하고 복구화한다.

I.서론

사용자의 데이터를 공격자들로부터 안전하게 보호하기 위해 보통 암호화된 상태로 데이터를 데이터베이스에 저장하게 된다. 여기서 발생하는 문제점은 암호화된 데이터베이스에서는 평문 상태의 데이터베이스에서 쿼리를 행하는 방식과 같은 방법으로는 쿼리를 행할 수가 없다는 점이다. 사용자의 데이터를 안전하게 보호하면서 데이터베이스 쿼리 또한 가능케 하기 위해 아무런 암호화를 적용하는 것이 아니라 deterministic 암호화, order-preserving 암호화, 동형암호화 등을 적용을 할 수 있다. 여기서 암호화가 deterministic하다는 것은 같은 값의 평문이 같은 키를 사용하여 암호화되었을 때 항상 동일한 값의 암호문이 생성이 되어 암호문만을 가지고 평문값의 equality를 판별할 수 있다는 것을 의미한다. 또한 암호화가 deterministic하면서 암호화되었을 때에도 평문 상태에서의 순서를 유지할 때 암호화가 order-preserving하다고 한다. 동형암호는 암호화된 데이터상에서 연산을 가능케하는 암호화이다.

사용자의 데이터를 보호하기 위해 암호화 뿐만이 아니라 인텔 SGX 하드웨어 enclave와 같은 하드웨어 유틸리티나 Function Secret

† 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업(IITP-2018-2015-0-00403), 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터(No.2017-0-00545, 암호화된 데이터상의 기계학습 라이브러리 설계 및 활용기법 연구)의 지원을 받아 수행되었음

Sharing을 사용하는 연구도 존재한다. 본 논문에서는 이와 같은 사용자의 데이터를 안전하게 보호하면서 데이터베이스의 쿼리를 가능케 하는 사전 연구를 소개하고 완전동형암호화 라이브러리인 Fast Fully Homomorphic Encryption over the Torus (TFHE)와 MySQL 프록시를 사용하여 암호화된 사용자의 데이터를 효율적으로 MySQL 데이터베이스에 저장하면서 암호화와 복구화를 효과적으로 할 수 있는 방법 또한 소개한다.

동형암호 [2]는 암호화된 데이터상에서 암호화를 풀지 않고 연산을 가능케 하는 암호방식이다. 따라서, 이러한 동형암호를 사용하면 암호화된 데이터베이스에서 복구화를 하지 않고 쿼리를 행할 수 있게 한다. 동형암호에는 부분동형암호와 완전동형암호가 존재를 하는데 부분동형암호는 암호화된 데이터베이스에서 덧셈이나 곱셈과 같은 특정 연산만 지원하는 반면 완전동형암호는 암호화된 데이터 상에서 어떠한 연산도 지원을 한다. 따라서, 이러한 완전동형암호화를 사용하면 사용자가 User-Defined Function (UDF)을 사용하여 정의하는 어떠한 쿼리도 암호화된 데이터베이스에서 행할 수 있게 된다.

본 논문에서 사용한 TFHE 라이브러리는 주어진 비트의 암호화 복구화를 제공하고 암호화된 비트간의 게이트 연산을 지원을 한다. 비트 단위에서의 게이트 연산 함수만 제공하기 때문에 구현하고자 하는 쿼리 연산이 있으면 게이트 연산 함수들을 사용하여 직접 구현해야 한다. 또한, 암호화 복구화에는 secret key를 사용을 하고 암호화된 데이터상에서의 연산을 할 때는 cloud key를 사용을 하는데 cloud key를 소지하고 있더라도 secret key 없이는 복구화가 불가능하기 때문에 cloud key만 데이터베이스 서버에 제공하고 secret key는 MySQL 프록시에게만 제공하게 되면 MySQL 프록시가 장악되지 않는 이상 데이터베이스 서버가 악의적이다 하더라도 사용자의 데이터가 유출되지 않는다.

II. 관련 연구

암호화된 데이터상에서 데이터베이스 쿼리를 행하는 방법에 대한 연구는 [1]에서 서술되어 있다. 논문에서는 암호화된 데이터에서의 데이터베이스 쿼리를 제공하는 CryptDB라는 시스템을 제안한다. 데이터베이스 스키마의 모든 콜럼에 대해서 여러 개의 암호화 layer를 구성하여 이를 달성한다. CryptDB에서 사용되는 암호화 layer는 RND, DET, OPE, HOM, JOIN, OPE-JOIN, SEARCH를 포함한다.

여기서 각각의 암호화 방식은 다른 보안 강도와 다른 기능을 제공을 한다. 예를 들어, Homomorphic encryption (HOM)은 필드값을 동형암호화 시키는데 CryptDB에서는 덧셈에 관한 부분동형암호화 중 하나인 Paillier 암호화를 사용한다. 따라서, 어떠한 쿼리도 가능케하는 것이 아니라 암호화된 상태에서 SUM과 같이 덧셈으로 표현될 수 있는 쿼리만 가능케 한다. 본 논문에서는 TFHE를 사용하여 어떠한 쿼리도 가능케할 수 있는 시스템을 제안한다.

III. TFHE를 사용하여 암호화된 데이터의 저장과 추출

완전동형암호화 라이브러리인 TFHE를 사용하여 데이터를 암호화 할 경우 암호화된 데이터를 MySQL 데이터베이스에 직접 저장할 수가 없다는 점이 존재한다. 이는 C++로 작성된 TFHE를 사용하여 비트를 암호화하였을 때, LweSample이라는 C++ struct가 반환이 되기 때문이다. 이 struct는 500개의 정수를 포함하고 있는 a 배열, 1개의 정수인 b, double 타입인 variance 필드값을 포함한다. MySQL 데이터베이스에 저장하기 위해서는 이 모든 필드값들을 저장해야 한다.

암호화된 값들을 저장하기 위해 LweSample struct의 모든 필드값에 대해서 콜럼을 할당하고 어떠한 사용자의 값이 저장되어 있는지 기록하기 위한 userid 필드값 또한 저장을 한다. 공격자가 이러한 userid 값을 얻음으로써 아무

유용한 정보를 얻지 못한다고 가정을 한다면 평문 상태로 userid를 저장할 수 있고 공격자에게서 이러한 userid 값 또한 감추고 싶다면 암호화된 상태에서 equality를 판별할 수 있도록 deterministic한 암호화를 사용하여 userid를 암호화할 수 있다.

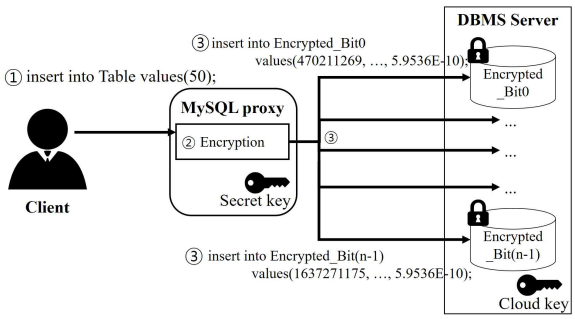
table	userid	0th_a	...	499th_a	b	variance
bit0	1	470211269	...	-1455671213	1110023459	5.9536E-10
...	...	...	...	...	...	...
bitn-1	1	1637271175	...	-463281669	-602048671	5.9536E-10

[그림 1] 암호화된 정수를 저장하는 예시 데이터베이스

하나의 암호화된 정수값을 저장하는 데이터베이스의 예시는 [그림 1]에 표현되어 있다.

암호화된 정수값뿐만 아니라 암호화된 double 데이터도 같은 방식으로 데이터베이스에 저장할 수 있다. Double 데이터의 정수 부분과 소수 부분을 표현하는 비트들을 각기 다른 테이블들에 저장하여 임의의 double 데이터를 저장할 수 있다.

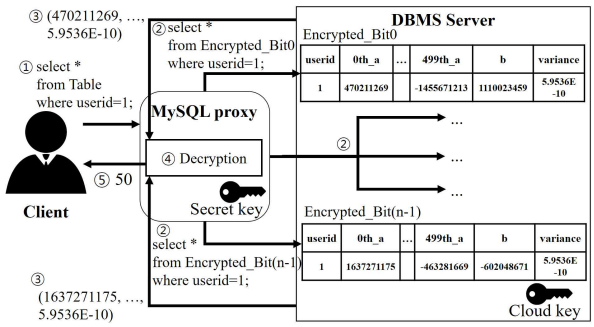
[그림 2]는 MySQL 프록시를 사용하여 사용자가 평문 상태로 전달한 쿼리를 암호화된 데이터베이스에서 실행될 수 있는 쿼리들로 변환해주는 것을 보여준다.



[그림 2] MySQL 프록시를 사용하여 암호화된 데이터베이스에 저장하는 과정

사용자가 제공한 평문 상태의 쿼리에 포함된 값을 MySQL 프록시가 secret key를 사용하여 암호화하게 되면 평문 값을 표현하는 데 사용된 비트 수만큼의 LweSample들이 형성이 되는데 각각의 LweSample들의 필드값들을 포함하고 있는 쿼리를 각각의 비트 테이블에 저장을 하게 된다.

[그림 3]은 MySQL 프록시와 select 쿼리를 사용하여 저장된 암호화된 데이터를 복구화하여 사용자에게 전달해주는 것을 보여준다. 사용자가 평문 상태의 select 쿼리를 보내면 MySQL 프록시가 암호화된 데이터베이스에서 실행될 수 있는 형태로 변환 된 쿼리들을 보내어 암호화된 필드값들을 받게 된다. MySQL 프록시는 받은 필드값들을 이용하여 LweSample들을 형성을 한 후, secret key를 이용하여 복구화하여 비트값들을 추출한 후, 이 비트값들을 이용하여 평문 상태인 사용자의 데이터를 얻을 수 있게 된다. 이러한 시스템은 사용자가 평문 상태의 데이터베이스에서 쿼리를 행하는 것과 동일하게 암호화된 데이터베이스에서 쿼리를 행할 수 있게 해준다.



[그림 3] MySQL 프록시를 사용하여 select 쿼리를 암호화된 데이터에서 행하는 과정

MySQL은 Lua 스크립트를 사용하여 프록시가 구현될 수 있게 해준다. 사용자가 제공한 SQL 쿼리들을 인터셉트하고 암호화된 도메인에서 실행할 수 있는 형태로 변환시켜주기 위해 역시 Lua 스크립트를 사용하여 구현하였다. DBMS 서버 쪽에서는 TFHE 라이브러리를 import하는 C++ 코드를 컴파일하여 생성된 .so 파일을 사용하여 사용자가 쿼리를 정의할 수 있게 해주는 User Defined Function (UDF)을 구현할 수 있다.

IV. 실험 결과

구현된 결과물을 테스트하기 위해 Intel Core i5-7500 CPU with 3.40GHz * 4와 64비트 우분투 16.04 LTS를 사용하였다.

구현된 결과물은 MySQL 데이터베이스에

TFHE를 사용하여 암호화된 LweSample의 필드값들을 저장할 수 있게 해준다. 16 비트의 정수를 앞서 설명된 스키마와 방법을 사용하여 암호화하고 insert하는 데 대략 0.27초 정도가 소요되었고 저장된 암호화된 16 비트의 정수를 select 쿼리를 사용하여 검색하고 암호문을 복구하는 데 2.045초 정도가 소요되었다. 또한, 32,000개의 행을 가지고 있는 지문 데이터셋인 "NIST special database 9"이 이미 암호화되어 있다고 가정하고 모든 행들을 데이터베이스에 저장하는데 14시간 정도가 소요되었다. userid 필드값까지 포함하여 16개의 정수를 가지고 있는 지문 벡터를 암호화하고 데이터베이스에 저장하는 데 27.07초 정도가 소요되어 암호화 과정까지 소요되는 시간에 포함하게 되면 더 오랜 시간이 소요되게 된다. 암호화를 하지 않고 동일한 데이터셋을 평문 상태로 저장을 하게 된다면 3분정도 밖에 소요되지 않았다.

또한, "NIST special database 9"을 16개의 double을 포함하고 있는 지문 벡터를 사용하고 정수 부분을 16개의 비트 수, 소수 부분을 16개의 비트 수를 사용하여 표현하여 저장하였을 때 29시간 정도가 소요되었다. 이는 같은 데이터셋을 지문 벡터를 정수로 표현하여 저장하였을 때보다 약 2배정도인데 소수 부분의 저장 또한 정수 부분을 저장하는 방법과 동일한 방법으로 실현되기 때문에 저장하는 데 소요되는 시간이 데이터를 표현하는 데 사용되는 비트 수에만 비례하여서 이러한 결과가 발생한다.

TFHE 라이브러리를 사용하여 암호화된 비트를 저장하기 위해서 501개의 정수와 하나의 double을 저장하기 때문에 암호화된 값을 데이터베이스에 저장할 때 사이즈가 증가할 수 있다. MySQL 데이터베이스를 사용할 때 지문 데이터셋인 "NIST special database 9"이 가지고 있는 행의 수와 같은 수의 행을 암호화한 상태로 저장을 하였을 때와 평문 상태로 저장을 하였을 때의 데이터베이스 사이즈를 기록하였다. 앞서 설명된 테이블 스키마를 사용하여 32,000개의 행을 각각의 테이블에 삽입하였을 때 하나의 비트를 표현하는 각각의 테이블이 71.88 MB 정도를 가진 반면 평문 상태로 동일한 데이터셋을 저장할 때는 0.23MB 정도의 사이즈를 가졌다. 따라서, 암호화된 상태로 저장할 때, (71.88 * 16) / 0.23 ≈ 5,000배 정도 사이즈가 증가한다는 것을 알 수 있다. 암호화된 double 데이터를 저장할 때, double의 정수 부분과 소수 부분을 표현하는 총 비트 수에 비례하게 데이터베이스 사이즈가 증가하는 것을 확인할 수 있었다. 예를 들어, double의 정수 부분과 소수 부분을 같은 수의 비트를 사용하여 표현하였을 때, 정수 부분만 저장하였을 때보다 2배 정도의 데이터베이스 사이즈를 요구하였다.

V. 결론

본 논문에서는 사용자의 데이터를 암호화된 상태로 저장하고 복구화하여 평문 상태로 데이터를 추출할 수 있는 방법에 대하여 서술하였다. 완전동형암호화 라이브러리인 TFHE와 MySQL 프록시를 사용하여 DBMS 서버가 악의적이라 하더라도 프록시가 안전한 이상 사용자의 데이터를 보호할 수 있는 시스템을 설립하였다. 또한, 암호화된 상태로 데이터베이스에 저장하였을 때와 평문 상태로 저장하였을 때의 시간과 데이터베이스 사이즈를 비교하였다. 본 논문에서 제시된 시스템을 확장하면 MySQL의 UDF와 TFHE를 컴파일시켜 얻은 .so 파일을 이용하여 사용자가 제공하는 어떠한 쿼리도 암호화된 데이터베이스에서 실행할 수 있다.

[참고문헌]

[1] Popa, R.A., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. pp. 85 - 100. ACM (2011)

[2] Craig Gentry. 2009. A Fully Homomorphic Encryption Scheme. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Boneh, Dan. AAI3382729.

Homomorphic encryption 라이브러리를 이용한 암호화 바이너리의 String signature 탐색 구현

황태윤*, 윤지원*

*고려대학교 정보보호대학원

Implementation of String signature search for encrypted binary using homomorphic encryption library

Tae Yoon Hwang*, Ji Won Yoon*

*Graduate school of information security, Korea University.

요 약

4차 산업이 발전함에 따라 대량의 데이터들을 처리하고 분석하는 업무가 중요해지고 있고 많은 기업에서 이러한 정보의 유출을 막기 위해 노력하지만 발전해가는 해킹공격으로 인해 유출되는 정보는 꾸준히 증가하고 있다. 그리고 분석이 필요한 개인 정보에 관한 데이터들과 바이너리들은 평문 상태에서의 분석을 수행하게 되면 내부의 정보가 공개될 수 있어 데이터에 대한 보안을 지킬 수 없게 된다. 따라서 본 논문에서는 특정 바이너리를 통해 암호화 및 복호화를 구현하고 암호화된 데이터 내부에서 문자열로 이뤄진 데이터를 탐색하는 방식을 구현한다.

I. 서론

클라우드 환경이 발달함에 따라 단순히 바이너리나 데이터에 대한 분석만이 아닌 분석을 진행할 데이터에 대해서 보안을 유지하는 것도 중요하게 여겨지고 있다. 바이너리 구조와 데이터에 관한 정보를 노출시키지 않고 분석을 진행해야한다면 데이터에 대한 암호화가 필요하다. 그리고 암호화가 되어있는 데이터끼리의 연산이 가능한 동형암호화 방식 [1]을 이용해야 한다.

만약 동형암호화 방식을 이용해서 내부 데이터에 대한 암호화를 진행한다면 암호문끼리의 연산이 가능해져 바이너리나 데이터의 내용을 알지 못해도 연산을 통한 분석이 가능하다. 따라서 본 논문에서는 테스트 바이너리에 대한 암호화를 수행하고 서버측에서 정의된 문자열 패턴을 암호화된 파일 내부에서 탐색을 수행하는 프로그램을 구현하였다.

II. 사전 지식

2.1 동형암호

동형암호는 암호화된 데이터 끼리의 연산이 가능한 암호화 방식을 말한다.

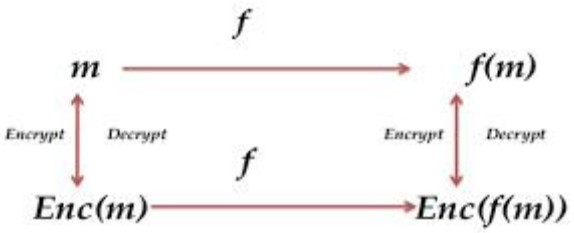


그림 1. Homomorphic encryption

그림 1.에서 보이는 것처럼 암호문끼리의 연산은 같은 데이터의 평문 연산과 동일한 결과를 얻을 수 있다. 처음 동형암호는 암호문끼리의 연산이 계속될수록 노이즈가 증가하여 특정

횟수를 넘어서면 복호화를 할 수 없다는 문제가 발생하였지만 2009년 동형암호의 연산과정 중 노이즈를 없애는 Gentry의 bootstrapping 기법을 소개하면서 암호문끼리의 연산이 진행될 때마다 노이즈 값이 늘어나 온전한 결과에 영향을 미쳤던 이전의 문제를 해결한 완전동형 암호 방식으로 발전됐다.

2.2 TFHE 라이브러리

동형암호에 관한 라이브러리는 Microsoft에서의 SEAL[2], MIT의 HElib[3] 등이 개발되었지만 실험에 사용할 TFHE 라이브러리[4]는 Ring-variant를 구현한 FHEW[5]를 발전시킨 형태의 라이브러리이고 FHEW보다 속도가 더 빠르도록 구성되어 있다. TFHE 라이브러리는 암호화시킬 데이터를 비트단위로 연산하여 한 bit당 ciphertext array의 한 칸에 담긴다. 이 배열을 이용하여 논리게이트(AND, OR, NOT 등)를 이용한 연산을 실행할 수 있으며 해당 게이트를 한번 연산할 때마다 bootstrapping 단계를 진행하여 연산을 수행하는 과정에서 암호화 노이즈를 해결하였다.

Ⅲ. 바이너리 암호화 및 패턴 탐색

3.1 바이너리 암호화

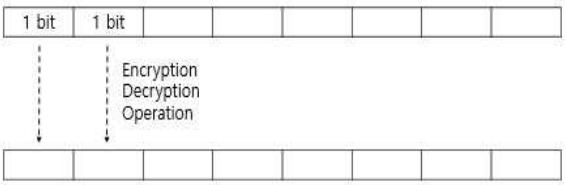


그림 2. LweSample Array

본 문서에서는 Homomorphic encryption을 수행하기 위해서 TFHE 라이브러리를 사용한다. 해당 라이브러리를 사용하여 암호화를 진행할 때는 하나의 bit를 LweSample이라는 배열 구조체의 한 칸에 넣는다. 이 구조체는 한 비트에서의 암, 복호화 및 gate를 이용한 연산이 가능하다. 암호화를 하게 되면 하나의 배열만큼 암호화를 진행하게 된다. 여기서는 크기가 일정하지 않은 바이너리에 대한 범용적인 암호화를 적용하려면 byte 단위로 암호화를 진행하는 것이 가장 효율적이다. 따라서 한 칸에 1bit를 나타내는 LweSample 배열을 8칸 생성하게 되면 암호화된 1bit의 정보를 8개를 담을 수 있어 1byte 단위로 암호화를 수행하게 된다. 이때 8bit의 데이터가 암호화된다면 총 16128 bytes의 자료 크기를 갖게 된다.

3.2 패턴 암호화

패턴 비교를 수행하기 전 서버에서도 가지고 있는 문자열을 암호화시켜야 한다. 하지만 서버 측에서는 암호화를 수행하는 비밀키가 존재하지 않기 때문에 클라이언트 측에서 0x00 ~ 0xFF까지의 암호문을 제공한다. 따라서 제공된 암호화 데이터를 가지고 C++의 vector 형태로 LweSample로 이뤄진 암호화된 패턴을 조합하여 비교를 수행한다.

3.3 패턴 탐색

서버측에서는 클라이언트가 보낸 암호화된 바이너리 정보와 패턴에 대한 암호문을 가지고 있기 때문에 TFHE 라이브러리에서 제공한 gate연산을 이용하여 암호화된 데이터간 비교연산을 수행하는 로직을 구현하였다.

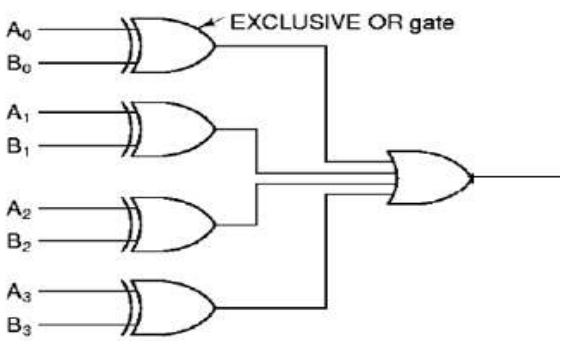


그림 3. Compare logic

각 bit마다 비교는 XOR, OR gate를 이용한 비교 로직을 사용하며 구현한 프로그램에서는 8bit를 기준으로 비교를 하기 때문에 1byte를

비교했을 때 둘의 데이터가 같다면 0, 다르다면 1이 암호화된 상태로 출력되며 해당 결과를 클라이언트에 보내 비밀키로 복호화를 진행한 후 결과물을 알 수 있다.

Ⅳ. 실 험

4.1 실험 환경

하나의 데스크탑에서 진행을 하여 클라이언트와 서버측의 자원은 같은 것을 사용하였다. Ubuntu 16.04.03 LTS 버전, Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz, 16GB RAM을 사용하였다. 실험은 printf()함수를 사용한 단순한 문자열을 출력해주는 프로그램을 타겟 프로그램으로 사용하였다. 그리고 찾는 패턴은 프로그램에서 사용한 함수명인 "printf"인 6글자의 문자열을 탐색하도록 패턴을 작성하였다.

4.2 실험 결과

찾는 패턴은 "printf"라는 단순한 6글자의 문자열 형태이다. strings 명령어를 사용하여 타겟 프로그램에 대한 문자열 시그니쳐를 뽑았을 때 "printf", ".edata"라는 6글자의 문자열이 2개 나온다. 따라서 길이가 같은 문자열에 대한 비교가 2번 수행이 된다. 6글자에 대한 2번의 비교 수행 간 57초가 소요되며 복호화를 통해 나온 결과는 총 12byte의 0과 1로 이뤄진 파일이 나오게 된다.

p	r	i	n	t	f
0x00	0x00	0x00	0x00	0x00	0x00
.	e	d	a	t	a
0x01	0x01	0x01	0x01	0x00	0x01

그림 4. 각 문자열 간 비교 결과

Ⅴ. 결 론

Cloud 시스템이 발전할수록 그에 따른 데이터의 보안도 중요해진다. 이 논문에서 구현한 암호화된 상태에서의 시그니쳐 탐색 방식은 아주 기본적인 바이너리의 특징인 문자열 추출에 대해 서술하고 있지만 이러한 기능은 암호화된 상태에서의 데이터 탐색에 대한 가능성을 제시한다. 하지만 사용하는 동형암호 라이브러리에 따라 암호화 데이터에 대한 연산 속도, 데이터 처리 방식, 동작하는 환경에 대한 한계가 존재하기 때문에 아직은 상용화단계는 무리일수도 있지만 표준화된 좋은 동형암호 라이브러리가 있을 경우 데이터를 암호화한 상태에서의 연산 처리가 가능하기 때문에 데이터에 대한 근본적인 보안을 유지할 수 있는 더 발전된 시스템을 만들 수 있을 것이다.

[참고문헌]

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, Mauro Conti. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. (2017). arXiv:1704.03578.

[2] Hao Chen, Kim Laine, Rachel Player. Simple Encrypted Arithmetic library - SEAL v2.1. (2016)

[3] Kevin C.King, Vinod Vaikuntanathan, Optimizing Fully Homomorphic Encryption. (2016)

[4] Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Improving TFHE: faster packed homomorphic operations and efficient circuit bootstrapping. Cryptology.(2017)

[5] Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in

less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). doi: 10.1007/978-3-662-46800-5_24

완전동형암호를 이용한 덧셈 연산: TFHE 소개와 구현

홍미연*, 윤지원*

*고려대학교 정보보호대학원 정보보호학과

Addition Method using Fully Homomorphic Encryption: Introduction and Impletementation of TFHE

Mi Yeon Hong* Ji Won Yoon*

*Graduate School of Information Security, Korea University.

요 약

기존의 암호화 시스템은 암호화된 메시지를 교환 하는데 있어 서로 공개키 또는 비공개키를 공유하는 방식을 사용한다. 그러나 이 방법은 특히 광범위한 클라우드 서비스에서 민감한 데이터의 개인정보를 통제할 수 없어 프라이버시에 대한 우려를 제기한다. 동형암호는 복호화 없이 암호화된 데이터를 처리할 수 있어 이러한 문제를 해결할 수 있다. 본 논문에서는 완전동형암호 라이브러리 중 하나인 TFHE(Fast Fully Homomorphic Encryption over the Torus)를 살펴보고 논리회로를 이용해 동형 덧셈 연산을 구현해 본다. 마지막으로 기계학습에서 TFHE를 활용한 연구 방향을 논의해 보고자 한다.

I. 서론

클라우드 컴퓨팅 기술이 발전하고 규모가 커짐에 따라 대량의 데이터 수집 및 분석이 네트워크상에서 이뤄지고 있다. 전통적인 암호화 방법을 구현하는 클라우드 컴퓨팅은 데이터를 처리할 때 이를 복호화 과정을 수행해야 하며, 이때 권한이 없는 사용자가 클라우드의 데이터에 액세스 할 수 있어 기밀성이 침해될 수 있다. 이러한 보안 문제를 해결하기 위한 방법 중 하나로 제시된 것이 바로 동형암호(Homomorphic Encryption)이다.

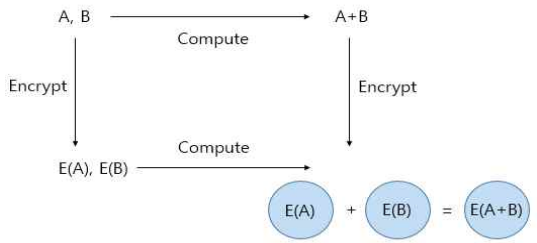


그림 1. 동형암호의 수학적 설명

암호화된 데이터를 복호화하지 않고 연산을 수행할 수 있는 동형암호를 이용하면 클라우드가 데이터를 복호화하지 않고 처리할 수 있어 기밀성을 유지할 수 있다.

1970년대 이론 연구가 시작된 이후, 안전성의 문제와 연산을 수행할 때마다 증폭되는 노이즈의 문제를 2009년 Gentry가 처음으로 해결하였고 기술적으로 증명하였다. 이를 완전동형암호(Fully Homomorphic Encryption)라고 하며 현재 다양한 완전동형암호 라이브러리들이 등장하고 있다.

본 논문에서는 완전동형암호 라이브러리 중 하나인 TFHE 라이브러리를 소개하고 덧셈 연산을 제시하여 이를 구현 해 본다. 또한 기계학습 분야에서의 응용 방안을 살펴본다.

II. Fast Fully Homomorphic Encryption over the Torus

2.1 TFHE 소개

Gentry가 제안한 완전동형암호 기술은 암호문에서 축적된 잡음(Noise)을 처리하기 위해 부트스트래핑(Bootstrapping)이라는 기술을 도입했다. 하지만 이 기법은 평문을 비트별로 암호화해야하기 때문에 암호문의 용량이 매우 크고, 부트스트래핑 연산 알고리즘이 복잡하여 암호화에 필요한 시간이 길어 현실적으로 적용하기 어렵다.

TFHE(I.Chillotti et al)는 GSW 및 ring variants를 기반으로 한 완전동형암호로, 알고리즘의 성능에 가장 큰 영향을 미치는 부트스트래핑 성능을 개선하였다. Gate - by-gate 부트스트래핑 알고리즘을 통해 1G의 부트스트래핑 키를 16MB로 줄였으며, 암호문의 오버헤드를 유지하면서 높은 보안 매개변수를 사용하여 시간을 690ms에서 13ms로 단축하였다. 사용하는 보안 파라미터는 lambda이고 암호문을 비트 단위로 암호화하여 연산하며, 암호화된 비트간의 NOT, AND, NAND, OR, XOR, NOR 등의 게이트 연산을 지원한다. 또한 다른 동형암호 라이브러리와 달리 게이트 수나 구성에 제한이 없어 최적의 회로는 가능한 최소 수의 게이트를 가지며 더 적은 범위에서는 병렬로 평가할 수 있다.

2.2 논리회로를 이용한 덧셈 연산

TFHE는 하나의 평문 비트를 암호화 하여 배열의 한 칸에 저장되며, 컴퓨터가 비트 단위로 데이터를 처리하기 때문에 일반화된 암호화 스킴(scheme)을 모든 데이터에 적용할 수 있다.

본 절에서는 TFHE를 바탕으로 다양한 게이트를 이용하여 사칙연산 중 하나인 덧셈 연산을 해 본다. 모든 입력 데이터는 이진수로 변환되어야 하므로, 0과 1로 인코딩한다. 배열 $A=[a_0,a_1,a_2,...,a_{i-1},a_i]$로 표현할 수 있고, 여기서 a_i는 최상위 비트이며 배열의 각 비트는 LWE암호화 방식을 이용하여 암호화 된다. 비트 단위 full adder을 만들기 위해 가장 간단한 완전동형암호 스킴은 NAND 게이트 후 부트스트래핑을 하는 것이다. 덧셈 연산에서는 9개의 NAND gate가 필요하지만, TFHE의 성격으로 인해 2개의 XOR 게이트와 2개의 AND 게이트, 1개의 OR 게이트를 사용하는 것이 빠르다. A와 B의 i번째 비트인 A_i와 B_i를 입력하면, 덧셈의 결과인 S_i와 다음 캐리비트 C_{i+1}를 반환하는 회로는 다음과 같다.

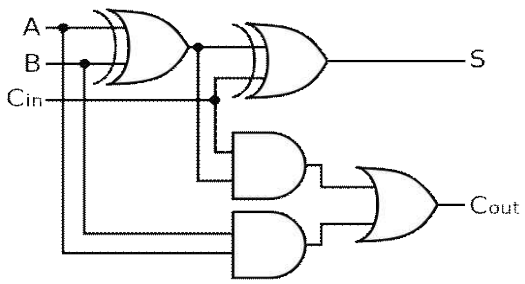


그림 2. 덧셈 연산의 회로도

위의 회로를 수행하는 의사코드는 아래의 [표 1]과 같다.

```
input:  A, B
output:  S
TFHEaddition(S, A, B, length){
    for(int i=0; i<length-1; i++){
        HomXOR(t[0], A[i], B[i]);
        HomAND(t[1], A[i], B[i]);
        HomXOR(S[i], t[0], C[i]);
        HomAND(t[0], t[0], C[i]);
        HomOR(C[i+1], t[0], t[1]);}
HomXOR(t[0], A[length-1], B[length-1]);
HomXOR(S[length-1], t[0], C[length-1]);
}
```

표 1. XOR, AND, OR을 이용한 두 암호문의 덧셈 연산 알고리즘 의사코드

Intel Core i7-490K 4.00GHz, 15.5GB RAM, Ubuntu 16.04 LTS 환경에서 정수 5와 7의 덧셈 연산을 구현해 본 결과 약 1.0337초의 시간이 소요되었다.

Ⅲ. 앞으로의 연구

인공지능의 한 분야인 기계학습은 크게 지도학습(Supervised Learning)과 비지도학습(Unsupervised Learning)으로 나뉜다. 레이블이 지정되지 않은 훈련 데이터를 이용해 레이블을

학습하여 분류하는 단계를 거친 후, 테스트 데이터를 입력하여 분류하거나 결과를 예측하는 과정을 통틀어 지도학습이라고 한다.

정보통신의 발달로 방대한 데이터가 축적되는 만큼 데이터를 이용하는 과정에서 민감한 개인정보가 유출될 수 있어 개인정보 보호가 중요한 문제로 떠오르고 있다. 이에 따라 개인정보를 보호하면서 데이터를 원활하게 이용하기 위한 방안으로 기계학습 알고리즘에 완전동형암호 스킴을 더하고자 한다. 현재 TFHE 라이브러리에 사칙연산을 비롯한 다양한 연산 알고리즘을 개발중에 있으며, 이를 바탕으로 부족한 연산을 채워 넣어 지도학습의 하나인 로지스틱 회귀를 비롯한 다양한 기계학습 알고리즘을 완성해 보고자 한다.

기계학습의 훈련 데이터와 테스트 데이터가 암호화 되어 입력되면 계산 역시 암호화된 상태에서 진행되기 때문에 어떠한 과정에서도 복호화가 일어나지 않아 데이터의 비식별화가 가능하다. 동형암호화된 기계학습의 기술이 발전하면 음성, 신용, 의료 데이터 등 데이터 사용 및 분석이 제한되어 있는 분야에서 활용될 가능성이 높을 것이라 예상한다.

[참고문헌]

[1] I. Chillotti, N.Gama, M.Georgieva, M.Izabachene, TFHE: Fast Fully Homomorphic Encryption over the Torus, 2017

[2] I. Chillotti, N.Gama, M.Georgieva, M.Izabachene, Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds, 2016

[3] S.B.Lee, B.K. Song, An Efficient Homomorphic Encryption Addition Method using Logic Circuit Arithmetic, 2017

[4] A.ACAR, H.AKSU, A.SELCUK ULUAGAC, A Survey on Homomorphic Encryption Schemes:Theory and Implementation, 2017

동형 암호를 이용한 효율적인 프라이버시 보존 헬스케어 시스템

박상우*, 송주석*

*연세대학교 컴퓨터과학과

Efficient Privacy-Preserving Healthcare System with Homomorphic Encryption

SangWoo Park*, JooSeok Song*

*Computer Science Department, Yonsei University.

요 약

헬스케어는 사물인터넷, 클라우드 컴퓨팅 등과 접목되면서 빠르게 성장하고 있지만 그와 함께 프라이버시 문제 또한 중요해지고 있다. 이와 관련하여 동형 암호를 이용해 서비스 제공자에게 사용자의 데이터를 노출하지 않는 방법이 제안되었지만 구체적인 과정이나 분석이 없었으며 완전 동형 암호를 이용하기 때문에 오버헤드가 크다는 단점이 있다. 본 논문에선 부분 동형 암호를 이용, 보다 효율적으로 헬스케어 정보를 서비스 제공자로부터 보호하는 방법을 제안한다. 또한 해당 방법을 실제 모델에 적용하여 구체적으로 제시하고 분석하였으며, 시뮬레이션을 통해 다른 시스템에 비해 효율적임을 보였다.

I. 서론

스마트 헬스케어는 사물인터넷(IoT, Internet of Things), 클라우드 컴퓨팅, 빅데이터 및 인공지능(AI)을 헬스케어에 접목한 분야로, 정보 기술의 발전과 함께 빠르게 성장하여 2020년 시장규모가 3,500억 달러에 달할 것으로 전망하는 등 각광받고 있다.[7,8] 주로 의료기관이나 일상생활에서 Wireless Body Area Network (WBAN)을 통해 정보를 전송받아 건강과 관련된 서비스를 제공한다. 하지만 이러한 생체정보는 유출될 시 다른 종류의 정보보다 더욱 치명적이기 때문에 헬스케어에 대한 보안, 프라이버시에 대한 사람들의 관심이 점점 더해지고 있다.[9]

이에 따라 헬스케어의 보안, 프라이버시에 대한 연구가 활발히 진행되고 있다. 이 중 프라이버시 보호 기술로는 익명화(anonymization), 암호화, 접근 제어 등이 있으며[1] 이러한 기술들은 주로 사용자 정보가 서비스 제공자가 아닌 제 3자로 유출되는 것을 막는 것에 초점이 맞춰져있다. 반면 생체 정보와 같이 민감한 데이터는 제 3자 뿐 아니라 서비스 제공자로부터도 보호해야 될 필요가 있다[4]. 예를 들어 체중관리나 운동 도움 서비스에서 사용자는 모르는 사이 민감한 정보가 서비스 제공자에게 노출될 수 있는데, 서비스 제공자는 이를 팔거나 사용자의 생명에 위협을 주는 행위에 악용할 수도 있다.

이러한 문제는 동형 암호(homomorphic encryption)을 통해 해결할 수 있다.[10] 동형 암호란 두 개의 원문(plaintext)과 각각의 암호문(ciphertext)에 대해, 원문들에 특정 연산을 수행한 결과와 암호문들에 대해 특정 연산을 수행한 후 복호화한 결과가 같은 성질을 만족하는 암호를 뜻한다. 이를 수식으로 표현하면 아래와 같다.

$$E(x_1) \triangle E(x_2) = E(x_1 \nabla x_2)$$

이러한 동형성(homomorphism)을 이용하면

서비스 제공자에게 민감한 정보를 숨기고도 사용자는 서비스를 정상적으로 받을 수 있다, 먼저 사용자는 동형 암호 비밀키를 이용해 민감한 정보를 암호화한 후 본래 데이터가 아닌 암호값을 서비스 제공자에게 전송한다. 서비스 제공자는 비밀키가 없기 때문에 데이터를 복호화할 수 없지만, 서비스를 위해 필요했던 연산을 암호문에 직접 수행할 수 있다. 사용자는 받은 정보를 복호화하여 본래 얻고자 했던 결과를 그대로 얻을 수 있게 된다.

하지만 이러한 동형 암호를 설계하는 것은 어려운 일이며, 특히 모든 종류의 비트 연산에 대해 동형성을 만족하는 완전 동형 암호(Fully homomorphic encryption)는 연산 오버헤드가 크기 때문에 실제로는 쓰기 힘든 수준이다. 반면 동형성을 일부 연산에 대해서만 만족하는 부분 동형 암호(patial homomorphic encryption)의 경우 일부 서비스들에 제한적으로만 보호를 할 수 있다.

이러한 동형 암호를 이용하여 헬스케어 프라이버시 보호에 기여한 연구들이 여럿 존재한다.[3,4,5,6,10] 하지만 이들은 프라이버시 보호 대상이 서비스 제공자가 아니거나, 서비스 제공자에 대한 프라이버시 보호 과정이 구체적으로 제시되지 않았거나, 현재 실제로 사용하기 힘든 완전 동형 암호를 이용해 제안하고 있다.

이에 본 논문에선 서비스 제공자로부터 사용자의 프라이버시를 보호할 수 있는 구체적이고 효율적인 헬스케어 시스템을 제안한다. 부분 동형 암호 중 하나인 Paillier 암호[11]를 이용하기 때문에 완전 동형 암호보다 오버헤드가 적고 실현 가능성이 더 높다. 또한 시나리오를 구체적으로 제시하고 이를 분석하여 프라이버시 보호를 증명하였고 실험을 통해 비교적 적은 오버헤드를 가짐을 보여주었다.

II. 관련 연구

동형 암호를 이용한 헬스케어 프라이버시 보호 연구들은 크게 서비스 제공자로부터의 프라이버시 보호를 제안한 연구와 그렇지 않은 연구가 있다. 전자에 해당하는 연구들을 보면, 첫 번째로 WBAN에서 프라이버시 보호를 위해 role기반 접근 제어와 함께 데이터 싱크(sink)로부터 데이터를 지키기 위해 완전 동형 암호를 이용한 방법을 제안한 연구가 있다.[3] 이를 통해 제 3자 및 싱크로의 부적절한 데이터 유출을 줄였다. 다음으로 헬스케어 멀티클라우드 환경에서 완전 동형 암호를 이용해 클라우드 서버에서의 데이터 유출을 막는 방법을 제안한 연구가 있다.[4] 마찬가지로 헬스케어에서 클라우드에 대한 프라이버시 보호를 위해 속성 기반 암호(attribute based encryption) 및 동형 암호를 이용한 방법을 제안한 연구가 있다[5]. 해당 연구들은 구체적인 과정에 대한 제안과 분석이 없고 완전 동형 암호를 이용하기 때문에 현재는 현실성이 부족하다.

반면 협업 필터링(collaborative filltering) 기반 헬스케어 추천 시스템에서 다자간 데이터 공유 시 paillier 동형 암호를 이용하여 프라이버시를 보호하는 방법을 제안한 연구와[6] Yao의 프로토콜 및 동형 암호를 이용하여 의학 연구자 및 기관이 프라이버시를 보호하며 데이터를 공유할 수 있는 방법을 제안한 연구가 있다.[10] 하지만 해당 연구들은 서비스 제공자로부터 프라이버시 보호를 하는 것이 아니기 때문에 본 연구에서 제안한 부분과 차이가 있다. 즉 서비스 제공자에 대한 프라이버시 보호를 구체적이고 현실성 있게 제안한 연구는 아직 없다.

III. 제안하는 방법

3.1 환경

사용자는 서비스 제공자에게 데이터를 암호화해서 보내고, 서비스 제공자는 암호화된 데이터를 복호화하지 않고 요청하는 서비스를 제공하는 것을 목표로 한다. Paillier 암호를 사용하기 위해 키가 필요한데, 통신이 시작되기 전 사용자와 서비스제공자는 외부 키 서버로부터 같은 공개키(PK)를 받았고 사용자는 추가로 비밀키(SK)를 받았다고 가정한다.

헬스케어 및 WBAN의 서비스엔 다양한 종류가 있다. 먼저 가장 기본적인 과정을 제안하고, 이를 특정 모델에 대해 구체화 해 보도록

한다.

3.2 기본 과정

사용자와 서비스제공자가 통신하는 과정에서 서비스제공자는 사용자가 전송한 정보에 맞는 서비스를 데이터베이스(DB)를 통해 결정하게 된다. 자세한 과정은 다음과 같다.

1) 사용자는 생체 데이터집합 $P=\{P_1, P_2, \cdots P_k\}$에 대해 Paillier 암호화를 수행하여 암호문집합 C 를 생성한다.

$C=\{C_1, C_2, \cdots C_k\}$, $C_j = E_{PK}(I_j)$

2) 서비스제공자는 output을 제외한 데이터베이스의 값을 PK로 암호화하여 다음과 같은 데이터베이스(DB')을 만든다.

ATT_1	ATT_2	$\cdots$	ATT_k	output
(암호화된 데이터)				(값)

3-1) C 를 기반으로 데이터베이스에 다음과 같은 질의를 보내 output을 결정한다.

```
SELECT    output
FROM      DB'
WHERE     DB'.ATT = C
```

이 때 $DB'.ATT=\{ATT_1, ATT_2, \cdots, ATT_k\}$이다. 즉 C와 값이 일치하는 행의 output 값을 찾아 사용자에게 알맞은 서비스를 제공할 수 있다.

하지만 보통 C값은 생체정보 측정값의 암호문이므로 $DB'.ATT$와 C가 정확히 일치하지 않는 경우가 많다. 이 땐 다음과 같은 질의로 output을 결정한다.

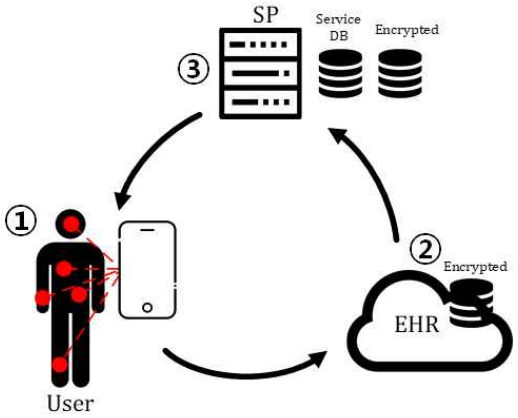
3-2)

```
SELECT    output
FROM      DB'
WHERE     -
ORDER BY  Σ_{i=1}^{k} W_i · (DB'.ATT_i − C_i)^2
LIMIT     1
```

이 때 W_i는 ATT_i의 가중치이며 서비스에 따라 결정된다. 위의 질의는 DB'의 튜플로 이루어진 벡터와 C 원소로 이루어진 벡터 사이의 거리를 구하는데, 이를 정렬시킨 다음 제일 작은 튜플의 output 값을 반환한다. C와 튜플의 거리가 좁다는 것은 즉 C와 가장 유사한 튜플이라는 뜻이다. 따라서 3-1)이 튜플과 C값이 같은 것을 찾는다면 3-2)에선 C값과 제일 가까운 튜플을 찾아 output을 반환한다. 여기서 중요한 점은 $DB'.ATT$와 C가 암호화 된 상태에서 이루어진다는 것이다. Paillier 암호는 덧셈과 상수곱셈에 대해 동형성(homomorphism)을 만족하기 때문에[11] 3-2)의 수식 계산($\Sigma, -$) 및 ORDER BY 연산이 유효하다(Ⅳ분석 섹션 참조). 이후 output을 사용자에게 전송함으로써 서비스가 종료된다.

3.3 전자건강기록 기반 프라이버시 헬스케어 서비스 모델

3.2에서 제시한 방법을 헬스케어 사례에 적용하여 살펴보자. 전자건강기록(Electronic Health Record, EHR) 이란 어떤 환자의 의료기록 및 헬스케어 정보를 모아 저장해놓은 것을 뜻한다. [그림1]은 일반적인 전자건강기록 기반 헬스케어 모니터링 시스템의 구조를 보여주고 있다. 이 때 사용자, EHR 클라우드, 서비스 제공자는 같은 공개키를 공유하며 비밀키는 사용자만 가진다.



[그림 1]

1) 웨어러블 디바이스를 통해 수집된 사용자 정보는 지속적으로 클라우드 내의 EHR로 전송되어 저장된다.

2) EHR은 클라우드 내부에 있는데, 이 때 클라우드 서비스 제공자를 믿을 수 없기 때문에 3.2 기본 과정대로 데이터를 암호화하여 전송한다. 따라서 EHR의 데이터베이스에는 자동으로 암호화된 정보가 저장된다.

3) 서비스 제공자는 접근이 허용된 범위에서 EHR에 접속하여 모니터링하고 사용자에게 능

동적으로 서비스를 제공한다. 서비스의 종류는 진료, 처방 등 생명과 연관된 것 뿐 아니라 운동 추천, 식단 관리 등 다양하며 후자일수록 사용자는 민감한 데이터가 보호되길 원할 것이다. 서비스 DB를 이용해 사용자에게 output을 전송하는 것은 3.2 기본 과정을 따르면 된다.

해당 모델에서 제안한 방법의 가지는 추가적인 장점이 두 가지 존재한다. 하나는 클라우드 서비스 제공자와 서비스 제공자로부터 프라이버시를 보호할 수 있게 된 것이다. 두 번째는 EHR과 서비스 제공자의 암호화가 통합되어 효율적으로 동작하게 된 점이다. 만약 EHR, 서비스 제공자가 따로 암호화 한다면 필요한 암호화 과정과 키 관리가 두배가 될 것이다.

IV. 분석

이번 섹션에선 Ⅲ에서 동형 암호를 통한 프라이버시 보존에 대해 분석할 것이다.

먼저 사용자의 PK로 암호화된 정보 C는 서비스 제공자가 볼 수 없으며, 서비스 제공자는 정보를 확인하지 않고 서비스를 제공할 수 있다. Paillier 암호시스템[11]은 다음과 같다.

공개키 : $n\ (=p \cdot q),\ g=n+1$

비밀키 : $p, q, \lambda=(p-1)(q-1), \mu=\lambda^{-1}$

이 때 원문 메시지 m, 랜덤 숫자 r에 대해 $c=g^m \cdot r^n \bmod n^2$이다. Discrete logarithm problem에 의해 c, g를 알고 있더라도 원문 m을 알아내기 힘들고 factorization problem에 의해 n을 알더라도 비밀키 p, q, μ을 계산하기 힘들다.

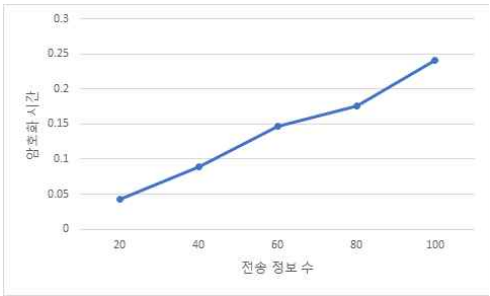
반면 3-2)에서 C와 튜플의 가중치 기반 유사도를 다음과 같이 계산한다.

$$\sum_{i=1}^{k} W_i \cdot (DB'.ATT_i - C_i)^2$$

이 때 DB' 튜플의 값과 C_i의 값은 모두 암호화된 상태이다. 하지만 pailliar 암호의 덧셈 동형성에 의해 DB'의 원본값(plaintext)과 P_i값을 먼저 계산한 후 암호화 한 것과 동일하다. 즉 데이터베이스 값과 생체데이터 값의 차이에 대해 유사도 계산이 이루어지는 것과 같다. 따라서 복호화 없이 유사도를 계산하는 것이 가능하다.

V. 실험

제안한 방법의 오버헤드를 테스트하기 위해 시뮬레이션을 진행하였다. 실험 환경은 intelCore(TM) i3-4150, 8GB메모리, 64-bit ubuntu vmware이다. 사용자의 생체 데이터를 Paillier 방식으로 암호화하는데 걸리는 시간을 분석하였다. 데이터는 공공데이터포탈[14]에서 다운로드한 체지방지수, 시력, 혈압의 정보를 이용했고, 20개에서 100개로 개수를 변화시켜서 관찰했다.



시뮬레이션 결과 데이터셋 100개를 암호화하여 전송하는데 걸리는 시간이 약 0.27초 정도로 소요되었다. WBAN 환경에서 핸드폰으로 암호화가 이루어짐을 감안해도 제안하는 방법이 실현가능함을 보여준다. 특히 완전 동형 암호를 이용한 방법들에서 걸리는 시간에 비해 훨씬 현실적임을 확인할 수 있다.

.

VI. 결론

본 논문에서는 헬스케어에서 동형 암호를 이용해 서비스 제공자로부터 민감한 정보를 보호할 수 있는 방법을 제안하고 분석을 통해 증명하였다. 또한 부분 동형 암호를 이용했기 때문에 실생활에 적용할 수 있다는 것이 큰 장점이다. 반면 완전 동형 암호에 비해 서비스 방식에 따라 제한적으로만 적용이 가능하다는 단점이 있다. 또한 데이터베이스 암호화 과정을 단순화시킬 수도 있어 지금보다 개선될 여지가 또한 있다.

[참고문헌]

[1] Sahi, Muneeb Ahmed, et al. "Privacy Preservation in e-Healthcare Environments: State of the Art and Future Directions." Ieee Access 6 (2018): 464-478.

[2] Giannetsos, Thanassis, Tassos Dimitriou, and Neeli R. Prasad. "People centric sensing in assistive healthcare: Privacy challenges and directions." Security and Communication Networks 4.11 (2011): 1295-1307.

[3] Gowtham, M., and S. Sobitha Ahila. "Privacy enhanced data communication protocol for wireless body area network." Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on. IEEE, 2017.

[4] Mekala Bhaskar, Dr M. Ashok, and M. D. A. Hasan. "Homomorphic Encryption Algorithm used in Multi-Format Data in Collaborated Healthcare Multi cloud Computing." (2017).

[5] Elmogazy, Huda, and Omaima Bamasak. "Towards healthcare data security in cloud computing." Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for. IEEE, 2013.

[6] Kaur, Harmanjeet, Neeraj Kumar, and Shalini Batra. "An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system." Future Generation Computer Systems (2018).

[7] 박종태, 천승만, and 고석주. "사물인터넷 기반 헬스케어 서비스 및 플랫폼 동향." 한국통신학회지 (정보와통신) 31.12 (2014): 25-30.

[8] 정기철, et al. "수요자 중심의 헬스케어 산업 전망과 대응전략." 정책연구 (2017): 1-160.

[9] Al-Janabi, Samaher, et al. "Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications." Egyptian Informatics Journal 18.2 (2017): 113-122.

[10] Chen, Feng, et al. "Precise: Privacy-preserving cloud-assisted quality improvement service in healthcare." 2014 8th International Conference on Systems Biology (ISB). IEEE, 2014.

[11] Paillier, Pascal. "Public-key cryptosystems based on composite degree residuosity classes." International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1999.

[12] Zhang, Rui, and Ling Liu. "Security models and requirements for healthcare application clouds." Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. IEEE, 2010.

[13] 허미영, and 이윤호. "완전동형암호기반 프라이버시 보호 Top-k 위치정보서비스." 한국시뮬레이션학회논문지 24.4 (2015): 153-161.

[14] https://www.data.go.kr

말랑말랑 톡카페 애플리케이션 데이터베이스 복호화 방안 연구※

김기윤*, 신수민*, 이종혁*, 김종성*,**

*국민대학교 정보보안암호수학과 **국민대학교 금융정보보안학과

Study on Database Decryption Method of MalangMalang Talkcafe

Giyoon Kim*, Sumin Shin*, Jonghyeok Lee*, Jongsung Kim*,**

*Dept. of Information Security, Cryptology and Mathematics, Kookmin University

**Dept. Financial Information Security, Kookmin University

요 약

개인정보 유출 사건이 빈번이 발생함에 따라 개인정보보호에 대한 관심이 높아지고 있다. 이에 따라 애플리케이션 개발자들은 사용자의 데이터를 은닉할 수 있는 애플리케이션을 개발하거나 데이터를 암호화하여 저장하는 방식을 택하고 있다. 인스턴트 메신저[1] 중의 하나인 말랑말랑 톡카페도 데이터를 암호화하여 저장한다. 인스턴트 메신저는 대화 내용, 마지막 사용 시간, 친구 목록, 휴대폰 번호, 이메일 등 중요한 내용이 저장되어있을 수 있으므로 반드시 조사 되어야 할 대상이다. 그러나 이와 같은 애플리케이션이 실제 수사에 사용되기 위해서는 암호화된 데이터를 복호화하는 것이 필요하다. 본 논문에서는 말랑말랑 톡카페 애플리케이션의 데이터베이스 암호화 과정을 파악하고, 데이터 복호화 방안을 제시한다.

I. 서론

디지털 기술이 발전함에 따라 사람들은 다양한 디지털 기기와 연관되어 있다. 대다수의 디지털 기기는 많은 데이터를 저장하고 있으며 저장된 데이터는 범행증거로 사용된다. 따라서 디지털 포렌식은 필수불가결한 수사 기법중 하나로 자리 잡게 되었다.

최근 잠금 또는 암호화 기능을 제공하는 애플리케이션이 증가하고 있다. 대표적으로, 인스턴트 메신저인 카카오톡, 위챗 등은 대화 내용을 데이터베이스에 저장한 후 데이터베이스 자체를 암호화 또는 대화내역과 같은 주요정보만 암호화하여 데이터베이스에 저장하는 방식을 취하고 있다. 이러한 대화 내용이 들어있는 데이터베이스는 수사에서 주요한 증거로 활용될 수 있으므로 다양한 기관에서 이를 복구하기 위한 연구를 진행한다[1,2]. 이러한 암호화 기능은 개인의 프라이버시를 위해 만들어졌으나, 자신의 범행을 감추려 하는 등 범죄자에 의해 악용이 될 수 있으므로 암호화된 파일에 대한 복호화 방안 연구는 필수이다. 본 논문에서는 한글과컴퓨터에서 제작한 말랑말랑 톡카페 인스턴트 메신저의 데이터베이스에 대한 복호화 방안 및 데이터 분석 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 암호화된 데이터베이스에 대한 복호화 프로세스를 설명하고 3장에서는 데이터베이스에 저장되는 데이터 종류 및 의미에 대해 언급하며 마지막 4장에서는 본 논문에 대한 결론으로 마무리한다.

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 작성되었습니다.(No.2017-0-00344, 최신 모바일 기기에 대한 암호해독 및 포렌식 분석)

1) Instant Messenger : 인터넷 같은 네트워클 리용한, 두 명 이상의 즉각적인 텍스트 통신에 이용되는 클라이언트

II. 데이터베이스 복호화 프로세스

말랑말랑 톡카페 애플리케이션의 데이터베이스는 "/data/data/com.hancom.talkafe/databases/"에 저장되며 사용되는 주요 데이터베이스는 user.db와 talk.db이다. 해당 데이터베이스는 파일 전체가 암호화되어있다.

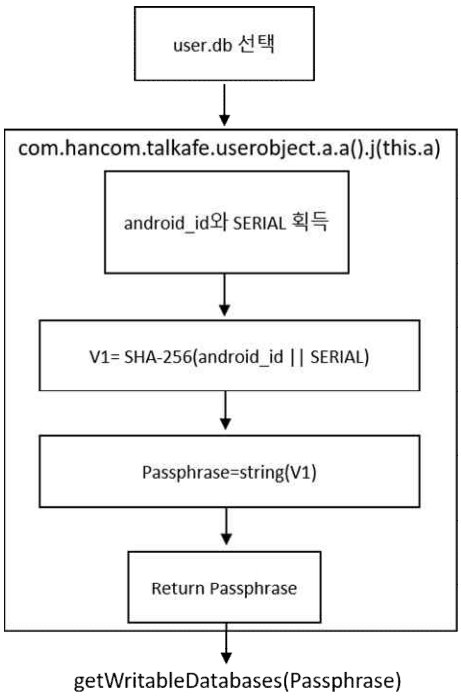
암호화되어있는 데이터베이스의 복호화 과정을 분석하기 위해 역공학 기법을 사용하였으며, JEB Decompiler를 이용하였다[3].

말랑말랑 톡카페 애플리케이션에는 다양한 외부 라이브러리(.so)가 사용된다. 외부 라이브러리는 그 특성상 프로그램 시작 시 native library[2]를 로드(load)해야 애플리케이션에서 해당 라이브러리의 기능을 사용할 수 있다. (그림 2-1)은 데이터베이스 암호화 기능을 제공하는 SQLCipher[3]의 외부 라이브러리를 로드하는 과정이다.

```
public static void loadLibs(Context arg2) {
    Class v1 = SQLiteDatabase.class;
    SQLiteDatabase.loadLibs(arg2, arg2.getFilesDir());
}
public static void loadLibs(Context arg2, File arg3) {
    Class v1 = SQLiteDatabase.class;
    System.loadLibrary("sqlcipher");
}
```

(그림 2-1) SQLCipher 라이브러리 로드 과정

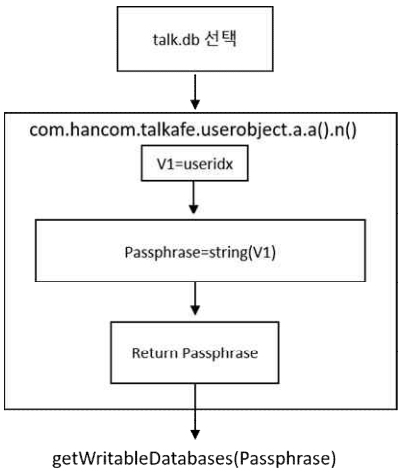
SQLCipher에 의해 데이터베이스가 암호화되어 있다면 데이터베이스에 새로운 값을 넣거나 읽을 때 getWritableDatabases() 혹은 getReadableDatabases() 함수를 사용하여 복호화된 데이터베이스에 접근할 수 있다[4]. 해당 함수는 패스프레이즈(Passphrase)를 입력 매개변수로 사용하는데, 패스프레이즈 생성과정은 (그림 2-2)와 같다[5].



(그림 2-2) user.db의 패스프레이즈 생성 과정

먼저, user.db를 복호화하기 위해 사용되는 SQLCipher의 패스프레이즈는 기기별 고윳값인 android_id와 시리얼 번호를 합치어 해싱한 값이다.

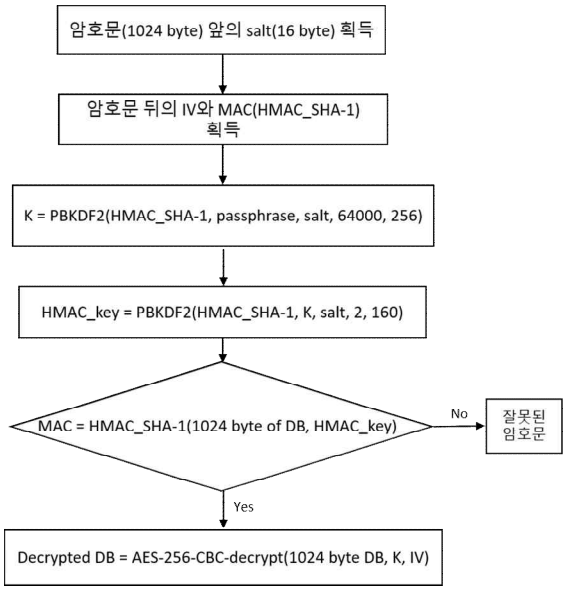
다음은 talk.db에 대한 패스프레이즈 생성과정이다. talk.db는 user.db와 다르게 useridx의 값을 패스프레이즈로 사용하는 것을 확인할 수 있다. 해당 값은 이용자의 id에 관한 고윳값으로 user.db 내부에 존재한다.



(그림 2-3) talk.db의 패스프레이즈 생성 과정

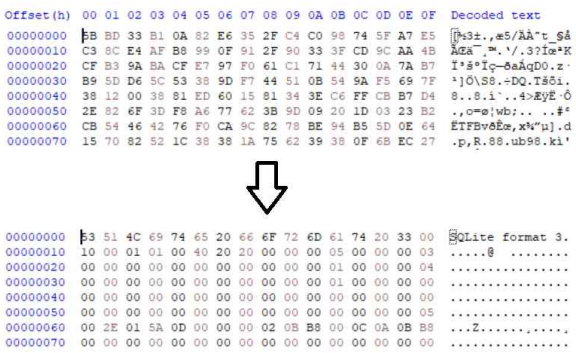
2) 자바가상머신이 함수의 원시정보를 사용할 수 있게 해주는 라이브러리
3) SQLite를 위한 암/복호화 모듈

생성된 패스프레이즈는 getWritableDatabases 혹은 getReadableDatabases 함수에 내에서 SQLCipher의 복호화를 하는 데 사용된다. 복호화를 하는 일련의 과정은 다음과 같다.



(그림 2-4) SQLCipher를 이용한 복호화 과정

실제로 실험에 사용된 기기의 고윳값들을 사용하여 패스프레이즈를 생성 후 SQLCipher를 이용해 복호화해 본 결과 정상적으로 데이터베이스가 복호화됨을 확인하였다(그림 2-5).



(그림 2-5) user.db의 복호화 결과

III. 데이터베이스 분석

본 장에서는 2장에서 복호화한 user.db와 talk.db에 대한 분석 결과를 제시한다.

user.db는 이용자의 개인정보를 보관하는 데이터베이스로 이용자뿐만 아니라 친구 추가가 되어있는 이용자 혹은 친구신청을 요청받은 이용자의 개인정보까지 담고 있다. talk.db는 이용자 간의 대화들을 저장하고, 그에 관련된 시간 정보나 대화 타입, 유저정보를 일부 담고 있다.

각각의 데이터베이스에 대한 주요 데이터들은 다음과 같다.

[표 3-1] user.db의 주요 데이터

user table		
컬럼 데이터	설명	비고
useridx	이용자 고유 인덱스	
userid	로그인 ID	
username	이용자 지정 이름	
profileMessage	프로필 메시지	
idType	아이디 유형	친구:1 공식친구:4
phoneNo	사용자 등록 휴대폰 번호	
updateTime	마지막 접속시간	유닉스타임
friends	친구 상태인 이용자 idx	
hideFriends	숨김 친구인 이용자 idx	
blockFriends	차단한 이용자 idx	
favoriteFriends	즐겨찾기한 이용자 idx	
newSearchID	친구 검색시 사용되는 ID	
inviteName	초대시 표시할 이름	
secretProfileFlag	특정 이용자만 볼 수 있는 정보 사용	True/False

[표 3-2] talk.db의 주요 데이터

chats table	
컬럼 데이터	설명
roomidx	채팅방 고유 index
type	메시지 유형
text	메시지의 내용
sender	보낸 사람
time	받은 시간
rooms table	
컬럼 데이터	설명
name	채팅방 설정 이름
roomsLastIdx	마지막 사용 채팅방
lastChatText	마지막 메시지
lastChatTime	마지막 채팅 시간
lastChatSender	마지막 송신 이용자
mqttPort	메시지 프로토콜 포트번호
pung	메시지 자동 삭제
autopung	메시지 전송 예약

talk.db는 user.db가 복호화되었다면 손쉽게 획득할 수 있다. 하지만 user.db의 경우 패스프레이즈 생성에 기기 고윳값을 사용하므로 직접 사용이 가능한 스마트폰을 갖고 있지 않은 이상 패스프레이즈 생성은 어렵다. 하지만 talk.db의 패스프레이즈는 이용자의 고유 인덱스값이고, 전수조사 가능한 크기이므로 손쉽게 복호화가 가능하다.

두 데이터베이스는 테이블 내부의 정보를 삭제할 때 DELETE()옵션과 DROP()옵션을 사용한다. 메시지만 삭제하는 경우는 DELETE()옵션이며, 대화방을 나갈 때에는 DROP()을 사용한다. 삭제된 데이터는 0으로 덮어씌워지기 때문에 복구자체는 불가능하나 메시지의 ID가 삭제된 만큼 행의 개수와 차이나기 때문에 데이터 삭제 여부를 추측할 수는 있다. 단, 대화방 나가기가 아닌 일부 대화내용 삭제에 대해서는 서버에서 데이터를 저장하고 있는 한 새로운 기기에서 로그인하는 것으로 대화내역 복구가 가능하다.

IV.결론

인스턴트 메신저에서의 대화 내역은 디지털 포렌식 수사에 있어 중요한 부분이며 핵심적인 증거 역할을 한다. 그러나 개인 정보 보호에 대한 인식이 높아지면서 데이터 저장 시 파일 암호화 기술이 적용되는 경우가 늘고 있다. 디지털 포렌식 관점에서는 해당 기술들은 안티 포렌식에 해당된다. 따라서 암호화된 데이터에 대한 원본 데이터 획득은 주요 연구 과제 중 하나이다.

본 논문에서는 말랑말랑 톡카페의 애플리케이션 분석을 통하여 암호화된 데이터베이스 복호화 방안을 제시하였다. 패스프레이즈의 생성 방안을 알아냈으며, 해당 패스프레이즈를 기반으로 실제 복호화에 성공하였다. 복호화된 데이터베이스에는 이용자의 개인정보 외에도 친구 추가가 되어있는 이용자의 개인정보, 대화 내역 등이 기록되어 있으며, 범죄행위에 대한 증거로써 활용할 수 있다.

[참고문헌]

[1] Songyang Wu, Yong Zhang, Xupeng Wang, Xiong Xiong, and Lin Du, Forensic analysis of WeChat on Android smartphones, Digital Investigation, Vol. 21, pp. 3-10, 2017.

[2] Zhongmin DAI, SUFATRIO, Tong-Wei CHUA, Dinesh Kumar BALAKRISHNAN, and Vrizlynn L. L. THING, Chat-App Decryption Key Extraction Through Information Flow Analysis, Cryptology and Information Security Series, Vol. 15, pp. 3-18, 2017.

[3] PNF Software, JEB, https://www.pnfsoftware.com/jeb/

[4] Android Developers, https://developer.android.com/reference/android/arch/persistence/db/SupportSQLiteOpenHelper

[5] Zetetic, https://www.zetetic.net/sqlcipher/sqlcipher-api/#key

모바일 환경에서의 광고 어플리케이션의 보안 취약점 분석

지우중, 김형식

성균관대학교 전자전기컴퓨터공학과

woojoong@skku.edu, hyoung@skku.edu

Security vulnerability analysis of advertising applications in mobile environment

Woojoong Ji, Hyoungshick Kim

Department of Computer Science and Engineering, Sungkyunkwan University.

요 약

광고 어플리케이션이란 모바일 광고의 일종으로, 스마트 폰으로 광고 업체가 요구하는 일정한 행동을 달성하고 현금이나 현금으로 교환할 수 있는 포인트로 이를 보상해주는 어플리케이션을 말한다. 광고 어플리케이션의 포인트는 금전적으로 교환될 수 있기 때문에 포인트를 불법적으로 획득하지 못하도록 보호하는 보안 기술이 구현되어야 한다. 본 논문에서는 광고 어플리케이션을 통해 사용자들에게 광고를 제공하는 국내 9개의 어플리케이션을 대상으로 어플리케이션에 대한 역공학 분석 및 네트워크 패킷 분석을 통해 광고 어플리케이션들의 보안성을 분석 하였다. 분석 결과, 재전송 공격을 통하여 사용자가 실제로 광고를 시청하는 행동 없이 불법적으로 광고 어플리케이션의 포인트를 획득할 수 있다는 사실을 확인할 수 있었다.

I. 서론

최근 스마트폰의 확산과 함께 모바일 광고시장도 크게 성장하고 있다. 특히 모바일 광고는 기존의 불특정한 다수를 대상으로 한 일반적인 광고와는 달리 특정 대상을 타겟팅을 할 수 있다는 큰 특징을 가지고 있다[1]. 이는 광고를 보는 사용자의 연령이나, 지역, 성별, 관심사를 분석하여 이에 맞게 사용자별로 노출되는 광고를 다르게 하여 광고 보여줌으로써 효율적으로 광고 홍보를 할 수 있는 장점이 있다. 이와 같이 스마트폰 환경을 기반으로 동작하는 광고 어플리케이션은 소비자의 시간과 장소에 구애받지 않고 광고에 대한 홍보를 할 수 있어 광고 시장에서 큰 인기를 얻고 있는 추세이다. 본 논문에서 소개하고자 하는 광고 어플리케이션이란 스마트폰을 이용하는 사용자가 원하는 광고 어플리케이션을 설치하여 어플리케이션에서 나오는 광고를 시청하거나, 이벤트 참여, 게임 및 광고주가 원하는 미션을 수행 할 경우 그에 따른 보상으로 현금이나 포인트를 적립해주는 어플리케이션이다. 광고주가 원하는 광고를 사용자가 시청하고 그에 따른 보상으로 받은 현금이나 포인트를 사용하여 다른 상품을 구입할 수 있다. 그리고 해당 광고 어플리케이션에서 지정한 특정 금액 이상 적립되면 자신의 은행계좌로 현금을 환급을 받을 수가 있다. 또한 상품권 교환, 기부 등으로 다양하게 활용할 수 있는 특징을 가지고 있다.

본 논문에서는 국내에 존재하는 수많은 광고 어플리케이션 중에서 광고 동영상을 시청하여 그에 따른 보상을 받는 9개의 광고 어플리케이션을 분석하였다. 광고 어플리케이션을 사용하는 사용자는 광고를 시청하는 대가로 금전적인 보상을 받을 수 있기 때문에 광고 어플리케이션은 안전하게 구현 및 관리 되어야 한다. 광고 어플리케이션의 다양한 보안 위협 가능성을 분석하기 위하여 네트워크 트래픽을 분석한 결과, 사용자가 광고 동영상을 시청한 후 일정 시간 이상 지나야 다시 광고를 시청 할 수 있게 제한이 되어 있지만 이를 무시하고 광고를 시청한 대가로 금전적인 보상을 받을 수 있음을 확인하였다. 그리고 사용자의 어떠한 행동 없이 자동적으로 금전적인 보상을 받을 수 있음을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 광고 어플리케이션에서 사용하는 광고 어플리케이션의 비즈니스 모델에 대해 설명하고 3장에서는 광고 어플리케이션의 비즈니스 모델 중 CPV 모델에서 동영상을 시청하여 그에 따른 대가로 보상을 받는 국내 광고 어플리케이션 9개에 대한 취약점에 대해 설명한다. 4장에서는 취약점 개선방안을 제시하고 마지막으로 5장에서는 결론을 제시한다.

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음. (IITP-2018-2015-0-00403)

II. 광고 어플리케이션의 비즈니스 모델

2.1 CPM

CPM(Cost Per Mille)은 광고의 노출에 따라 광고 단가를 책정하는 방식이다. 쉽게 말해 CPM 모델은 어떤 광고가 사용자에게 1,000회 노출되었을 때를 기준으로 광고에 대한 단가를 계산한다. 예를 들어 광고주와 광고 어플리케이션 업체와 직접 광고를 거래한다고 하면 어플리케이션 업체가 만든 광고 어플리케이션에 광고주의 광고가 사용자에게 1번 노출될 때마다 광고 업체는 1원의 광고 수익을 얻게 된다. 이를 기준으로 광고주는 마케팅의 효율성과 수익성을 평가하고 분석할 수 있다.

2.2 CPC

CPC(Cost Per Click)는 유저를 광고주의 사이트로 유입시키기 위해 사용되는 방식으로 광고주는 유저가 광고를 클릭한 횟수 당 광고비를 지급한다. CPC는 CPM과 달리 클릭 횟수를 기반으로 하는 만큼 광고주는 클릭 횟수를 통해 마케팅의 효율성과 수익성을 평가할 수 있다.

2.3 CPV

CPV(Cost Per View)는 모바일 동영상 광고에서 진행되는 과금 방식으로 유저가 광고 동영상을 시청하거나 광고 내에 랜딩 페이지로 넘어가는 버튼을 클릭하였을 경우 광고비를 지급하는 방식이다. 예를 들어 광고 어플리케이션을 사용하는 사용자가 해당 광고 동영상을 30초 동안 시청하게 되면 그에 따른 일정한 금전적인 보상을 받게 된다. 하지만 사용자는 동영상이 끝까지 재생이 되어야지만 보상을 받을 수 있기 때문에 해당 광고를 끝까지 시청하여야 한다.

2.4 CPA

CPA(Cost Per Action)는 광고에 노출된 유저가 회원가입, 이벤트 참여 등 광고주가 원하는 '특정 행동'을 하였을 경우 광고비를 지급하는 방식이다. 예를 들어 사용자의 스마트폰에 해당 광고 어플리케이션이 설치되면 사용자의 스마트폰 잠금화면에 광고주의 광고를 사용자에게 보여줌으로써 사용자가 잠금화면을 해제하는 특정 행동을 통해 해당 광고를 시청한 대가로 광고에 대한 보상을 받게 된다.

2.5 CPI

CPI(Cost Per Install)는 CPA에서 광고주가 원하는 특정 행동을 '어플리케이션 설치'로 지정한 방식이다. 광고를 통해서 유저가 광고주의 어플리케이션을 설치했을 때 광고주는 광고비를 지급하게 된다. 예를 들어 CPI 모델인 게임 어플리케이션을 설치한 보상 가격이 100원인 경우 만약 100명의 사용자가 해당 게임 어플리케이션을 광고 어플리케이션을 통해 설치하였다면 A게임사는 10,000원을 광고비를 광고 어플리케이션 업체에게 지급하게 된다. 10,000원을 받은 광고 어플리케이션 업체는 업체의 내부 사정에 맞게 광고비를 계산하여 사용자들에게 현금이나 포인트로 보상해준다.

III. 보안 취약점 분석 결과

본 논문은 Google Play Store에서 사용자가 동영상 광고를 시청한 대가로 보상을 제공하는 국내 9개의 CPV 모델 광고 어플리케이션을 대상으로 한다. 어플리케이션을 설치하고 패킷을 캡처하기 위해 루팅된 안드로이드 Nexus5 기기를 이용하였으며, Windows 10 환경에서 네트워크 트래픽 분석 툴인 Wireshark를 통해 패킷을 분석하였다.

3.1 재전송 공격

재전송 공격이란 프로토콜 상에서 유효 메시지를 골라 복사한 후 나중에 재전송함으로써 정당한 사용자로 위장하는 공격기법이다. 본 논문에서의 재전송 공격을 수행하기 위해서 악의적인 사용자는 광고 어플리케이션에서의 CPV 모델인 광고 동영상[그림 1]을 시청한다.

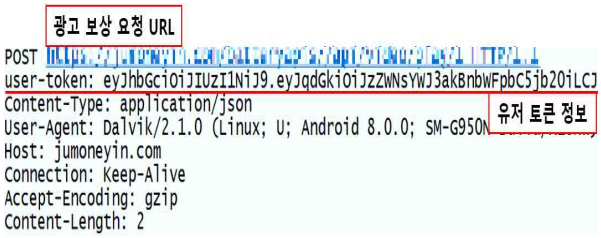
[그림 1] A사의 CPV 모델 광고 동영상

이렇게 광고 동영상[그림 1]을 15초 동안 시청한 사용자는 광고를 시청한 대가로 금전적인 보상을 받는다. 하지만 A사의 광고 어플리케이션[그림 2]에서는 사용자가 계속해서 광고 동영상을 통해 보상을 받지 못하게 일정한 시간이 지나야만 다시 시청할 수 있게 해당 광고에 대해 시간을 제한하였다.

[그림 2] A사의 동영상 광고 시간제한

이때 광고 동영상을 끝까지 시청한 악의적인 사용자는 해당 어플리케이션에서 서버로 전송하는 보상을 요청하는 패킷[그림 3]을 캡처하여 저장한다.

[그림 3] A사의 서버로 보상을 요청하는 패킷



[그림 4] E사의 서버로 보상을 요청하는 패킷

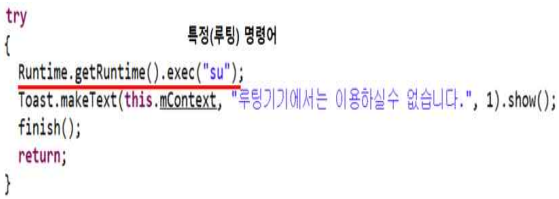
[그림 3]에는 A사 광고 어플리케이션을 사용하는 사용자를 식별하는 IDX 값과 해당 패킷이 서버로 전송되는 목적을 담은 func 값, 마지막 authkey라는 토큰 값을 JSON 파일로 저장하여 서버로 보상을 요청한다. 또 E사[그림 4]의 어플리케이션 같은 경우 유저의 토큰 값만 저장하여 해당 어플리케이션 서버로 보상을 요청하여 보상을 받는다.

[그림 5] C사의 동영상 보상 적립 내역

[그림 3]과 [그림 4]처럼 보상을 요구하는 패킷 정보를 저장한 악의적인 공격자는 다시 서버로 보상을 요청하는 패킷을 자동적으로 전송하는 악성 프로그램을 제작하여 재전송함으로써 부적절한 보상[그림 5]을 얻을 수 있는 것으로 확인되었다. 이와 같이 광고 어플리케이션에서 보상을 요청하는 패킷을 악의적인 사용자가 재전송하여 서버에게 재전송하면 서버는 이와 같은 행위가 정상적인 사용자가 요청한 것으로 착각하여 금전적인 보상을 요청한 악의적인 사용자에게 광고에 대한 보상을 [그림 5]처럼 악의적인 사용자에게 포인트를 지급 한다. 이처럼 악의적이게 보상을 받게 되면 광고 어플리케이션 업체와 사용자는 금전적인 이득을 취할 수 있지만 광고주는 계속해서 금전적인 피해를 볼 수 있게 된다.

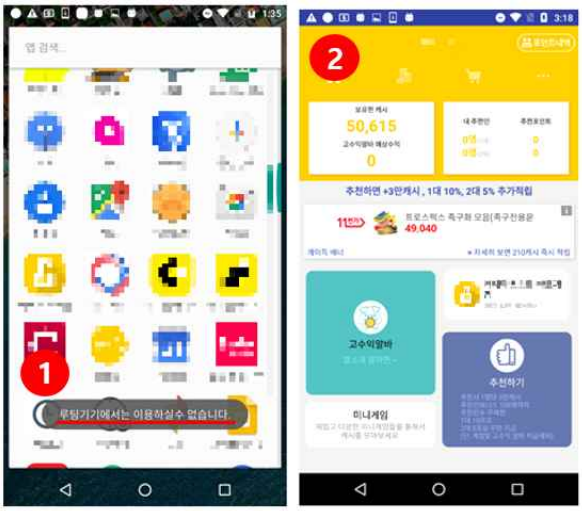
3.2 루팅 우회

루팅[2]이란 안드로이드폰의 운영체제를 해킹해 관리자의 권한을 얻는 행위를 말하며, 리눅스에서 관리자 권한을 얻는 행위를 지칭하는 용어에서 파생되었다. 안드로이드는 리눅스를 운영체제로 사용하는데, 리눅스에서 최고 권한을 가진 계정은 루트(root)이다. 즉, 루팅으로 안드로이드 운영체제의 사용자 권한을 '슈퍼 유저'로 바꿔 안드로이드 운영체제가 지원하지 않는 기능을 추가하거나 지원하는 기능을 삭제할 수 있다. 한 마디로 말하자면 최고의 관리자 권한을 얻는다는 의미이다. 루팅된 안드로이드의 장점은 모든 프로그램을 제어 할 수 있으며 스마트폰을 악의적인 공격자가 마음껏 변경할 수 있다. 이러한 장점 때문에 스마트폰 제조사뿐만 아니라 어플리케이션 업체에서도 루팅된 안드로이드에서 어플리케이션이 실행되지 않게 하기 위해 루팅 탐지 기법[3]을 통해 루팅된 스마트폰을 탐지한다. 국내 동영상 광고 어플리케이션 중에서도 루팅 탐지 기법을 통해 루팅을 탐지하였지만 C사의 경우 루팅 탐지 기법이 단순히 특정 명령어[그림 6]의 존재 유무로만 탐지 하는 것으로 확인되었다.



[그림 6] C사의 루팅 탐지 기법

C사의 광고 어플리케이션에서는 [그림 6]처럼 특정 명령어 'su'라는 루팅 명령어가 실행되는 유무에 따라 루팅을 탐지하는 것으로 확인되었다. 이렇게 간단한 루팅 탐지기법은 악의적인 사용자가 해당 명령어만 변경하게 된다면 루팅 탐지기법을 우회 할 수 있다. [그림 7]은 악의적인 사용자가 루팅 명령어를 변경하여 루팅 탐지 기법을 우회한 것을 보여준다. 1번은 처음 루팅된 스마트폰에서 C사의 어플리케이션을 실행한 모습이고 2번은 특정 명령어(su)를 다른 명령어로 변경하였을 때 루팅 탐지 기법이 우회된 모습을 보여준다.



[그림 7] C사의 루팅 탐지 기법 우회

3.3 광고 어플리케이션 취약점 결과

본 논문에서 소개한 3.1, 3.2절과 같은 분석 방법으로 국내 9개의 동영상 광고 어플리케이션을 분석하였다. [표 1]에서 볼 수 있듯이 국내 9개의 광고 어플리케이션 중 6개의 업체에서 광고 보상 요청 패킷을 재전송하였을 때 서버에서는 이를 탐지하지 못하고 사용자에게 보상을 지급한 것으로 확인되었다.

	루팅 탐지	패킷 암호화	재전송 공격	보상 여부
A사	X	O	O	O
B사	X	O	X	X
C사	O	O	O	O
D사	X	X	X	X
E사	X	O	O	O
F사	X	O	X	X
G사	O	O	O	O
H사	X	O	O	O
I사	X	O	O	O

[표 1] 국내 9개 광고 어플리케이션 점검 결과

이러한 취약점이 존재하는 이유는 광고 어플리케이션의 비즈니스 모델 특성상 광고주 보다 광고 어플리케이션 업체가 금전적인 이득을 볼 수 있는 구조로 되어있어 어플리케이션 업체에서는 크게 보안적인 측면에 대해 충분히 고려하지 않는다. 또한 많은 어플리케이션 개발 업체에서는 비용적인 측면에서 자신들이 만들고자하는 어플리케이션과 비슷한 어플리케이션의 APK 파일에서 소스코드를 추출하여 소스코드를 약간만 수정하여 런칭하는 경우도 많다. 이러한 경우 한 어플리케이션에서 발생하는 보안 문제점은 또 다른 어플리케이션에서도 똑같은 문제점으로 존재할 확률이 높다. 이와 같은 점 때문에 어플리케이션 업체에서는 더욱더 안전하게 개발 할 필요성이 있다.

IV. 취약점 개선방안

3.1절에서 설명한 재전송 공격을 막기 위해서는 서버에서 사용자의 비정상적인 행위에 대해서 탐지하는 방어기법이 존재하여야 한다. 예를 들어 사용자가 동영상 광고를 시청한 후 다음 광고 동영상 시청까지의 제한 시간이 60분이라고 가정하면 사용자는 60분 후에 동영상 광고에 대한 보상 패킷을 서버에게 전송할 수 있다. 하지만 60분 이내에 사용자가 보상을 요구하는 패킷을 서버에게 보내게 되면 서버는 이러한 행위에 대해 비정상적인 행위로 간주하여 보상을 요구하는 패킷을 무시하거나 계속해서 비정상적인 행위를 한 사용자를 차단하여 재전송 공격기법을 방어 할 수 있다. 또 다른 방어 기법으로는 서버에서만 생성하고 확인할 수 있는 값을 일정 시간마다 사용자들에게 전송하여 사용자가 광고 동영상을 시청한 후 서버에게 보상을 요청할 때 서버에서 생성한 값을 함께 서버로 보내 서버가 이를 확인하여 비정상적인 행위에 대해 탐지 및 방어 할 수 있다.

V. 결론

본 논문에서는 동영상 광고 어플리케이션의 취약점에 대하여 분석하였다. 국내 9개의 광고 어플리케이션 업체의 어플리케이션 중 6개의 어플리케이션에서 사용자의 어떠한 행동 없이 금전적인 보상을 받을 수 있는 취약점을 발견하였다. 6개의 어플리케이션에서 제공하는 현금과 포인트들은 현금으로 인출하거나 다른 상품을 구입 할 수 있기 때문에 광고자, 광고 어플리케이션, 기타 상품권 업체에게는 막대한 금전적인 피해가 있을 수 있다. 광고 어플리케이션은 앞으로도 많이 사용될 것으로 보이며 광고 어플리케이션 업체들은 프로토콜과 서버에 대해 보안 점검이 필요하다. 향후 연구에서는 다른 광고 비즈니스 모델에 대해서 분석해 볼 것이며, 부적절한 보상 요청에 대해 효과적인 보호 방법을 연구할 것이다.

[참고문헌]

[1] Cho, Geumhwan, et al. "Combating online fraud attacks in mobile-based advertising." EURASIP Journal on Information Security 2016.1 (2016): 2.

[2] 이형우. "안드로이드 기반 모바일 단말 루팅 공격에 대한 이벤트 추출 기반 대응 기법." 정보보호학회논문지, 23.3 (2013.6): 479-490.

[3] 정윤식, 조성제. "루팅된 안드로이드 폰에서 SEAndroid를 이용한 효과적인 앱 데이터 보호 기법." 정보과학회논문지, 44.4 (2017.4): 352-362.

안드로이드 앱 보호를 위한 동적 보호 기법 안티 에뮬레이션 분석 및 성능 평가*

김문회†, 오희국‡

한양대학교 컴퓨터공학과

Evaluating the Anti-Emulation Technique for Protecting Android Application on Runtime*

Wenhui Jin†, Heekuck Oh‡

Department of Computer Science and Engineering, Hanyang University.

ABSTRACT

Emulator detection is one of the Android dynamic protection techniques. Generally, It is used to be integrated into some application protection mechanism or services such as packer service to prevent significant information such as register which contains indirect target address or privacy sensitive app such as bank application from dynamic analyzed by hackers. With the technique development, methods which just compare the general features such as hardware, CPU, kernel, battery state can be modified as same as real devices easily. While there also have many effective methods to detect emulator, which also used in many applications security companies in practice. In this paper, we analyze various emulator detection methods and evaluate the corresponding detection performances.

I. INTRODUCTION

The Android application owns a feature which the package (APK) file can easily be analyzed especially by those hackers or who has an undesirable motivation. For this reason, application those sensitive to privacy or need to enhance the secure degree generally apply some protection mechanisms such as applying the packer service. In general, these protection methods can be classified into static protect method such as obfuscation in source code level or binary level and dynamic protect method such as anti-root, anti-emulation and anti-debug to prevent the static and dynamic analysis and ensures the apps execution context secure.

The emulator is a dynamic analysis method which runs an app on the emulated context, by this way hacker can get the register value which can be used to calculate the indirect control flow transfer target address or to analyze the app execution flow. For this reason, emulator detection method also called anti-emulation is necessary for those information sensitive apps.

In this paper, we will introduce some

* This work was supported under the framework of international cooperation program managed by the National Research Foundation of Korea (NRF-2017K2A9A2A06016147, FY2018).

And this research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2014-1-00636) supervised by the IITP(Institute for Information & communications Technology Promotion).

†Main author : jinwenhui@hanyang.ac.kr

‡Corresponding author : hkoh@hanyang.ac.kr

related works about emulators and emulator detection methods in Section 2. And we evaluate their the detection performance in Section 3. At least, we will have an conclusion and introduce future related works.

II. EMULATOR AND ANTI-EMULATION

When we consider this problem at a hardware and software components insight. Most widely[2] used way is to compare the hardware features such as IMEI, Model, CPU, Mac address information. In general Android emulator, these information are empty or filled related emulator's product information that different with the real android device. If we detect the software components such as baseband, board, Build.flavor information, these values as same as the hardware information which are empty or filled with the emulator related values. But unfortunately, with the development of Android emulators, these information also can be modified which looks like a real device by emulator or by modify the Build.prop files after rooting.

There also exit another real device features to consider: battery temperature, battery changes in fixed times, etc. But now Android Emulator also can emulate the battery state change such as Gentmotion, and what's more these kinds of detection uauslly uses the Android API to read the value. Whether emulator can also hook these API to prevent the detection[4], but you never know when the application call these API.

All these methods we mentioned above are so weak that emulator can easy to bypass or prevent. Petsas[2] purpose an significant anti-emulation method called Identifying QEMU execution using self-modifying code that utilizes the difference between arm architecture and x86 architecture.

Even though there exist Android phones running on x86 architecture, as the Table 1 shows, after 2015 almost all the main Android devices start to use the ARM architecture as Table 1. But the emulators are running on the PC which is x86 architecture. And most widely used Android emulators they use the QEMU emulator or virtual Box which also based on QEMU. QEMU translates instructions to be executed into the host architecture and execute. Thus, even though you execute an arm instruction in an ARM emulator, actually it will be translated into the host architecture instruction sets and send to host architecture to execute.

Device Model	Architecture	Instruction set	CPU bits
Samsung s9	ARM	ARMv8-A	64bit
Samsung s8+	ARM	ARMv8-A	64bit
Samsung s8	ARM	ARMv8-A	64bit
Samsung s7/edge	ARM	ARMv8-A	64bit
Samsung s6/edge	ARM	ARMv8-A	64bit
Samsung s5	ARM	ARMv7	32bit
LG G8	ARM	ARMv8	64bit
LG G7	ARM	ARMv8-A	64bit
LG G6	ARM	ARMv8-A	64bit
LG G5	ARM	ARMv8-A	64bit
LG G4	ARM	ARMv8-A	64bit
LG G3	ARM	ARMv7	32bit
Nexus 6P	ARM	ARMv8-A	64bit
Nexus 5X	ARM	ARMv8-A	64bit
Nexus 6	ARM	ARMv7-A	32bit
Nexus 5	ARM	ARMv7	32bit

Table 1 Android Device Architecture since 2015

ARM uses RISC instructions sets and the modified Harvard memory architecture that different within x86 which uses CISC instruction sets. as graph , In ARM the instruction and data are split into different

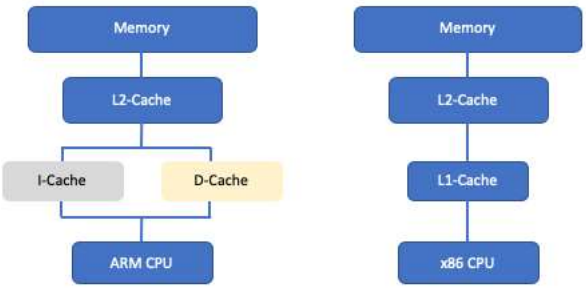


Figure 1 Cache on ARM and x86

cache independently. While in x86 the instruction and data they use the same cache as shown in Figure 1. If the self-modifying code modifies some instructions in x86, the modified instructions will update to cache rapidly.

In ARM system will consider the modified instructions as modified data and will update to the data cache(D-cache) as a result if in a loop context, after the instruction modified CPU will continue to execute the old instruction. Because this feature, we can detect the emulator exactly.

III. EVALUATION

In this section, we introduce the details of the experiment on emulator and Android real device. We classify the emulator detection method into 2 categories: architecture lever and specificated feature level.

We first select the widely used Android emulator: Android studio default emulator, Genymotion, Nox, Blustacks comparing with the Samsung s8+ real device as Table 2 shown.

We select some features frequently used on the emulator detection tools on open source. These features can be detected at the first time without any changing. After we change the emulator settings of the camera,

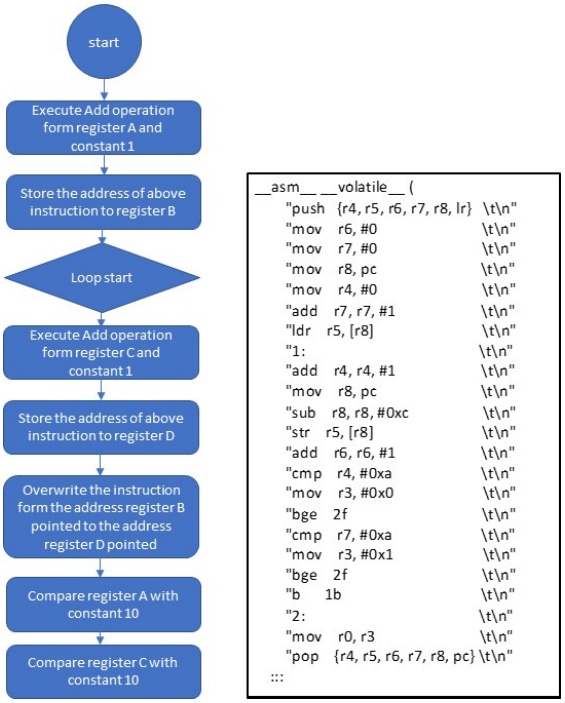


Figure 2 Self-modifying code implementation

battery state, sensor, wifi, phone number. And other information from Build.prog files such as MAC address, CPU, Model, IMEI could be changed after you get the root privilege. As a result, all these features can be modified as same as a real device.

Then we implement some handwriting arm assembly code in Figure 2 that an instruction *add r7, r7, #1* overwrite another instruction *add r4, r4, #1* in a loop structure. In the next loop turn, if the updated instruction *add r7, r7, #1* executed we thought the execution context is in x86. Otherwise, if CPU continues to execute the *add r4, r4, #1* instruction we thought the execution context is in real arm architecture.

During our experimentation, in Nox and Blustacks emulators can not select the arm architecture. So we test the Samsung s8+, Android studio emulator, Genymotion which can emulate arm architecture Android virtual machine on PC, except the real device such as Samsung s8+ or LG G3, CPU execute the

modified instruction on other emulators as shown on Table 3.

Virtual Device	IMEI	Model	CPU	MAC	Battery/ temperature/ state/ changes	camera
Android Studio	O	O	O	O	O	O
Genymotion	O	O	O	O	O	O
Nox	O	O	O	O	O	O
Blustacks	O	O	O	O	O	O
Samsung S8+	O	O	O	O	O	O

Table 2 Usually used features to detect the emulator

Test Divice	modified instruction execution
Android studio	O
Genymotion	O
Samsung s8+	X
LG g3	X

Table 3 Self-modifying code experimentation result.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduce the most used features and methods used to detect the emulator and also introduce an emulator detection method that still robust enough until now.

Moreover, we use various emulators and devices to evaluate these emulator's detection performance[4].

While beside the emulator detection methods mentioned in this paper, there also have other methods which use the machine learning methods to detect the emulator. And CPU bug to detect the emulator.

In the later, we will continue to analyze the better methods that can be used to detect the emulator. And this experimentation is about the ARMv7(32bit) architecture, so we will also analyze the evaluate the AArch64 architecture and find some key features. Moreover, we also can develop an novel anti-emulator method based on this concept.

[References]

[1] Petsas, Thanasis, et al. "Rage against the virtual machine: hindering dynamic analysis of android malware." Proceedings of the Seventh European Workshop on System Security. ACM, 2014.

[2] Blem, Emily, Jaikrishnan Menon, and Karthikeyan Sankaralingam. "A detailed analysis of contemporary arm and x86 architectures." UW-Madison Technical Report (2013).

[3] Aafer, Yousra, et al. "Precise Android API Protection Mapping Derivation and Reasoning." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018.

[4] Amos, Brandon, Hamilton Turner, and Jules White. "Applying machine learning classifiers to dynamic android malware detection at scale." Wireless communications and mobile computing conference (iwcmc), 2013 9th international. IEEE, 2013.

안드로이드 어플리케이션의 OAT 파일 무결성 검증

김수린+, 최주섭+, 김형식*

성균관대학교 전자전기컴퓨터공학과+, 성균관대학교 소프트웨어학과*

soolinkim@skku.edu, cjs1992@skku.edu, hyoung@skku.edu*

Runtime integrity verification of OAT files for Android application

Soolin Kim+, Jusop Choi+, Hyoungshick Kim*

+Department of Computer Science and Engineering, Sungkyunkwan University

*Department of Software, Sungkyunkwan University

요 약

안드로이드는 어플리케이션의 실행 속도를 향상시키기 위해 Dalvik 가상머신 대신 Android RunTime (ART) 방식을 제안했다. 기존 안드로이드 5.0 이후에 런타임 방식이 변경되었으며, 안드로이드 8.0 업데이트에서는 APK 설치 시 생성되는 OAT 파일들의 구조적 변화가 있었다. 그러나 안드로이드 8.0 업데이트로 구조적으로 변경된 OAT 파일들에 대한 무결성 검증에 대한 연구는 미비한 상태이다. 본 연구에서는 안드로이드 8.0의 OAT 파일에 대한 캐시 파일 변조 공격의 가능성을 확인한다. 또한 변조 공격으로 캐시 파일들이 변경됐을 경우, 실제 어플리케이션의 동작을 바꿀 수 있다는 것 또한 실험을 통해 확인한다. 마지막으로 OAT 파일 변조를 방지하기 위한 검증 시스템을 제안하고 실제 15 개의 어플리케이션을 통해 성능을 확인한다.

I. 서론

최근 모바일 보안에 대한 보안 연구의 흐름은 안드로이드 설치 파일인 Android application PacKage (APK)에 대한 무결성을 검증, 루팅 기기 탐지, 코드 난독화 등과 같은 솔루션을 연구하는 방향으로 진행되고 있다.

각종 보안적인 결함 혹은 기능적 결함을 개선하기 위하여 안드로이드는 운영체제를 업데이트한다. 이러한 업데이트 과정에서 이전의 안드로이드 운영체제와 업데이트 된 운영체제 간 다른 솔루션이 적용되곤 한다. 안드로이드 운영체제 (5.0 이후)에서는 APK 파일 설치 시 Dalvik 가상머신 대신 Android RunTime (ART)가 dex2oat[1)]를 이용하여 Of Ahead Time (OAT) 파일들로 저장한다 [1]. ART는 Dalvik 가상머신과 같은 가상머신의 일종으로 안드로이드 5.0 이상의 운영체제에서 실행되는 기기의 기본 런타임 방식이다. 기존의 안드로이드 운영체제에서는 DEX 파일을 Dalvik 가상머신 위에 올려두고 필요에 따라 실시간으로 컴파일 하는 방식이었다. 그러나 안드로이드 5.0 이상의 운영체제에서는 ART에서 DEX를 미리 컴파일 하여 OAT 파일에 저장하고 실행 시 OAT 파일에 저장되어 있는 최적화 된 DEX를 실행하는 방식으로 업데이트 되었다 [3].

업데이트로 인하여 새로운 OAT 파일에 대한 무결성이 위협받고 있으며 이를 방지하기 위해 OAT에 대한 무결성을 보장할 수 있는 기술 연구가 필요하다 [2]. 이러한 위협을 완화하기 위해 안드로이드 운영체제 업데이트로 인해 컴파일 환경 변화가 생긴 캐시 파일들에 대한 무결성 검증 연구가 진행되었다 [2].

기존 연구 방법은 어플리케이션 당, 기기와 운영체제 버전, 프로파일 등에 따라 많은 양의 hash 값이 생성되는 문제점이 존재한다. 기존

1) https://android.googlesource.com/platform/art/+/kitkat-release/dex2oat/dex2oat.cc

연구에서 제안한 방법은 한정된 저장 공간을 가진 모바일 기기에서는 비효율적이며, 데이터베이스의 크기가 증가할수록 관리가 어려워지고, 탐색 시간이 길어지기 때문에 어플리케이션이 실행될 때 검증이 필요한 경우 검증 시간의 오버헤드가 증가하는 문제점이 존재한다. 또한, OAT 파일 중 ODEX 파일을 보호하기 위한 시스템을 제안하였다.

본 논문에서는 ART 환경에서 동작하는 Android 8.0을 기반으로 새롭게 생성되는 OAT 파일인 VDEX에 대한 무결성을 검증하는 시스템을 제안하고자 한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 OAT 파일에 대한 공격 기술 분석하고 OAT 파일에 대한 변조 공격을 수행하기 위한 과정 및 결과를 서술한다. Ⅲ장에서는 섹션 Ⅱ장에서 OAT 파일에 대한 변조 공격을 막기 위한 무결성 검증 시스템을 제안한다. Ⅳ장에서는 선정한 15가지 어플리케이션에 제안한 검증 시스템을 적용하여 실험한 후 그 결과를 서술하며, Ⅴ장에서는 향후 연구를 위한 내용 및 결론을 서술할 것이다.

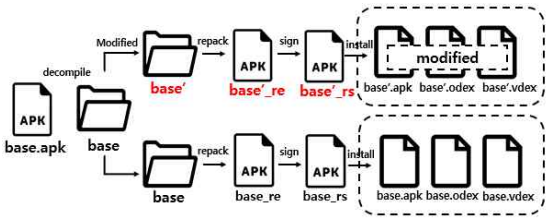
II. OAT 공격 기술 분석 및 결과

2.1 OAT 공격 기술 분석

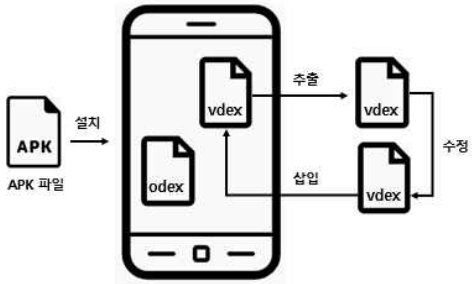
DEX는 어플리케이션의 실행과 관련한 바이트코드이며 안드로이드는 설치 파일인 APK 파일의 DEX를 최적화 하여 캐시 파일 형태로 저장한다. 안드로이드 8.0 이전의 운영체제에서는 base.odex에 APK 파일을 컴파일 하여 저장하였다. 특히, base.odex에 base.apk의 DEX를 최적화하여 저장하였다. 그러나 Android 8.0 운영체제가 탑재된 기기부터는 APK 파일 컴파일 시 dex2oat에 의해 OAT 파일인 base.odex파일과 base.vdex파일이 생성된다. 안드로이드 7.0과 달리 안드로이드 8.0에서는 APK 파일의 DEX가 최적화 되어 base.vdex에 저장되며, 어플리케이션 실행 시 base.vdex에 저장되어있는 최적화 된 DEX 부분이 실행된다.

2.2 OAT 파일 변조 공격 결과

OAT파일 변조 공격을 수행하기 위해서 어



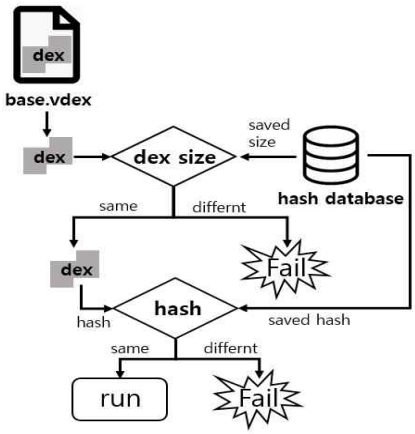
[그림 1] 변조 영역을 확인하기 위한 과정



[그림 2] 캐시 변조 공격 과정

플리케이션에 실제 실행 코드가 최적화 되어 저장된 base.vdex 파일에 대하여 캐시 변조 공격을 시도하였다. 변조할 영역을 쉽게 알아내기 위해서 [그림 1]과 같이 원본 base.apk 파일을 변조하여 base'.apk를 생성한 후 설치하여 base'.odex, base'.vdex 파일을 얻어낸다. 원본 base.apk를 설치하여 생성된 파일들인 base.odex와 base.vdex를 추출하여 변조된 파일들과 비교하였다. 그 결과 base.apk 파일과 base.vdex 파일에서는 원본과 변조된 파일사이에 많은 차이점이 발생하며, base.odex 파일에서는 4 바이트를 제외한 나머지 부분에서 전부 일치하는 것을 확인하였다.

OAT 파일 변조를 위해 [그림 1]의 실험 결과를 토대로 base.vdex 파일을 [그림 2]과 같이 파일 변조 공격을 설계하였다. base.vdex 파일은 크게 3가지 파트(Dexfiles, Verifier dependencies, Quickening Info)로 구성되어 있다. 실질적으로 프로그램 코드 영역인 Dexfiles 영역의 값을 변조하여 OAT 파일 변조 공격을 진행하였다. APK 파일을 설치 후 해당 APK 파일의 base.vdex 파일을 adb를 통해 추출하였다. 추출한 base.vdex 파일 내의 문자열을 변경하여 기기 내부에 다시 삽입하였다. 해당 어플리케이션을 다시 실행하였을 때 임의로 바꾼 문자열로 바뀌어 실행되었다.



[그림 3] OAT 무결성 검증 시스템

III. OAT 무결성 검증 시스템 제안

3.1. OAT 무결성 검증 시스템 설계

안드로이드 8.0 에서는 base.apk의 DEX부분이 dex2oat에 의해 최적화되어 base.vdex의 Dexfiles 영역에 저장된다. 이러한 특성을 이용하여 [그림 2]과 같이 OAT 파일 변조 공격을 진행했다. base.vdex의 Dexfiles 영역의 값을 변경하여 APK을 변조하는데 성공하였다. 따라서 base.vdex의 Dexfiles 영역에 대한 변조를 막기 위한 검증 시스템을 [그림 3]와 같이 구성하였다.

본 시스템([그림 3])은 안드로이드 8.0의 ART 환경에서 어플리케이션이 설치 및 실행될 때 dex2oat에 의해 생성되는 base.vdex에 대한 무결성을 검증하는 방식이다. 검증을 위하여 사전에 hash 값과 DEX의 크기를 저장하고 있는 데이터베이스가 존재해야 한다.

1. 실행하려는 어플리케이션의 실제 작동 코드가 들어있는 base.vdex에 접근하여 DEX 영역을 추출한다.
2. 추출한 DEX의 크기를 설치할 때 생성된 데이터베이스에서 DEX의 크기와 상호 비교한다. 이때, 서로 값이 다르면 5 번으로 이동하고 값이 동일하면 3 번 과정을 수행한다.
3. 추출한 DEX의 해시 값을 계산하고, 계산된 해시 값을 데이터베이스의 저장된 해시 값과 비교한다. 비교 결과 값이 다르면 5 번으로 이동하고, 값이 동일하면 4 번을 수행한다.

[표 1] 검증 시스템에 사용된 데이터 셋

카테고리 명	어플리케이션 명
금융	K* Bank
	A* Bank
	P*
생산성	O*
	On*
	D*
	S*
비즈니스	A* Authenticator
	B* Authenticator
의료	T*
도구	G* Authenticator
커뮤니케이션	F* messenger
소셜	Pi*
음악/오디오	A*
여행 및 지역정보	B*

4. 실행 파일 데이터의 변조가 일어나지 않았다고 판단하고 어플리케이션을 실행한 뒤, 무결성 검증 시스템을 종료한다.
5. 캐시 변조 공격이 발생했다고 판단하여, 어플리케이션을 실행하지 않고 종료한다.

IV. OAT 무결성 검증 시스템 실험 결과

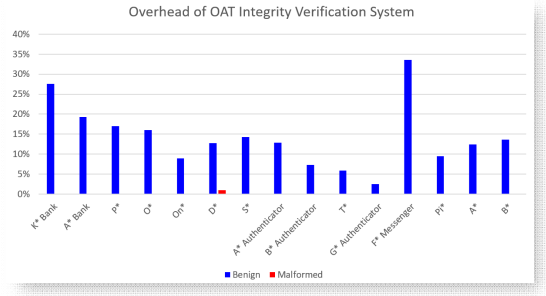
4.1. 실험 환경

APK 파일의 대한 변조가 사용자에게 금전적 손해를 입히거나, 민감한 개인 정보가 유출되는 등의 심각한 상황을 발생시킬 수 있기 때문에, APK 파일의 변조를 탐지하는 로직이 있는 어플리케이션을 선정하였다. 9 개의 영역에서 총 15 개의 어플리케이션을 [표 2]과 같이 선정하였다.

실험에는 실제 기기인 Nexus 5x를 사용했으며, 한국에서 출시된 기본 모델로, 안드로이드 8.0.0이 탑재된 상황에서 실험하였다.

4.2. 측정 방식 및 결과

캐시 변조 공격이 발생한 상황을 가정하기 위해서 [표 2]에서 선정한 15 개의 어플리케이션의 OAT 파일을 수작업을 통해 변조했고, 실제 상황과 유사하도록 실제 기기에서 값을 측정하였다.



[그림 4] OAT 무결성 검증 시스템 오버헤드

무결성 검증 시스템이 어플리케이션의 실행 시간에 주는 부담을 측정하기 위한 검증 시간 오버헤드와 기기에 주는 부담을 확인하기 위한 저장된 데이터베이스의 크기, 검증 시스템의 성능을 측정하기 위한 캐시 변조 공격 탐지율과 오탐률을 측정하였다.

OAT 무결성 검증 시스템의 오버헤드는 어플리케이션의 실행 시간과 검증 시간을 각각 구한 후, [수식 1]의 공식을 이용하여 계산하였다. 측정 오차를 줄이기 위해 각 어플리케이션 당 오버헤드 비율을 10 회를 측정한 뒤, 평균을 구하였다. 각 어플리케이션 당 오버헤드 비율은 [그림 4]와 같다.

$$\text{검증시간오버헤드비율}(\%) = \frac{\text{검증시소요되는시간}}{(\text{어플리케이션실행시간} + \text{검증시소요되는시간})} * 100$$

[수식 1] 무결성 검증 시간 오버헤드 계산식

검증에 걸리는 시간은 VDEX 파일 내의 DEX 영역의 크기와 비례하는 것을 확인할 수 있다. 즉, 어플리케이션의 코드 크기가 증가할수록 검증에 걸리는 시간 또한 증가한다.

OAT 무결성 검증 시스템의 결과는 [표 3]에 정리하였다. 총 검증에 걸리는 시간에 대한 오버헤드 비율은 7.15 %로 어플리케이션의 평균 실행 속도인 3.48 초에 대비하여 사용자가 큰 체감을 느끼기는 힘들다. 또한 해시 값을 이용한 검증 시스템의 특성상 탐지율과 오탐률은 각각 100 %와 0 %를 달성하였다. 그리고 기존 연구 [2]에 대비하여 저장해야할 데이터베이스의 크기는 32 Bytes로 줄이는 것이 가능했다.

[표 2] 무결성 검증 시스템 실험 결과

항목	결과
무결성 검증 시간 오버헤드 비율	7.15%(σ=0.09)
코드 변조 탐지율	100%
저장된 데이터 크기	32Byte
코드 변조 오탐률	0%

V. 결론 및 향후연구

본 논문에서는 안드로이드 8.0 운영체제에 새롭게 적용된 OAT 파일에 대한 캐시 변조 공격을 진행 하였으며 공격을 탐지하기 위해 검증 시스템을 제안하였다. 본 논문에서 제안한 검증 시스템은 사용자가 느끼기 힘든 시간 내에 탐지율과 오탐률을 각각 100 %와 0%를 달성하였고, 저장된 데이터베이스의 크기는 32 Bytes로 줄이는 것에 성공하였다. 그러나 OAT 파일 무결성 검증 시스템은 기기에 종속적이라는 한계점을 가지고 있다. 같은 APK를 설치하여 다른 제조사의 기기에서 확인한 결과 hash 값이 서로 다른 것을 확인하였다.

추후 연구를 통하여 base.vdex와 base.odex의 유기적인 관계를 분석하여 기기에 종속적이지 않은 OAT 파일 무결성 검증 시스템을 연구할 계획이다.

[참고문헌]

[1] Purkayastha, D. S., and N. Singhla. "Android Optimization: A Survey." International Journal of Computer Science and Mobile Computing-A Monthly Journal of Computer Science and Information Technology 2.6 (2013): 46-52.

[2] Wan, Jia, et al. "Defending Application Cache Integrity of Android Runtime." International Conference on Information Security Practice and Experience. Springer, Cham, 2017.

[3] Android Open Source Project, Configuratin g ART, https://source.android.com/devices /tech/dalvik/configure, 9, Nov, 2018.

사용자의 공인인증서에 관한 인식과 브라우저 선택에 미치는 영향 연구

우지혜* 전소윤** 최지원* 김형식*** 우사이먼*

한국뉴욕주립대학교 컴퓨터과학과*, 응용수학통계학과**
성균관대학교 소프트웨어학과***

Understanding Users' Perception on Digital Certificate and Their Web Browser Usages in Korea

Jihye Woo* Soyoon Jeon** Ji Won Choi* Hyoungshick Kim*** Simon S. Woo*

*Computer Science, SUNY Korea

**Applied Mathematics & Statistics, SUNY Korea

***Sungkyunkwan University Korea

요 약

한국 인터넷 이용자들은 본인인증을 위해서 공인인증서를 사용한다. 공인인증서의 자체적인 보안은 뛰어나지만, 부가적인 프로그램 설치 과정이 사용자에게 불편함을 일으키고 있으며, 공인인증서의 사용을 위해서는 특정 브라우저와 운영체계에 의존하는 경향이 있다. 본 연구에서는 사용자의 공인인증서에 대한 보안성과 편의성에 대한 인식조사 및 공인인증서가 사용자의 브라우저 선택에 미치는 영향들을 유저스터디를 통하여 분석하였다. 본 연구에서 62명의 지원자 중 92%가 공인인증서가 제공하는 보안성을 제대로 이해하고 있지 않으며, 64%가 불편함에도 불구하고 공인인증서를 사용하고 있다는 것을 보여준다. 또한, 공인인증서가 사용자의 브라우저 선택에 큰 영향을 미친다는 사실을 정량적 유저스터디를 통해 보여준다.

I. 서론

공인인증서란, 인터넷상에서 전자거래 등을 안심하고 사용할 수 있도록 해주는 사이버 증명서이다[5]. 공인인증서를 사용하기 위해서는 이용자가 직접 전자서명 용도의 추가 프로그램(ActiveX, exe)을 설치해야 한다. ActiveX는 웹사이트 호환성이 낮고, 보안 위협으로부터 안전하지 않다는 단점이 있어 최근에는 Java Applet 형태의 공인인증서 사용이 증가하였으나 아직 문제점들이 존재한다. 공인인증서는 초기에 IE 환경에서 작동하도록 만들어졌기 때문에 다른 브라우저에서 공인인증서 발급과 사용 절차가 까다롭고 복잡하다. 이는 IE 환경에서 공인인증서를 사용하도록 하여 이용자의 브라우저 선택에 영향을 주는 직접적인 요인으로 작용할 수 있다. 이렇듯 공인인증서는 보안 위협, 복잡한 절차 그리고 브라우저 선택권 침해 등 사용자와 직접적으로 연관된 문제들이 있지만, 이용자의 관점에서 공인인증서의 사용에 관한 인식조사와 문제점 개선방법을 모색하기 위한 관련 연구는 미비한 편이다. 본 연구에서는 사용자 측면에서 공인인증서의 사용과 보안성 등에 대한 사용자의 인식조사와 이런 요인들이 사용자의 실제 브라우저 선택에 미치는 영향을 정량적 온라인 유저스터디를 통하여 분석하였다.

II. 공인인증서 Background

1999년 7월 1일 전자서명법이 발효되면서 공인인증서는 안전한 온라인 상거래 체계를 구현하기 위해 도입되었다[2]. 하지만 ActiveX의 문제가 지속적으로 제기되자 정부는 파일방식(exe)을 사용하여 모든 브라우저에서 공인인증서를 사용할 수 있게 하였다. 또한, 최근 들어 Java Applet을 통한 웹 호환성이 높고 다양한 운영체제에 사용 가능한 프로그램의 사용이 높아지고 있다. 그러나 여전히 IE가 아닌 환경에서는 공인인증서 사용이 불편하므로 웹 브라우저의 선택권이 제한되었다[1]. 그리고 공인인증서는 서비스 제공자 중심이 아닌 사용자 중심의 인증방식이다. 이는 사용자에게 서비스 제공자가 요구하는 조건의 마이크로소프트 제품과 그에 따른 운영체제 서비스를 사용하도록 요구한다. 또한, 공인인증서로부터 야기된 문제점들에 대한 책임을 사용자들에게 전가하는 경향이 해외와 가장 큰 차이점 중의 하나이다[2]. 이런 문제점들을 해결하기 위하여 2015년 3월 18일 금융위원회는 전자금융거래의 기술 중립성 구현과 전자지급수단의 사용 활성화를 위해 전자금융감독규정을 개정하고 공인인증서 의무사용 폐지를 발표하였다[3]. 최근 2018년 1월 22일 규제혁신토론회에서 과학기술정보통신부는 현행 공인인증서 제도의 문제점을 해결하기 위해 공인인증서 제도 폐지 방침을 발표하였다. 이를 통해 정부는 다양한 전자서명 기술·서비스가 시장에서 동등하게 경쟁할 수 있는 제도적 여건을 조성하려 한다[4].

III. 유저스터디 설계 및 실험

공인인증서의 사용자 인식 및 실태조사 그리고 공인인증서와 브라우저의 관련성 조사를 위하여 크게 다음의 항목들에 대해 총 27개의 질문을 통하여 설문조사를 시행하였다. 1) 공인인증서의 저장장소, 2) 공인인증서 사용 이유, 3) 공인인증서의 편의성, 4) 공인인증서의 안전성, 5) 공인인증서의 취약점, 그리고 6) 각 브라우저의 성능 7) 멀티브라우저의 사용 의도에 대해 질문하였다.

설문조사 시 참가자들의 선호도 분석은 Likert 0-10 스케일 (0: 매우 동의하지 않는다, 5: 중립, 10 : 매우 동의한다)을 이용하였다. 그리고 이유를 묻는 질문사항은 개방형 답변으로 구성하였다.

IV. 유저스터디 결과

페이스북과 카카오톡을 통해 62명의 지원자에게 설문조사를 시행하였다. 조사대상의 83%가 20대 대학생이었고, 여성의 비율이 10% 정도 더 많았다.

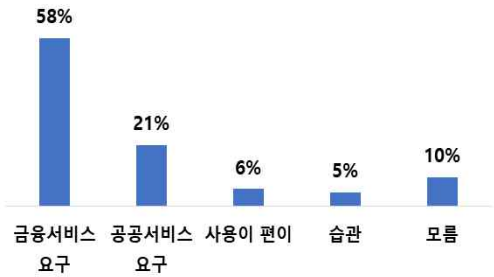
4.1 공인인증서 저장장소 분석

공인인증서를 저장할 수 있는 장소에 대하여 62%의 사용자들은 컴퓨터의 내부저장소에 저장한다는 결과를 얻었다. 그 이유를 분석해 본 결과 컴퓨터 내부저장소에 저장하는 이유는 브라우저에서 공인인증서 사용이 잦기 때문이며, 자주 사용하는 기기이기 때문이라는 결과를 얻었다.

4.2 의무사용제도 폐지 인식조사

2015년 이후로 전자금융거래상에서는 공인인증서 이외의 본인인증수단을 선택 할 수 있게 되었다. 이 사실을 인식하고 있는지에 대해 조사해본 결과 놀랍게도 65% 이상의 공인인증서 사용자들은 공인인증서의 의무사용 제도가 폐지된 것을 인지하지 못하고 있다. 이러한 결과는 다수의 공인인증서 사용자들이 본인인증 혹은 개인 보안 관련 이슈에 대한 관심이 적다는 것을 보여준다.

4.3 공인인증서 사용이유 분석

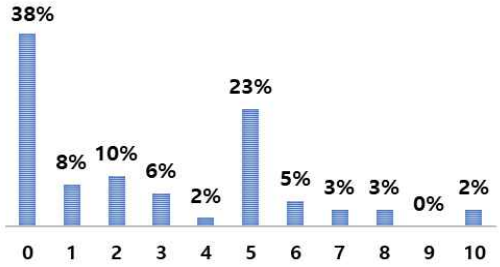


[그림 1] 공인인증서 사용이유

설문결과([그림 1]) 아직 79%는 서비스 제공자(은행 등 금융서비스 업체, 정부 사이트 등)의 요구 때문에 공인인증서를 사용하고 있다고 답했다. 이는 전자금융거래 시 공인인증서의 의무사용제도가 폐지되었음에도 아직 인증수단으

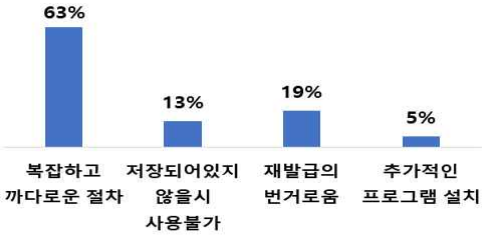
로 공인인증서를 고수하는 서비스 제공업체들의 영향이 적지 않음을 의미한다.

4.4 공인인증서의 편리성 분석



[그림 2] 공인인증서 편리성 평가

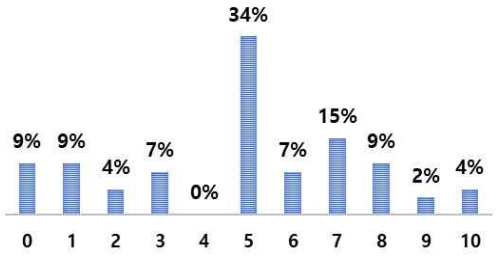
공인인증서의 사용자들에게 공인인증서를 이용한 인증방식의 편의성을 조사해보았다. 공인인증서를 사용할 때, 편리하다고 생각한다는 질문에 (0:매우 동의하지 않는다, 5: 중립, 10: 매우 동의한다) 64%는 공인인증서를 통한 인증이 편리하다는 점에 동의하지 않는다고 답하였다. 또한, 23%는 공인인증서가 편리하다는 점에 중립을 택하였다. 이를 더 자세히 알아보기 위하여 추가로 공인인증서의 불편한 점을 알아보았다.



[그림 3] 공인인증서의 불편한 점

공인인증서가 불편하다고 생각하는 사람들에게 구체적인 이유를 물어본 결과, 37%는 공인인증서 인증방식의 복잡하고 까다로운 절차를 꼽았다. 또한, 조사대상의 11%는 공인인증서는 유효기간이 있으므로 기한이 지날 때마다 재발급해야 한다는 것을 불편하다고 여겼다. 그 외에는 저장소를 사용할 시점에 보유하고 있지 않거나(8%) 컴퓨터 또는 모바일에서 부가적인 프로그램 설치를 요구한다는 점(3%)을 꼽았다.

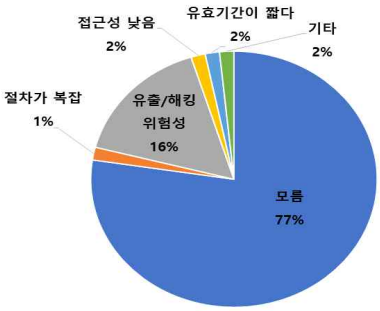
4.5 공인인증서 안정성 분석



[그림 4] 공인인증서 안전성 평가

[그림 4]는 공인인증서의 안전성에 대한 인식을 조사한 결과이다. 대상의 약 33%는 중립이라고 답하였고 중립을 기준으로 동의하지 않는다는 이용자들보다 동의한다고 답한 이용자들이 조금 더 많은 것으로 나타났다. 또한, 공인인증서가 어떠한 보안 위협을 해결할 수 있는지에 대한 질문에서는 놀랍게도 92%가 전혀 모른다고 답하였다. 이는 이용자들이 공인인증서의 어떠한 점이 안전한지 모르고 있다는 점을 보여준다. 현재 공인인증서의 취약한 안전성으로 인해 정부 또한 공인인증서의 인증방식을 변화하려는 상황에서 사용자들은 공인인증서의 안전성에 대한 인식이 아주 낮다는 것을 알 수 있다.

4.6 사용자 입장 공인인증서 취약점 분석



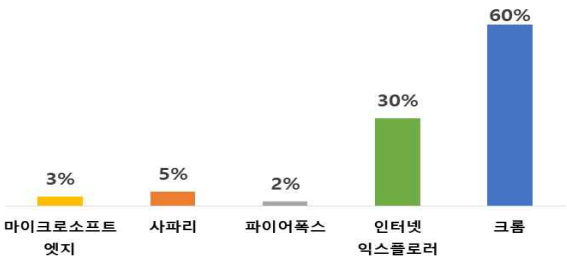
[그림 5] 공인인증서의 취약점 인식조사

공인인증서는 인증의 수단으로 보안이 핵심 기술이다. 그러나 현재 공인인증서를 사용하는 과정에서 보안 문제가 지속적으로 제기되고 있고 이는 공인인증서의 가장 큰 취약점이라 할 수 있다. 그러나 77%의 사용자들은 공인인증서의 취약점을 모른다고 답하였다. 그 다음은 유출 및 해킹의 위험성이 16%를 차지했다. 이는 이용자들이 공인인증서와 개인 보안에 관해 관

심이 부족하다고 해석될 수 있다.

V. 공인인증서와 브라우저의 상관관계

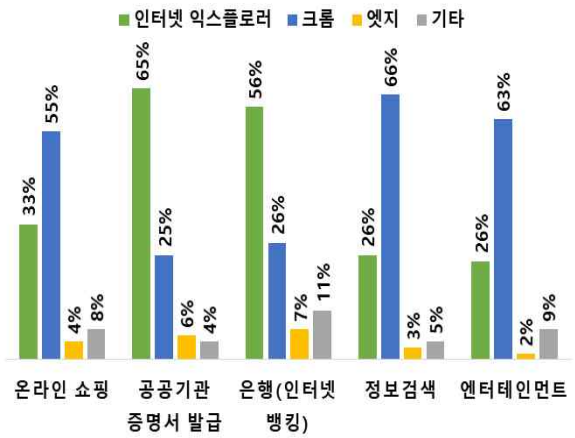
5.1 브라우저 이용도 분석



[그림 6] 데스크탑에서 각 브라우저 이용도

위 그래프는 가장 자주 사용하는 브라우저를 조사한 결과이다. 이용자들은 크롬(60%)을 가장 자주 사용한다고 답했으며 두 번째로 많이 선택한 브라우저는 IE(30%)이다. 이는 대다수의 해외 사용자들이 크롬(60.6%)을 가장 많이 사용하고 IE(2.85%) 사용하는 것과 비교하면[6] 위의 IE 사용률은 큰 차이를 보인다. 더 나아가 브라우저 선택에 대한 구체적인 이유를 분석하기 위해 다음 특정 브라우저의 사용 용도에 관해 질문하였다.

5.2 브라우저 사용 용도 분석



[그림 7] 각 서비스의 브라우저 이용도

[그림 7]을 보면 온라인 쇼핑, 정보 검색, 엔터테인먼트에서는 크롬 이용률이 55%, 66%, 63%를 차지하지만, 공공기관 증명서 발급, 은행 업무(인터넷 뱅킹) 등 공인인증서가 필요한 서비스 사용 시에는 IE 이용률이 65%, 56%를 차지한다. 위의 상반된 결과를 통해 특정 분야(금융과 공공 서비스 이용)에서는 IE가 크롬을 앞서고 있다는 것을 알 수 있다. 이를 통해 사람들이 각 서비스를 이용할 때 사용하는 브라우저가 두 개 이상이며 여러 개의 브라우저를 사용하는 데 있어 공인인증서가 큰 비중을 차지하고 있음을 알 수 있다.

VI. 결론

브라우저 사용 관련 조사 결과에서는 60%로 크롬이 가장 높은 이용률을 보였으나 공인인증서 사용이 불가피한 환경(공공기관, 은행)에서는 IE 이용률이 공공기관의 경우 65%, 은행의 경우 56%로 급격하게 높아졌다. 이를 통해 공인인증서가 개인의 브라우저 사용에도 영향을 미치고 있다는 사실이 밝혀졌다. 이는 해외의 브라우저 선호도와는 크게 다른 현상으로 공인인증서의 사용으로 인한 한국에서만 존재하는 특이한 현상으로 보인다. 본 연구에서는 사용자 측면에서 공인인증서와 브라우저 선택, 용도를 분석하였으면 앞으로 다양한 계층 간에 좀 더 많은 참가자를 대상으로 설문을 실행할 예정이다.

[참고문헌]

[1] H. Kim, J. H. Huh, and R. Anderson, "On the Security of Internet Banking in South Korea," Oxford University Computing Laboratory, CS-RP-10-01, 2010.

[2] Hun Myoung Park, The Web Accessibility Crisis of the Korea's Electronic Government: Fatal Consequences of the Digital Signature Law and Public Key Certificate, 2012

[3] 금융위원회, "전자금융감독규정 개정", http://www.fsc.go.kr/info/ntc_news_view.jsp?menu=7210100&bbsid=BBS0030&no=30287, 2015

[4] 과학기술정보통신부, '공인인증서 제도' 20년 만에 전면 폐지 추진, https://www.msit.go.kr/web/msipContents/contentsView.do?cateId=mssw311&artId=1378272, 2018

[5] 전자서명인증관리센터, "안전하고 간편한 공인인증서", https://rootca.kisa.or.kr/kor/accredited/accredited03_05.jsp

[6] 해외 웹 브라우저 이용률 http://gs.statcounter.com/browser-market-share

Mifare Classic 태그 타입 사용 출입 통제 시스템의 보안 취약점 및 대응방안에 대한 연구

고영건*, 문성훈*, 김재희*, 조대환*, 오병윤*, 유재욱*, 이원**, 신정훈***

*한국정보기술연구원 차세대 보안리더 양성 프로그램 (Best of the Best),
**드림에이스, ***티오리

A study on security and countermeasures of access control system using Mifare Classic tag

Young-gun Go, Sunghoon Moon, Jaehui Kim, Daehwan Jo, Byeongyun Oh, Jaewook You, Won Lee, Jeonghoon Shin

*Best of the Best, Korea Information Technology Research Institute,
**DRIMAES, ***Theori

요 약

출입 통제 시스템이란 가정과 회사 등의 안정성 확보를 위해 출입기록을 데이터화하여 비인가 사용자 접근을 제한하는 시스템을 말한다. 많은 기업에서 NFC를 이용하여 출입기록을 데이터화 하고 있는 현재, NFC 태그 타입으로 NXP 반도체 기술의 RFID인 Mifare Classic을 사용하는 경우가 많다. 오래된 Mifare Classic 태그 타입의 NFC는 태그 복제 위험성을 가지고 있어 영국 ITSO 규격에선 Mifare Classic 1K와 4K 칩을 제외한 바 있다. 본 논문에서는 Mifare Classic 태그 타입을 사용하는 출입 통제 시스템에서 태그 타입의 취약점에 관한 연구를 통해 Mifare Classic 기반 출입 통제 시스템에 대한 위험성을 소개하고 이에 대한 대응방안을 제시한다. 향후 본 논문에서 제시하는 대응방안을 적용한다면 Mifare Classic 태그 타입을 사용하는 출입 통제 시스템의 안전성을 보다 높일 수 있을 것이라 기대된다.

I. 서론

출입 통제 시스템이란 가정과 회사 등에서 인가된 사용자에게 출입할 수 있도록 제한하는 시스템을 말한다. 최근 비밀번호 인증, 생체인증, NFC 태그 등 다양한 방법의 사용자 인증 방식들이 나오고 있다. 본 논문에서는 많은 인증 방식 중 NFC 태그를 이용하는 출입 통제 시스템에 대해 연구하였다.

NFC는 13.56Hz 대역의 주파수를 사용하는 RFID 기술 중 하나이다. 양방향 통신이 가능하며 필요에 따라 태그 역할 뿐만 아니라 태그의 정보를 읽거나 쓰는 Reader/Writer 역할을 수행할 수 있다[1]. NFC 태그는 비접촉식 근거리 무선분야의 대표적인 기술을 모두 포괄하고 있으며 교통카드, 결제, 출입 통제 등 다양한 분야에 적용되고 있다[2].

그러나 다양한 NFC 태그 타입 중 전 세계 시장의 72.5%를 점유[3]하고 있는 Mifare Classic 태그는 태그와 리더기 간의 인증 프로토콜, 무작위 생성 함수, 인증데이터 생성에 사용하는 스트림 암호의 취약성으로 인해 복제 가능성이 존재한다[4]. 이와 같은 취약점으로 Mifare Classic 태그의 데이터를 복호화 하여 NFC 태그가 복제될 수 있다면 비인가 사용자의 접근을 허용할 수 있어 출입 통제 시스템이 가진 목적에 어긋나게 될 것이다.

본 논문에서는 Mifare Classic 태그 타입을 사용하는 출입 통제 시스템의 보안 취약점에 대해 연구하고 이에 대한 대응방안을 두 가지 관점으로 제시하고자 한다. 2장에서는 NFC 보안 기술 및 위협, Mifare Classic 취약점 및 공격 기법과 출입 통제 시스템에 대한 기존 연구와 본 논문의 취지를 설명한다. 3장에서는 Mifare Classic 1K의 취약점을, 4장에서는 출입 통제 시스템에서 발생하는 Mifare Classic 1K의 취약점과 분석 절차/결과에 관해 서술하며, 5장에서는 4장의 분석 결과에 대해 하드웨어와 소프트웨어 관점에서의 대응방안을 제시한다. 끝으로 6장에서 결론을 내며 본 논문을 마친다.

II. 관련 연구

본 장에서는 Mifare Classic 태그 타입에서 발생하는 취약점과 NFC를 활용한 출입 통제 시스템의 기술들에 대한 선행 연구들을 알아본다.

현재 교통카드, 휴대전화, 도어록 등 NFC 기술이 적용된 기기가 증가하고 있는 상황에서, NFC 기술의 보안 취약점과 대응방안에 관한 연구들이 진행되고 있다.

NFC 보안 서비스를 위하여 ECMA-385는 NFC-SEC 보안 구조를 정의하며 데이터 교환을 위한 보안 서비스 및 보안 프로토콜을 제시하였지만, 데이터 변조 및 수정, 중간자 공격, NDEF Signature RTD 취약점 등 물리적·논리적 보안 위협들이 존재한다는 연구를 밝혔다[5].

특히 NFC 태그 중 전 세계적으로 가장 많이 사용되고 있는 Mifare Classic에서 무작위 생성 함수의 낮은 엔트로피, 키 생성 함수의 입력 비트 등의 취약점과 더미 리더기 공격, 스니핑 공격, 향상된 Dark Side Attack 등의 공격 기법에 관한 연구가 진행되었다[4].

또한 현재 대부분의 사내 출입 통제에 사용되는 RFID 카드 사용의 위험성에 관한 연구와 스마트폰의 NFC 기능을 이용한 출입 통제 시스템 설계 및 구현에 관한 연구가 제시되었다[6].

본 논문에서는 일상생활에 널리 쓰이고 있는 RFID 기반 출입 통제 시스템에서 사용하는 Mifare Classic 1K 태그 타입의 취약점을 집중 조명하여 그 위험성을 각인시킨다. 또한 대응방안을 제시하여 Mifare Classic 1K 태그 타입을 사용하는 RFID 출입 통제 시스템 환경에 맞는 대응방안을 제시하여 NFC 기술 활용에서의 보안성을 향상시키고자 한다.

III. Mifare Classic 1K 개요

본 장에서는 NFC 태그 타입의 일종인 Mifare Classic 1K의 개요와 취약점에 대해 설명한다.

3.1 Mifare Classic 1K 개요

Mifare Classic 1K(1,024byte)는 1994년에 개발된 NFC 태그 타입의 일종으로, Mifare Classic 계열 중 최초로 출시된 태그 타입이다. Mifare Classic의 통신 프로토콜은 ISO/IEC 14443 Type A를 따르며 암호화 알고리즘인 Crypto 1을 적용한 인증 및 암호화 방식이 적용되고 있다[4].

3.2. Mifare Classic 1K 취약점

본 절에서는 Mifare Classic 1K 시스템에서 발생할 수 있는 취약점들을 소개한다.

Mifare Classic NFC 환경에서 Default Key Search Attack, Successful Authentication Data Attack 등의 취약점들로 인해 Mifare Classic 태그의 데이터를 복호화 하여 NFC 태그를 복제할 수 있다[4].

3.2.1. Default Key Search Attack

Mifare Classic 칩 제조사는 NFC 태그를 제조할 때 모든 섹터(Sector)의 Key 데이터를 Default Key로 설정하여 판매한다. 일반적으로 사용되는 Default Key는 아래 [표 1]과 같다[4].

[표 1] Mifare Classic Default Key

0xFFFFFFFFFFFF	0xA0B0C0D0E0F0
0xA1B1C1D1E1F1	0xA0A1A2A3A4A5

하지만 서비스를 제공하는 상당수의 회사는 Default Key를 변경하지 않고 그대로 사용하는데, 이 경우 Default Key Search Attack을 수행할 수 있다[6].

이에 따라 Mifare Classic 칩을 사용하는 제품의 제조사는, 고객사에게 기본으로 설정된 Default Key를 자신의 시스템에 맞게 변경하도록 권고하고 있다.

3.2.2. Successful Authentication Data Attack

성공적으로 등록된 Mifare Classic 타입의 카드에는 UID, 인증데이터 등 섹터 키를 복구할 수 있는 정보가 포함되어 있다. 섹터 키가 다른 섹터의 인증 데이터에서 유도 되는 것을 이용하여 카드를 읽는 것만으로 섹터 키를 복구할 수 있다[8].

IV. 출입 통제 시스템에서 발생하는 Mifare Classic 취약점 분석 절차

본 장에서는 실제 상용화된 출입 통제 시스템에서 Mifare Classic 태그의 UID만을 등록하여 이를 사용자 인증 수단으로 사용하는 점과, 3.2.2절에서 밝힌 등록된 인증 데이터가 암호화되지 않고 노출되는 취약점을 이용하여, 출입 통제 시스템에 등록된 키 태그의 복제가 일어날 수 있음을 실험을 통해 증명한다.

본 연구에서는 Mifare Classic 1K 태그 타입으로 사용하는 도어록 5종, S사의 출입 통제 시스템 1종, 총 6종을 대상으로 실험하였고, UID만을 사용자 인증수단으로 사용하는 모든 제품에서 키 태그 복제가 가능한 것을 확인하였다.

4.1 UID를 통한 사용자 인증

[그림 1] NFC 태그 등록 전/후 섹터 데이터 일부

Mifare Classic Tool을 사용하여 출입 통제 시스템에 등록 전과 등록 후의 Mifare Classic 1K 태그 섹터 데이터를 비교한다.

[그림 1]과 같이 등록 전과 후, 모든 섹터의 데이터에 변화가 없음을 확인할 수 있다. 또한 섹터 0번 외의 섹터에 임의의 값을 작성하여 출입 통제 시스템에 등록하고 인증을 시도하였을 때 정상적인 사용자로 인식한다는 점을 파악하였다.

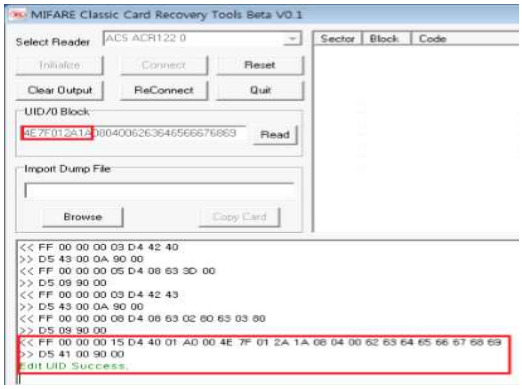
이를 통해 Mifare Classic 1K 기반 출입 통제 시스템에서 섹터 0번의 UID와, UID의 검증값인 BCC(Bit Count Check)를 등록하고 이를 사용자 인증 수단으로 이용한다는 것을 확인할 수 있다.

4.2 NFC 태그 복제

ACR122U NFC Reader/Writer 프로그램과 UID 변조가 가능한 카드를 이용하여 출입 통제 시스템에 등록된 UID와 BCC를 복제한 뒤, 출입 통제 시스템에 정상적인 사용자로 인증될 수 있음을 증명한다.

[그림 2] UID 변조가 가능한 카드와 카드의 데이터 일부

[그림 2]의 왼쪽은 UID 변조가 가능한 Mifare Classic 타입의 NFC 카드이며 오른쪽은 해당 카드의 섹터 0번의 데이터이다. 해당 카드의 UID는 "7C D9 23 01"이며, BCC는 "87"임을 알 수 있다.



[그림 3] ACR122U NFC Reader/Writer를 통한 UID 변조

[그림 3]은 UID Changeable 카드에 [그림 1]에서 확인한 UID인 "4E 7F 01 2A"와, UID 검증 값인 BCC "87"을 수정, 삽입한 결과이다. 이로 인해 정상적으로 UID가 변조된 것을 확인할 수 있다.

이후 변조한 UID Changeable 카드를 통해 출입 통제 시스템에 사용자 인증을 시도했을 때 정상적인 사용자로 인증됨을 확인할 수 있었다.

더 나아가 출입 통제 시스템의 펌웨어를 분석하여 Mifare Classic 태그에 등록된 UID와 BCC를 유추하고, 이를 통해 복제된 NFC 태그를 생성한다면 출입 통제 시스템에 등록된 NFC 태그의 데이터를 읽지 않은 상태에서 비인가 사용자의 접근을 허용할 수 있다.

V. 대응방안

4장에서 밝힌 출입 통제 시스템의 Mifare Classic 태그 타입 취약점에 대한 대응방안을 하드웨어와 소프트웨어 두 가지 관점에서 제시한다.

5.1 하드웨어 관점에서의 대응방안

Mifare Classic 1K 태그 타입은 NFC 통신 기술을 이용한 태그 타입 중 가장 많은 취약점이 존재하는 것으로 밝혀진 태그 타입이다[7]. 3장에서 설명한 바와 같이, 해당 취약점을 이용한 공격을 통해 Mifare Classic 태그 타입이 모든 섹터의 기본 키를 사용하지 않더라도 실현 가능한 시간 이내의 모든 키 값을 획득할 수 있다. 이를 통하여 데이터 위/변조 및 복제 공격이 가능하다. 이러한 이유로 Mifare Classic 1K의 데이터에 5.2절과 같은 인증 방식을 추가한다고 하더라도, 해당 태그 타입은 근본적으로 취약한 방식의 태그 타입이므로 사용을 지양하고 128bit의 암호화 방식을 지원하는 Mifare Plus 2K/4K 이상의 태그 타입 버전을 사용하는 것을 권고한다.

5.2 소프트웨어 관점에서의 대응방안

```
nfc_card = nfc_readcard()

nfc_original_UID = getUID(nfc_card)
nfc_original_BCC = getUID(nfc_card)
nfc_original_Manufacturer_Number = getUID(nfc_card)

data_string = "ABCDEFGHICJ1234512345123"

nfc_writecard(0,0,data_string)

nfc_renewal_UID = getUID(nfc_card)

if(nfc_original_UID != nfc_renewal_UID){
    exit();
    }
```

[그림 4] NFC 태그의 UID Writable여부를 확인하는 소스 코드

출입 통제 시스템의 취약한 태그 타입인 Mifare Classic 1K를 이용하기 위해선 사용자 인증 시 펌웨어의 NFC 태그의 섹터 0번에 [그림 4]와 같이 UID와 BCC에 Writable 여부를 확인하는 로직을 추가해야 한다. 소스 코드 로직는 [그림 4]와 같다. 이를 통해 UID 변조를 확인하여 비인가 사용자의 접근을 막을 수 있다.

VI. 결론

본 논문에서는 Mifare Classic 1K에서 발생하는 취약점을 이용해 실제로 태그를 복제할 수 있다는 점을 연구를 통해 확인하였다. 이에 대한 대응방안으로는 취약한 태그 타입인 Mifare Classic 1K보다 높은 버전인 Mifare Plus 2K/4K를 적용하는 방법과 펌웨어에서 해당 Mifare Classic Tag에 UID와 BCC가 Write가 가능할 시 복제된 Mifare Classic 태그로 감지하는 로직을 추가하는 방법을 제안하였다.

이러한 대응방안이 실제 제품에 적용된다면 출입 통제 시스템의 보안성을 높여 출입 통제 시스템을 보다 안전하게 할 수 있을 것으로 기대한다.

[참고문헌]

[1] 김종대, "NFC, 모바일 서비스의 한계를 넘는다",LG Business Insight, pp. 14, April, 2011

[2] 노용래, 이명진, 문태준, 강기환, "코리아 NFC표준화포럼포럼운영", 한국정보통신기술협회, pp. 43, December, 2016

[3] 강정호, 김형주, 이재식, 박재표, 전문석, "13.56Mhz RFID 환경에서 안전한 보안 스킴 구축을 위한 암호 스킴 및 보안 프로토콜 연구", 한국산학기술학회논문지(한국산학기술학회), pp. 1393-1401, March, 2013

[4] 이예림, "MIFARE classic 출입 통제 카드에 대한 효율적인 공격기법 연구", 국민대학교 석사학위논문, pp. 139, October, 2014

[5] 김미선, 김현곤, 이범기, 서재현, "NFC 보안 기술 및 보안 취약점에 관한 연구", 스마트미디어저널 제2권 제3호, pp. 54~61, March, 2013

[6] 김덕경, 양환석, 이병천, "NFC를 활용한 출입 통제 시스템", 한국정보보호학회 동계 학술대회, pp. 4, 2014

[7] Yi-Hao, ChiuWei-Chih Hong, Li-Ping Chou, Jintai Ding, Bo-Yin Yang, "A practical attack on patched MIFARE Classic", Lecture Notes in Computer Science Springer vol 8567, pp. 150~164, August, 2008

[8] 안현진, 이예림, 이수진, 한동국, "실제 환경에 최적화된 MIFARE Classic 공격", 전기학회논문지, pp. 2240~2250, December, 2016

FIDO 기반의 웹 표준 인증을 위한 웹 브라우저 안전 저장소 시스템 설계 및 구현

박재연[1], 신동일[1], 신동규[1], 이재영[2], 이형석[2]

[1]세종대학교 컴퓨터공학과, [2]㈜아이리플

Design and implementation of web browser secure repository for web standard authentication based on FIDO

Jaeyeon Park[1], Dongil Shin[1], Dongkyoo Shin[1], Jaeyoung Lee[2], Hyungseok Lee[2]

[1]Dept. of Computer Engineering, Sejong University., [2]iRIPPLE, Inc.

요 약

최근 공인인증제도 폐지 등 정부의 정책 방향에 따라 웹 표준을 준수하는 인증 수단에 대한 연구가 활발해졌다. 기존의 웹 저장소에 저장된 데이터는 변조가 가능하여 취약점이 발생할 수 있다는 문제점이 존재했다. 본 논문에서는 FIDO(Fast IDentity Online) 표준을 따르는 웹 브라우저에서 인증 취약점을 해결하기 위하여 상호 검증 기법과 API(Application Programming Interface) 위/변조 차단 및 난독화를 적용하였다. 또한 별도의 프로그램 설치를 요구하지 않는 No-Plugin을 구현함으로써 사용자의 편의성도 향상된다. 이를 통하여 안전한 웹 저장소를 개발하고 공인인증서 제도의 폐지에 따른 대체방안으로서 유용하게 사용될 것으로 예상한다.

I. 서론

기존의 보안 형태인 ActiveX/Plungin 퇴출로 나타난 EXE형태의 실행모듈이 보안 취약점뿐만 아니라 낮은 사용자 편의성을 갖는 문제점을 보인다. 뿐만 아니라 최근 정부의 공인인증 제도 폐지 정책으로 웹 표준을 준수하는 새로운 형태의 인증 수단에 대한 연구가 활발히 이루어지고 있다.

본 연구는 웹 브라우저에 저장되는 데이터 유출 및 재사용으로 인한 문제점 해결과 웹 표준을 준수하는 새로운 보안 정책의 필요성 확대에 따라 기존의 방법들을 대체할 수 있는 기술을 제시하기 위한 연구 내용을 소개한다. 아래의 표 1은 2010 년부터 2017 년까지의 OWASP(The Open Web Application Security Project) TOP10에서 웹 브라우저에 저장되는 각종 데이터 보호에 대한 중요성을 지속적으로 강조하고 있는 현황이다 [1].

표 1 OWASP Top10 현황

OWASP Top10-2010	A2-Cross-Site Scripting(XSS)
OWASP Top10-2013	A3-Cross-Site Scripting(XSS)
OWASP Top10-2017	A7-Cross-Site Scripting(XSS)

별도의 프로그램 설치가 전혀 필요 없는 'No-Plugin'을 정책 목표로 하며 웹 데이터의 탈취 및 재사용을 차단하는 기술을 통하여 차별화한다.

본 논문에서는 FIDO 기반의 웹 브라우저 안전 저장소 개발을 위하여 클라이언트 모듈에 WebCrypto를 적용하였다. 또한 API Level 호출자와 피 호출자 간의 상호 인증을 통하여 무결성을 검증하며, 웹 API에 대한 위/변조 차단 및 난독화를 적용하였다.

II. 관련 연구

키를 생성하는 다양한 방법이 있지만 각각의

방법들이 지원하는 기능도 다르며 동일 방법이라도 버전에 따라 구조적인 측면에서 차이가 있기에 선택에 따라 구현하는데 차이가 생긴다. <표 1>에 키를 생성하는 방법에 대한 표준 기술 현황에 대하여 정리하였다.

표 1. 키 생성 알고리즘 현황

구분	내용
PKCS8 [2]	암호화된 개인키를 저장하는 방법을 정의 공개키로 데이터을 암호화 가)RSA 키의 인코딩 ① X.509를 사용하여 인코딩 사용(공개 키와 인증서를 전달할 때 사용하는 공개 형식) ② 개인 키는 PKCS#8로 인코딩 ③ 공개/개인 키든 getEncoded()를 호출하면 자동으로 인코딩 됨 나)RSA 키의 디코딩 ① 인코딩된 키의 스펙 객체를 생성 ② KeyFactory를 사용하여 적절한 키 타입의 인스턴스를 활성화 ③ 공개 키 디코딩
p7b/pkcs7 [3]	PKCS # 7 또는 P7B 형식은 Base64 ASCII 형식으로 저장되며 파일 확장자는 .p7b 또는 .p7c. P7B 파일에는 개인키가 아닌 인증서와 체인 인증서 (중개 CA) 만 포함.
PFX/P12/ PKCS#12 [4]	PKCS # 12 또는 PFX / P12 형식은 서버 인증서, 중간 인증서 및 개인키를 하나의 암호화 가능 파일에 저장하기 위한 이진 형식임. 이러한 파일에는 대개 .pfx 및 .p12와 같은 확장자가 존재. 일반적으로 개인 컴퓨터에서 인증서 및 개인키를 가져오고 내보내는 데 사용.
CMS signed CMS [5], Enveloped CMS [6]	CMS / PKCS # 7은 메시지 서명 및 봉인 기능을 제공. 전자 서명 및 전자 메일 첨부는 SignedCms 메시지 및 EnvelopedCms 메시지 항목에서 별도로 설명한 모든 보안 서비스를 적용하여 보호.
PKCS5 [7]	개인키의 패스워드를 비밀키로 암호화하는 방법에 대한 정의. 비밀키를 생성하는 알고리즘과 비밀키 패스워드를 생성하는 방법에 대한 정의. PKCS8과 함께 개인키와 공개키의 개인키 생성방법과 저장방법에 대한 정의를 기술하고 있다.

또한 브라우저별로 지원하는 Webcrypto방법이 상이하다. 따라서 인증 시스템이 브라우저에 따라 각기 다른 방법으로 작동할 수 있도록 구현한다. <표 2>에서는 브라우저별 지원하는 알고리즘의 현황이다.

표 2. 브라우저별 지원하는 Webcrypto 알고리즘 현황

	Webcryto (PBKDF2)	Webcryto (AES-CVBC)	Webcryto (RSA PKCS1)	CrytoJS (PBKDF2)
IE 11	X	O	O	O
IE Edge	X	O	O	O
Chrome	O	O	O	-
Firefox	O	O	O	-
Opera	O	O	O	-
Safari (sierra)	O	O	O	-

2.1 Replay Attack

Replay Attack은 Playback Attack이라도 불리며 유효한 데이터 전송이 악의적으로 반복되거나 지연되는 네트워크 공격의 한 형태인데 오랫동안 논의되어 왔지만 완벽하게 차단되지 못하고 있다 [8]. 정상적으로 의도되었던 메시지를 공격자가 가로채어 변조된 메시지로 재사용함으로써 원본 메시지의 수신자가 성공적으로 완료했다고 착각하게 하는 방식으로 수행된다 [9].

2.2 FIDO 2.0 기술

FIDO client모듈이 웹 브라우저에 내장 될 예정이기에 브라우저 기반의 인증이 점차 확대될 전망이다. 해외에서는 이미 Web Secure Storage에 대한 논의가 활발히 진행되고 있는 상태이다 [10]. 이에 따라 해당 인증정보에 대한 새로운 보안 정책 역시 중요성이 높아지고 있다. FIDO란 Fast Idntity Online의 줄임말로서 비밀번호 방식뿐만 아니라 지문, 홍재, 얼굴 인식 등의 생체 정보를 활용하여 인증시스템으로 사용하는 기술표준이다. 시스템 특징으로는 인증 프로토콜과 인증 수단을 분리함으로써 보안과 편리성을 확보한다. 일반적으로 인증 프로토콜은 2가지로 제안되었다 [11].

2.2.1 UAF 프로토콜

첫 번째 모델은 UAF로 Universal Authentication Framework의 줄임말이다. 사용자의 기기에서 제공하는 인증방법을 기반으로 온라인 서비스와 연동하여 권한을 인증하는 기술이다. 대표적으로 지문인식을 통하여 결제 서비스를 진행하는 예시가 있다. (그림 1)에서는 UAF 프로토콜의 상세한 작동 모델이다. (그림

2)에서는 UAF 프로토콜 모델 중에 Authenticator를 구성하는 요소들을 보여준다.

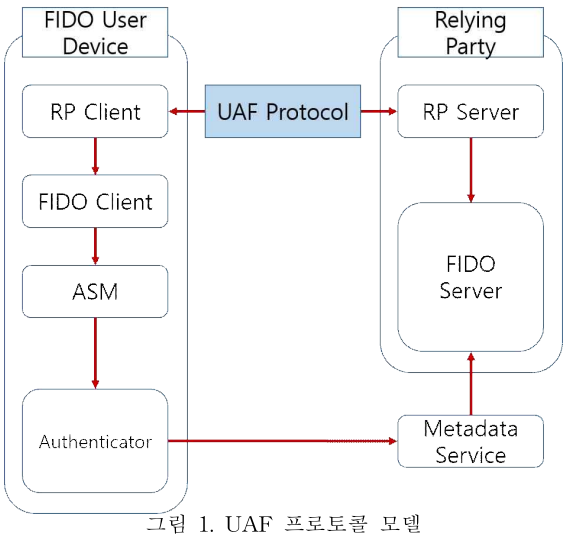


그림 1. UAF 프로토콜 모델

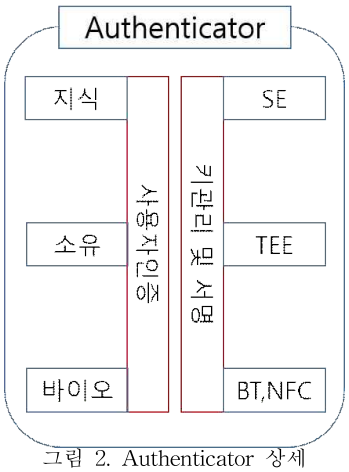


그림 2. Authenticator 상세

2.2.2 U2F 프로토콜

두 번째로 모델은 U2F로 Universal 2nd Factor의 줄임말이다. 기존의 아이디와 비밀번호 기반 인증서비스에서 추가적인 사용자 인증을 요구하는 방법이다. USB 보안키를 이용한 2차 인증에 사용하는 것이 대표적인 예이다.

III.웹 브라우저 안전 저장소 시스템 설계 및 구현

본 연구에서는 안전한 웹 저장소를 활용하여 FIDO 기반의 웹 표준 인증 시스템을 설계 구현하였다.

클라이언트 부분에서는 Webcrypto기법을 사용하여 키 쌍을 생성하고 저장하는 기능을 구현하였다. 전자서명 모듈과 암/복호화 모듈을 개발하였으며, 난독화 알고리즘을 개발하였다. 위/변조를 방지하기 위하여 API 호출자와 호출된 데이터 간에 상호 검증을 진행하도록 알고리즘을 작성하였다. (그림 3)은 클라이언트가 작동하는 구조를 보여준다.

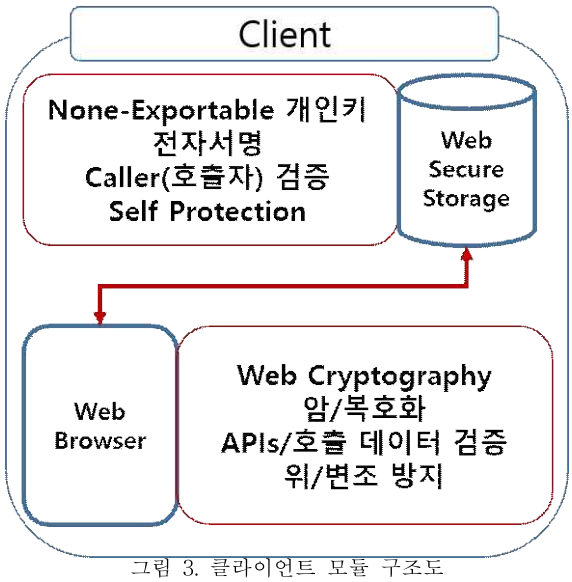


그림 3. 클라이언트 모듈 구조도

- Caller/Callee 상호 검증
  - -Javascript API와 데이터간의 상호검증
  - -자기제어를 통한 동적데이터 생성 및 보호
- 위/변조 차단
  - -Javascript 함수/변수의 위변조 차단 기능
  - -Overriding을 통한 재정의 방지
  - -Debugger keyword를 통한 디버깅 방지
- Web Cryptography
  - -W3C에서 웹 표준화 작업 진행 중
  - -키 쌍 생성 시 추출 불가능한 개인키 생성
  - -전자 서명, 암/복호화, 해시 지원
  - -전체 웹 브라우저의 92%이상 지원

서버 부분에서는 서명에 대하여 검증작업을 수행하는 모듈과 암/복호화 모듈을 적용하였다.

- 상호 검증을 위한 동적 Hash 데이터를 생성하는 모델을 구현하였다. 동적 Hash 데이터를 생성함으로써 매 상호 검증마다 새로운 데이터를 사용하고, 클라이언트에서 데이터

가 유출되더라도 재사용을 원천적으로 봉쇄하는 역할을 담당한다.

- 인증 세션을 보호하기 위하여 Replay Attack 방어기능도 포함된다. (그림 4)는 서버가 작동하는 구조를 보여준다.

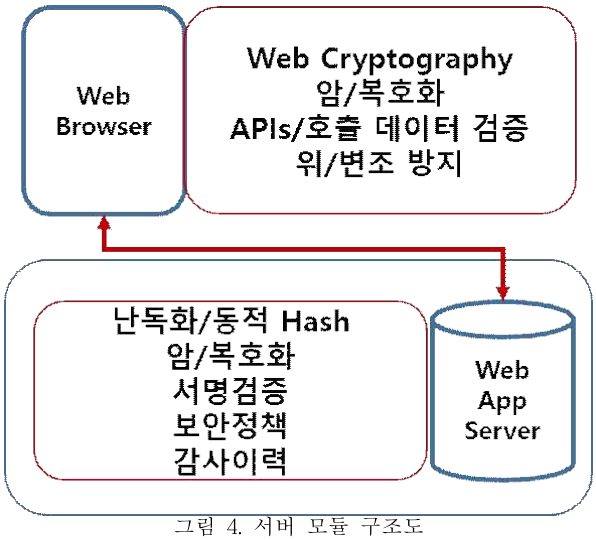


그림 4. 서버 모듈 구조도

IV.결론

본 논문에서는 FIDO 기술에 적용할 수 있는 웹 브라우저 안전 저장소를 구축하여 웹 표준 인증을 수행하는 시스템 연구를 진행하였다.

기존의 설치프로그램 형태를 벗어남으로써 보안 취약점을 해결할 수 있을 뿐만 아니라 사용자 편의성도 확보 하였다. 또한 난독화와 동적 Hash를 통하여 유출된 정보에 대한 해석을 불가능하게 함으로써 OWASP Top 10에서 지속적으로 언급되었던 유출 정보 재사용 문제의 해결에 도움을 준다.

인증 정보를 서버에만 저장하지 않고, 클라이언트에 분산 저장하는 방법을 통하여 웹 보안성을 향상하고 사용자 편의성을 확보하며 공인인증서 제도의 폐지에 따른 대체 방안으로서 유용하게 사용될 것으로 예상된다.

[ACKNOWLEDGMENT]

본 연구는 2018년도 중소벤처기업부의 기술개발사업 지원에 의한 연구임을 밝힙니다. [S2593659]

[참고문헌]

[1] Owasp.org , 2018. [Online]. Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

[2] B. Kaliski, "Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2," RFC Editor, May 2008.

[3] B. Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5," RFC Editor, Mar. 1998.

[4] M. Nystrom, S. Parkinson, A. Rusch, and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1," RFC Editor, Jul. 2014.

[5] R. Housley, "Cryptographic Message Syntax," RFC Editor, Jun. 1999.

[6] R. Housley, "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type," RFC Editor, Nov. 2007.

[7] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0," RFC Editor, Sep. 2000.

[8] P. Syverson, "A taxonomy of replay attacks [cryptographic protocols]," in Proceedings The Computer Security Foundations Workshop VII.

[9] S. Malladi, J. Alves-Foss and R. B, "On Preventing Replay Attacks on Security Protocols", Dtic.mil , 2002. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a462295.pdf.

[10] M. Jemel and A. Serhrouchni, "Security assurance of local data stored by HTML5 web application," in 2014 10th International Conference on Information Assurance and Security, 2014.

[11] Hwa-Gun. Cho, and Hae-Sool. Yang, "FIDO 생체기술과 안전영역을 연계한 공인인증서 효율화 방법," 디지털융복합연구, vol. 15, no. 8, pp. 183 - 193, Aug. 2017.

함수암호를 이용한 차세대 인증 Template 보호 기술*

박동희, 박영호

세종사이버대학교 정보보호대학원

Biometric Template Protection Using Function Encryption

Dong Hee Park, Young-Ho Park

Graduate School of Information Security, Sejong Cyber University.

요 약

최근 스마트폰 등 다양한 디바이스에서 생체정보를 이용한 차세대 인증기법이 증가하고 있다. 하지만 생체정보는 정보 유출시 패스워드와 달리 변경 등이 불가능하여 무엇보다 생체정보의 안전한 보호가 필수적이다. 본 논문에서는 함수암호를 이용하여 생체정보를 암호화한 상태에서 평문을 노출하지 않고 인증 할 수 있는 기법을 제안한다.

I. 서론

오늘날 클라우드 환경이 보편화되고 IoT 기기의 보급과 전자상거래가 활성화 되면서 기기간의 연결 또는 개인과 기기의 연결에 있어서 사용자 인증이 중요해지고 있다. 최근에 스마트폰 등 다양한 디바이스에서 지문, 홍채, 얼굴 등 생체 데이터를 활용하여 인증하는 방법이 활발하게 사용되고 있다. [1]

생물학적 특징은 지문, 얼굴, 장문, 손 모양, 홍채, 정맥 등에 해당하며 행동적 특징은 서명, 음성, 키보드 입력, 걸음걸이 등이 있다. 이러한 사용자의 생물학적, 행동적인 특성을 이용하여 사용자의 편리함과 안전성을 유지하며 사용자의 자각을 최소화하는 차세대 인증연구가 활발하게 이루어지고 있다. 하지만 생체정보의 누출로 인해 야기되는 피해는 패스워드의 누출로 인해 발생하는 피해보다 더 크다고 할 수 있는데, 이는 사용자의 생체정보가 일생 동안 변경되지 않기에, 노출된 생체정보를 삭제하고 재발급하기 어렵다는 데 기인하다.[2]

본 논문 2장에서는 생체정보를 인증하기 위한 Fuzzy 인증과 Distance measure 방식에 대하여 소개하고 3장에서 안정성을 보장할 함수암호를 소개한다. 4장에서는 함수암호를 이용한 차세대 인증 Template 방법을 제안한다. 5장에서는 결론 및 향후 연구 과제에 대해 논의한다.

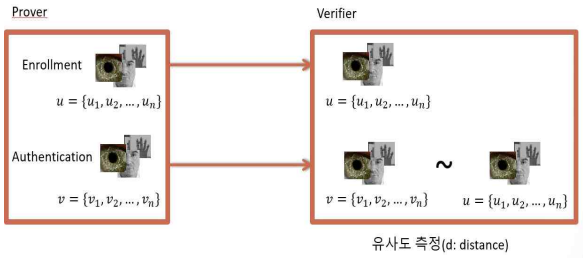
II. 생체 인증 정보

2.1 Fuzzy 인증

생체정보를 이용한 인증은 Noise가 존재하며 인가자를 비인가자로 판단하는 본인거부 에러(FRR)와 비인가자를 인가자로 판단하는 오인식

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00380, 차세대 인증 기술 개발)

에러(FAR)가 발생한다.[3] 따라서 등록된 생체 정보와 인증 받을 정보 간의 차이가 발생한다. Fuzzy 인증 기법은 Distance measure를 사용하여 Template 정보와 인증 대상 정보와의 유사도 분석결과를 도출한다. 이때 유사도 결과는 일치, 거의 일치, 불일치로 등으로 표현되며 거리 함수를 이용하여 판단한다.[4]



[그림 1] Fuzzy 인증

2.2 Distance measure

거리함수는 크게 유클리디언 거리함수와 비유클리디언 거리함수로 구분된다. 유클리디언 거리함수에는 대표적으로 L_2 norm distance, L_∞ norm distance 그리고 L_1 norm distance가 있다. 비 유클리디언 거리함수는 집합론에 근거한 Jaccard distance, Cosine 함수에 근거한 Cosine distance, 곱집합 위에서 정의되는 Hamming distance가 있다. Hamming distance는 서로 다른 비트의 개수를 출력하는 방식이다. Hamming distance는 $x, y \in \{0,1\}$ 일 때 데이터를 1 비트로 표현 하면 $HD(x,y) = y(1-2x)+x$ 으로 표현할 수 있다.

X	Y	H
1	1	0
0	1	1
1	0	1
0	0	0

[표 1] Hamming distance 진리표

n 비트의 데이터 $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$의 거리는 $HD(X,Y) = \sum\{y_i(1-2x_i)+x_i\}$ 으로 표현 할 수 있으며 내적으로 표현하면 다음과 같이 표현할 수 있다.

$$HD(X,Y) = <1-2X, Y> + \sum x_i$$

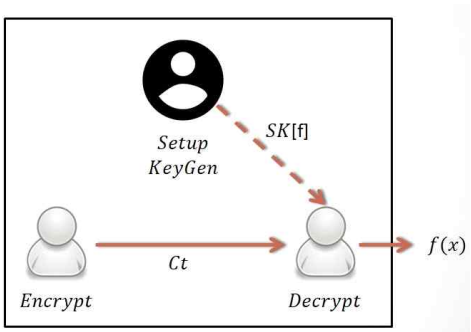
또한 조금 더 효율적인 거리함수를 다음과 같이 얻을 수 있다. 주어진 벡터 $X, Y \in \{0,1\}^n$를 한 비트가 0일 경우 -1로 1일 경우 1로 변환한 새로운 벡터 $X', Y' \in \{-1,1\}^n$를 얻는다. 즉 $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$이면 $X' = (2x_1 - 1, \ldots, 2x_n - 1)$ $Y' = (2y_1 - 1, \ldots, 2y_n - 1)$으로 변환한다. 따라서 주어진 벡터 X,Y의 거리를 새로운 벡터 X', Y'를 사용하여 다음과 같이 새롭게 정의 할 수 있다.[5]

$$HD(X,Y) = \{n - <X', Y'>\}/2$$

본 논문에서는 위와 같은 효율적인 거리함수를 사용하여 인증기법을 설계하였다.

Ⅲ. 함수암호

공개키 암호는 평문을 공개키를 이용하여 암호화하고 개인키로 암호문을 복호화하여 평문을 얻는다. 하지만 함수암호는 평문을 공개키로 암호화하고 주어진 함수와 관련된 비밀키를 사용하여 암호문을 평문으로 복호화하지 않고 함수값 만을 얻을 수 있다. 따라서 함수암호를 이용하면 인증정보를 암호화한 상태에서 인증을 수행할 수 있다. 함수암호는 다음 4가지 알고리즘으로 구성되어 있다.[6]



[그림 2]. 함수암호

● Setup(1^λ): 보안상수 1^λ 를 입력으로 하여 마스터 공개키 mpk와 마스터 비밀키 msk를 생성한다.

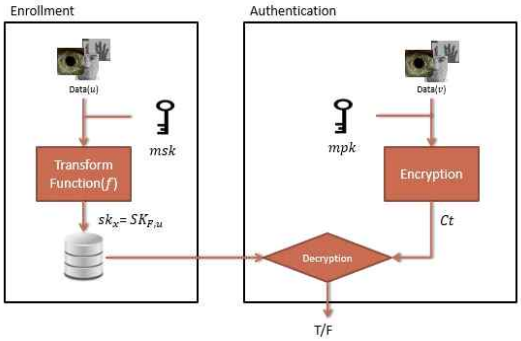
● Keygen(msk,x): msk와 x를 이용하여 함수

f 에 근거한 비밀키 sk[f]를 생성한다.

- Encrypt(mpk,y): mpk를 이용하여 y를 암호문 Ct로 암호화 한다.
- Decrypt(sk[f],Ct): Ct로부터 비밀키 sk[f]를 이용하여 f(x)를 얻는다.

Ⅳ. 함수암호를 이용한 인증기법 설계

본 장에서는 앞에서 정의한 거리함수와 함수암호를 이용한 새로운 인증기법을 제안한다.



[그림 3] 함수 암호를 이용한 인증기법

4.1 내적함수를 이용한 인증기법

제안할 인증기법은 Decisional Diffie-Hellman(DDH) 가정에 근거하여 내적함수로 설계하였다. 군 G는 위수가 소수 p인 생성원 g로 생성된 순환군이라 하자. DDH가정은 (g^a, g^b, g^{ab})이 주어 졌을 때 (g^a, g^b, g^c)를 계산적으로 구분할 수 없다는 것을 가정한다. 제안한 인증 기법은 DDH가정에 근거한 설계로 선택적 평문공격에 안전하다.[7]

4.1.1 사전단계

제안 인증 시스템에 사용될 파라미터와 군 G, 소수 p, 생성원 g를 설정한다. 또한 시스템의 마스터 공개키 mpk와 마스터 비밀키 msk를 생성한다.

- 1^λ: 보안 상수
- l : 암호문의 벡터의 길이
- GroupGen(1^λ):(G,p,g) : $G=<g> = \{g^1,...,g^p\}$
- $Setup(1^\lambda, 1^l) \rightarrow mpk = (h_1,...,h_l) = (h_i = g^{s_i})$
  $msk = s = (s_1,...,s_l) \in Z_p^l$

4.1.2 등록단계

사용자의 생체정보 u를 Hamming distance 거리함수에 적용하기 위해 x으로 치환하여 다음과 같이 등록한 후 거리함수에 근거한 비밀키 sk_f를 생성한다.

$$x = 2u - 1 = (2u_1 - 1, 2u_2 - 1, ..., 2u_l - 1)$$
$$u = (u_1, u_2, ..., u_l) \in Z_p^l$$

- $KeyGen(msk, f) \rightarrow sk_f = <x, s>$

4.1.3 인증단계

인증대상 데이터 v를 y으로 치환 후 Ct으로 암호화한다.

- $Encrypt(mpk, v) \rightarrow Ct = (ct_0, (ct_i)_{i \in [l]})$
  $r \leftarrow Z_p, ct_0 = g^r, ct_i = h_i^r g^{y_i}, i \in [l]$
  $y = 2v - 1 = (2v_1 - 1, 2v_2 - 1, ..., 2v_l - 1)$
  $v = (v_1, v_2, ..., v_l) \in Z_p^l$

Ct를 복호화하여 원하는 인증 비교값을 얻을 수 있는 $g^{<x,y>}$를 도출한다.

- $Decrypt(sk_f, Ct) \rightarrow \prod_{i \in [l]} ct_i^{x_i} / ct_0^{sk_{z_i}}$
  $= \prod_{i \in [l]} (g^{s_i r + y_i})^{x_i} / g^{r(\sum_{i \in [l]} x_i s_i)}$
  $= g^{\sum_{i \in [l]} x_i s_i r + \sum_{i \in [l]} y_i x_i - r \sum_{i \in [l]} x_i s_i}$
  $= g^{\sum_{i \in [l]} y_i x_i} = g^{<x,y>}$

4.1.4 인증 검증 단계

인증 단계에서 얻은 $g^{<x,y>}$값을 사용하여 검증 Table과 비교하여 검증한다. Decrypt 단계에서 얻은 $g^{<x,y>}$은 $HD(u,v) = \{n - <x,y>\}/2$ 으로 표현하면 $g^{<x,y>} = g^{n-2HD(u,v)}$이다. [표 2]검증 Table은 $HD(u,v)$ 값을 0을 시작으로 threshold 값 t 까지 적용시켜 미리 저장해 놓는다.

Value	$g^{<x,y>}$
0	g^n
1	g^{n-2}
2	g^{n-4}
...	...
t	g^{n-2t}

[표 2] 검증 Table

따라서 검증 결과 값이 검증 Table에 포함될 경우 True으로 반환하고 포함되지 않을 경우 False로 반환한다.

Ⅴ.결론

본 연구에서는 기존 인증과 달리 함수암호를 이용하여 생체정보를 암호화된 상태에서 복호화 하지 않고 인증을 수행할 수 있으며 인증 템플릿을 안전하게 보호할 수 있다. 또한 기존 RSA, DSA, ECC 등 기존 공개키 기반 암호의 도메인 파라미터를 적용할 수 있으며 기존 암호가 탑재된 환경에 효율적으로 적용 할 수 있을 것으로 기대된다.

제안된 인증 기법은 홍채와 같이 작은 템플릿 정보를 갖는 생체인증에 효율적이며 안면인식과 같은 큰 템플릿 정보를 갖는 경우 계산량이 많이 필요하다. 따라서 차세대 인증 데이터 u를 안전하게 보관하는 기법과 검증 Table의 용량 문제를 효율적으로 해결하는 연구가 추가로 필요하다.

[참고문헌]

[1] Heejin Park, Younho Lee , Survey on Privacy-preserving Techniques for Biometric Authentication, Journal of KIIT. Vol. 16, No. 4, pp. 109-122 , Apr, 2018.

[2] L.Lai, S.Ho and H.V Poor, Privacy-security trade-offs in biometric security systems-Part Ⅱ Multiple use case, IEEE Transactions on Information Forensics and Security, Vol. 6, No 1 , pp. 140-151 , Mar, 2011.

[3] Byungchul Cho and Jong-man Park, Technology Review on Multimodel Biometric Authentication, The Journal of Korean Institute of Communications and Information Sciences '15-01 Vol40 No.01 , Jan, 2015.

[4] Fuchun Guo, Willy Susilo and Yi Mu, Distance-Based Encryption: How to Embed Fuzziness in Biometric-Based Encryption, IEEE Transactions on Information Forensics and Security,VOL 11, NO.2 , Feb, 2016.

[5] Sam Kim, Kevin Lewi, Avradip Mandel, Hart Montgomery, Arnab Roy, and David J. Wu, Function-Hiding Inner Product Encryption is Practical, Security and Cryptography for Nerworks pp 544-562, Mar, 2002.

[6] Dan Boneh, Amit Sahai and Brent Waters, Functional Encryption: Definitions and Challenges, Public-Key Cryptography –PKC 2015 pp 733-751 , Jan, 2015.

[7] Michel Abdalla, Florian Bourse, Angelo De Caro and David Pointcheval, Simple Functional Encryption Schemes for Inner Products, Public-Key Cryptography – PKC 2015 pp 733-751 , Jan, 2015.

악성코드 패밀리명의 Ground-Truth를 위한 레이블링 기법 연구*

오상진*, 박래현*, 박준형**, 권태경*
* 연세대학교 정보대학원 정보보호 연구실
** 한국 인터넷 진흥원 (KISA)

A Study of Labeling for Ground-Truth of Malware Family Names

Sangjin Oh*, Laehyun Park*, Jun-hyung Park**, Taekyoung Kwon*
*Information Security Lab, GSI, Yonsei University.
** Korea Internet and Security Agency (KISA)

요 약

기계학습을 이용한 악성코드 분류 방법들이 많이 연구되고 있다. 기계학습에서의 ground truth는 지도학습에 사용되는 학습 데이터 분류의 정확도를 뜻한다. 안티바이러스 벤더에서 제공하는 악성코드 패밀리명은 ground truth가 낮으며, 악성코드 분석 전문가들이 이러한 기존 레이블링 기법을 기반으로 한 패밀리명에 의존하기 때문에 분류에 대한 신뢰성이 떨어지게 된다. 본 연구에서는 높은 ground truth를 가진 데이터셋을 구성하기 위해 통합된 악성코드 패밀리명 레이블링 기법을 제안한다. 본 실험에서는 2,241개 악성코드 샘플을 이용하였고, 3개의 클러스터링 알고리즘을 활용하여 레이블링의 정확도를 평가하였다. 실험 결과, 제안한 레이블링 기법이 Agglomerative Clustering에서 85.58%, K-meas에서 85.58%, Birch Clustering에서 82.10%로 기존 레이블링 기법과 같이 하나의 레이블만 이용하였을 때보다 높은 정확도 값을 보였다.

I. 서 론

악성코드 샘플은 과거에 비해 기하급수적으로 증가하고 있다. 이에 따라 악성코드 분석 전문가가 수동적으로 탐지하는 방법을 대체하여 기계학습을 이용한 악성코드 분류 방법들이 많이 연구되고 있다 [5][6][7]. 특히, 지도학습 기반의 알고리즘을 이용하기 위해서는 악성코드 샘플의 레이블명이 존재해야 한다. 이러한 레이블명은 안티바이러스 벤더에서 제공해준 악성코드 패밀리명을 사용하게 된다. 하지만, 악성코드 패밀리명은 안티바이러스 벤더마다 다르게 제공되며 하나의 악성코드 샘플에도 다양한 패밀리명이 부여되기 때문에 신뢰도가 낮다. 악성코드 분석 전문가들은 이러한 신뢰도가 낮은 패밀리명에 의존하여 레이블링 작업을 수행하기 때문에 이에 따른 문제가 발생한다[1].

본 연구는 보다 높은 ground truth를 위해 새로운 레이블링 기법을 제안한다. 본 연구를 위해서 안티바이러스 벤더 3사의 악성코드 패밀리명과 쿠쿠 샌드박스를 통해 추출한 악성코드 패밀리명을 이용한다. 각각의 악성코드 패밀리명을 파싱하는 과정을 거치고 파싱된 토큰 중 제네릭 토큰을 필터하는 과정을 거친다. 이러한 과정을 거쳐 정제된 토큰을 통합하는 새로운 레이블링 기법을 제안한다. 제안하는 레이블

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2016-0-00081, 악성코드 全 생명주기 통합 프로파일링 및 공격그룹 식별 기술 개발).

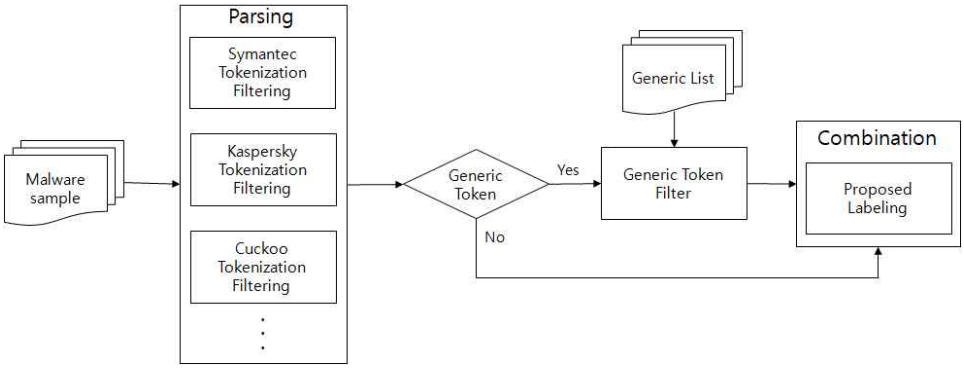


Fig 1. System Overview

링 기법을 평가하기 위해서 세 가지 클러스터링 알고리즘을 이용하여 제안하는 레이블과 각 벤더사가 제공하는 레이블의 클러스터링 결과를 비교한다.

II. 기반 기술 소개

2.1 Clustering Algorithm

기계학습은 크게 지도학습 기반과 비지도학습 기반의 알고리즘이 존재한다. 클러스터링 알고리즘은 비지도학습 기반에 속하며 이는 지도학습 기반의 방법과 다르게 레이블링이 되어 있지 않은 데이터를 병합할 때 사용된다.

Agglomerative Clustering. 계층 기반 알고리즘으로써 단일 클러스터로 병합할 때까지 계층을 상향하면서 클러스터 쌍을 연속적으로 병합한다.

K-Means. 거리 기반을 두는 알고리즘으로 k개의 중심점을 두고 묶인 클러스터의 거리 차이의 분산을 최소화하여 클러스터로 병합하는 알고리즘이다.

Birch Clustering. 대용량 데이터를 클러스터링하기 위해서 설계되었다. 이 알고리즘의 장점은 사전에 전체 데이터 세트가 필요하지 않으며, 유입되는 데이터 샘플에 대해 동적으로 클러스터링한다.

2.2 Feature Selection

기계학습에서 데이터를 분석할 때 데이터의 특징이 무수하게 많아질 경우 차원의 저주 문제가 발생한다. 이를 해결하기 위한 방법으로는 특징 선택(feature selection)이 있다. 특징 선택은 차원의 저주를 해결할 뿐만 아니라 훈련 시간을 감소시켜주며 과적합의 위험을 줄인다. 특징 선택 방법 중에 대표적이며 효율적인 방법인 순방향 선택(Forward Stepwise Selection) 알고리즘이 있다. 이 알고리즘은 모든 단일 특징에 대하여 정확도를 측정하고 이에 점차 특징을 추가시켜 최적의 특징 부분 집합을 찾는다[8].

III. 시스템 설계

본 연구에서는 ground truth가 높은 레이블을 만들기 위해 새로운 레이블링 기법을 제안한다.

본 연구는 신뢰성이 있는 레이블을 만들기 위해 Fig 1.과 같이 각 벤더사의 패밀리명 파싱, 제네릭 리스트를 이용한 제네릭 토큰 필터링, 토큰의 통합 순으로 단계적으로 수행된다.

파싱 단계에서는 각 벤더 3사(Symantec, AhnLab, Kaspersky)가 가지고 있는 악성코드 샘플의 패밀리명 규칙이 상이하기 때문에 벤더사에 맞추어 파싱을 한다. 파싱 하기 위해서는 '@, !, .'을 구분자로 지정하여 토큰을 분류하며, AhnLab의 V3의 경우 추가로 'Win32', 'Win-Trojan', 'Win-PUP', 'Win-95' 등 다음에 위치하는 문자열을 파싱하여 토큰화한다. 쿠쿠 샌드박스를 이용하여 패밀리명을 분류하는 방법은 VirusTotal과 쿠쿠 샌드박스를 연동하여 패밀리명을 지정하며, 이를 파싱하여 토큰화한다.

제네릭 토큰 필터 단계는 파싱 된 토큰에는 제네릭 토큰이 존재할 수 있기 때문에 이를 제거한다. 제네릭 토큰은 클러스터링을 할 때, 행위가 다른 악성코드 샘플끼리 클러스터링이 될 수 있는 위험이 있다. 이러한 위험을 제거하기 위해서 제네릭 리스트를

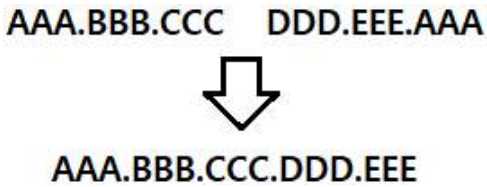


Fig 2. Combination of Label

생성하여 이를 통해 제네릭 토큰을 필터 한다.

토큰의 통합 단계는 정제된 토큰을 구분자 '.'으로 지정하여 병합을 진행한다. 하나의 레이블이 병합되었을 때 다른 벤더사의 토큰이 중복되면 하나의 토큰만 남기고 제거한다. 또한 Fig.2 와 같이 병합된 두 개의 레이블이 중복되는 토큰이 존재할 경우 하나의 레이블로 재병합하고 이를 반복한다.

제안된 기법으로 만들어진 레이블의 ground truth를 평가하기 위해 Agglomerative Clustering, K-means, Birch Clustering을 활용하며, 클러스터링 활용을 위해 쿠쿠 샌드박스를 통하여 8개의 특징 카테고리(command, compile, mutex, service, registry, droppedfile, network, file)을 추출한다.

IV. 실험 설계 및 결과

본 실험에서는 Intel (R) Xeon(R) Gold 6134 CPU @ 3.20 GHz, 190 GB RAM and Ubuntu 18.04.1 LTS 환경에서 Python 2.7 활용해서 실험을 수행했다. 또한, 실험을 위해서 2,241개의 악성코드 샘플을 사용하였다.

2,241개의 악성코드에 대해서 파싱 한 결과 Symantec은 55개, AhnLab은 77개, Kaspersky는 52개, 쿠쿠 샌드박스는 87개로 토큰화되었다. 제네릭 토큰을 없애기 위해 제네릭 리스트를 이용해 'heur', 'generic', 'trojan', 'trojan horse' 등 7개의 제네릭 토큰을 필터링하여 제거하였다. 정제된 토큰을 제안한 레이블링 기법으로 적용한 결과, 37개의 레이블로 병합되었다.

제안된 레이블과 기존의 각 벤더사의 레이블의 Ground truth의 비교를 위해 agglomerative clustering, K-means, Birch clustering 3개 알고리즘을 활용하였다.

클러스터링 알고리즘을 활용하여 나온 결과는 Table 1과 같으며, 이는 순방향 선택 알고리즘을 이용하여 8개의 특징 카테고리 중 최적의 특징을 선별하여 나타낸 신뢰도 값이다.

실험 결과, 제안한 레이블링 방법의 레이블 결과가 Agglomerative Clustering에서 85.58%, K-meas에서 85.58%, Birch Clustering에서 82.10%로 모든 알고리즘에서 기존 하나의 레이블만 이용했을 때보다 높은 결과 값을 보였다.

V. 관련 연구

기계학습 분야에서 'Ground truth'는 지도학습 기반에 사용될 학습 데이터 분류의 정확도를 뜻한다. 이에 따라 악성코드 분석의 영역은 높은 ground truth를 가진 데이터가 필수적이다. 데이터 셋의 ground truth를 높이기 위해서는 레이블링 작업이 필요하다. 이에 따라 최근에 여러 레이블링 생성 방법에 관련된 연구가 진행되고 있다.

Sebastian et al. [1]는 안티바이러스 엔진이 제공하는 레이블을 이용하여 악성코드 패밀리 이름을 추출하는 시스템인 'AVClass'를 개발하였다. AVClass는 VirusTotal을 통해 여러 안티바이러스 엔진의 레이블을 얻었으며, 이를 처리하기 위해 정규화, 일반 토큰 제거, 별칭 탐지 단계를 진행했다.

Kantchellian et al. [2]은 기존의 레이블링보다 정확하고 신뢰할 수 있는 ground truth를 도출하기 위해서 Antivirus 엔진에 가중치 할당하는 방법을 제안했다. 가중치 할당하기 위해 지도학습 기반과 비지도학습 기반의 알고리즘을 이용했다.

Hurier et al. [3]은 정량적인 방법으로 데이터 셋의 ground truth를 평가하는 새로운 측정 기준을 제안했다.

Mohaisen et al. [4]은 안티바이러스 벤더의 레이블링 시스템을 평가하기 위해서 ground truth를 측정하는 기준을 제안했다. 이 논문에서는 Completeness, Correctness, Consistency, Coverage 네 가지 기준을 제안하고 평가했다.

VI. 결 론

본 연구는 Ground truth가 높은 레이블을 만들기 위해서 새로운 레이블링 기법을 제안했다. 이를 통해 안티바이러스 벤더에서 제공하는 부정확한 패밀리로 인하여 많은 악성코드 분석 전문가와 연구자들의 잘못된 연구 진행과 결과를 방지할 수 있다. 추후 연구에서는 안티바이러스 벤더의 수를 확장해 제안한 새

Table 1. Clustering Algorithm Result

Algorithm	Vendor	optimal feature	Precision	Recall	F_1 Score
Agglomerative Clustering	**Our Label**	**mutex, service**	**91.66%**	**80.25%**	**85.58%**
	Symantec	mutex, service	85.17%	77.73%	81.28%
	AhnLab	mutex, service	82.37%	75.81%	78.95%
	Kaspersky	mutex, service	86.74%	81.29%	83.93%
	Cuckoo	mutex, service	81.68%	78.62%	80.12%
K-Means	**Our Label**	**mutex, service**	**91.66%**	**80.25%**	**85.58%**
	Symantec	mutex, service	86.22%	70.99%	77.87%
	AhnLab	mutex, service	82.51%	73.36%	77.67%
	Kaspersky	mutex, service	85.93%	74.41%	79.76%
	Cuckoo	mutex, service	80.13%	77.47%	78.78%
Birch Clustering	**Our Label**	**mutex, command**	**89.12%**	**76.10%**	**82.10%**
	Symantec	file, droppedfile, command, compile	79.08%	71.19%	74.93%
	AhnLab	mutex, service, command, droppedfile, compile, network	72.31%	74.37%	73.32%
	Kaspersky	file, droppedfile, command, compile, service	80.13%	72.54%	76.15%
	Cuckoo	droppedfile, compile	79.64%	76.02%	77.79%

로운 레이블링 기법에 다수의 패밀리명을 적용할 필요가 있으며, 본 연구에서 사용된 8개 특징 카테고리를 뿐만 아니라 최신 연구 동향에서 사용되는 보다 효율적인 특징 카테고리를 적용하면 보다 향상된 성능이 기대된다.

References

〔1〕 Sebastián, Marcos, et al. "Avclass: A tool for massive malware labeling." International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 230-253, Sep. 2016.

〔2〕 Kantchelian, Alex, et al. "Better malware ground truth: Techniques for weighting anti-virus vendor labels." Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security. pp. 45-56, Oct. 2015.

〔3〕 Hurier, Médéric, et al. "On the lack of consensus in anti-virus decisions: Metrics and insights on building ground truths of android malware." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment(DIMVA). pp.142-162, Jul. 2016.

〔4〕 Mohaisen, Aziz, and Omar Alrawi. "Av-meter: An evaluation of antivirus scans and labels." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment(DIMVA). pp 112-131, Jul. 2014.

〔5〕 Li, Jin, et al. "Significant Permission Identification for Machine Learning Based Android Malware Detection." IEEE Transactions on Industrial Informatics, pp.3216-3225, Jan. 2018.

〔6〕 Hu, Xin, et al. "MutantX-S: Scalable Malware Clustering Based on Static Features." USENIX Annual Technical Conference, pp 187-198, Jun. 2013.

〔7〕 Graziano, Mariano, et al. "Needles in a haystack: mining information from public dynamic analysis sandboxes for malware intelligence." USENIX Security Symposium. pp. 1057-1072, Aug. 2015.

〔8〕 M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto. "Novel feature extraction, selection and fusion for effective malware family classification," In Proc. Data and Application Security and Privacy (CODASPY), pp. 183-194, Mar. 2016

악성코드 위협 행위기반 엔드포인트 솔루션

이동주, 김인수 ,임강빈
순천향대학교 정보보호학과

Endpoint solution based on malware threat behavior

Dong-Ju Lee, In-Soo Kim, Kang-bin Yim
Department of Information Security Engineering, Soonchunhyang University.

요 약

악성코드는 컴퓨터가 발명된 이후부터 현재까지 지속적으로 발견되고 있으며, 기존 개인 PC 및 모바일 위주 환경에서 사물인터넷, AI 등 새로운 환경이 구축됨에 따라 유형도 다양해지고 진화하고 있다. 악성코드에 대한 피해영역은 사용자 개개인의 PC에서 국가적 주요 시스템으로 영역이 확장되고 있으며 피해규모도 커지고 있다. 또한 대응할 수 있는 시간이 지속적으로 짧아지고 있으며 신규 취약점을 이용하거나 제로데이 공격을 시도하는 악성코드들이 늘어나고 있어 그에 따른 대응은 어려운 것이 사실이다. 특히 국가 주요 시스템의 악성코드 감염은 국가 시스템 마비를 초래해 사회 혼란을 야기시킬 수 있다. 이에 우리는 악성코드가 주로 변경하는 DNS, 레지스트리 등의 값을 조회하여 악성코드를 탐지하고 치료하는 백신을 만들고자 하였다. 행위기반 백신은 사용자의 환경을 고려한 엔드포인트 기반이기 때문에 섬세한 작업이 요구되고 이후 세밀한 연구가 진행되어야 한다.

I. 서론

악성코드는 컴퓨터가 발명된 이후부터 현재까지 지속적으로 발견되고 있으며, 신규 악성코드의 발생빈도도 지속적으로 증가하고 있다. 또한 기존 개인 PC 및 모바일 위주 환경에서 사물인터넷, AI 등 새로운 환경이 구축됨에 따라 악성코드의 유형도 다양해지고 진화하고 있다. 이에 현재 상용화된 백신들은 악성코드를 탐지하기 위해 시그니처 기반과 행위 기반 탐지 기법을 모두 사용하고 있지만, 행위 기반 탐지 기법의 경우 시그니처 기반 기법보다 오탐율이 높기 때문에 주로 시그니처 기법을 사용하고 있다. 따라서 우리는 신규 및 변종 악성코드에 대해서도 대응이 가능한 행위 기반 기법을 사용해 백신 모델을 설계하였다.

II. 악성코드

2.1 악성코드의 정의

악성코드는 악성 소프트웨어(malicious software)의 줄임말로 하나의 컴퓨터, 서버 또는 네트워크상에 피해를 입히도록 설계된 모든 소프트웨어이다. 악성코드는 컴퓨터의 초창기부터 존재해왔으며 현재에 이르러 까지 진화를 해왔다.

2.2 악성코드의 동작과정

악성코드에 대한 피해 사례는 사용자 개개인의 PC를 떠나 국가적인 시스템에서도 피해가 발생되며 이 경우 국가 시스템 마비 등의 피해가 발생한다. 사용자 개개인에 대한 감염 피해 사례의 경우 메신저, SNS, 메일, P2P 서비스 등을 통하여 감염되는 사례가 많으며 이에 따른 유포 방식은 복잡해지고 지능화되어가고 있다. 국가적인 시스템에 대한 감염 피해 사례는 공격에 대한 방식이 고도화되었으며 사용자 개개인에 대한 감염이 목적이 아닌 국가 시스템을 목적으로 한다. 이러한 시스템은 대부분 내부 네트워크인 인트라넷(Intranet) 방식이기

때문에 공인 IP로는 접속을 시도할 수 없다. 하지만, 특정 외부 네트워크와 연결된 포트, USB, 상용 소프트웨어 둔갑 패치 파일 등 다양한 방식으로 국가 시스템을 악성코드에 감염시킨다.

2.3 악성코드 최근 동향

순위	악성코드 종류	개수	비율
1	랜섬웨어	275	44%
2	원격제어	224	35%
3	정보탈취	80	13%
4	파밍	38	6%
5	DDOS, 결제유도	16	2%
합계		633	100%

[그림 1] 한국 인터넷 진흥원(KISA) 사이버 위협 동향

랜섬웨어는 개개인의 PC 사용자가 그 대상이 아니라 국가인프라 시스템으로 그 영역이 확대되어가고 있다.

기존의 랜섬웨어는 불특정 다수를 겨냥한 무작위 공격의 형태를 지니었다면 최근에 발견되는 랜섬웨어는 타겟을 정해서 공격하는 타겟형 랜섬웨어로 점차 진화되고 있다. 최근 특정 그룹, 특히 민감한 정보와 환자 정보가 많은 의료기관을 타겟으로 한 랜섬웨어 사고 사례가 증가되고 있으며, 특정 그룹 내부에서 사용하는 인프라의 취약점이 있는지를 조사하고 이것을 활용하여 내부인프라 및 네트워크에 연결된 사용자 컴퓨터를 공격하는 형태를 보인다.

이러한, 타겟형 랜섬웨어가 증가하는 것에 대한 근본적인 원인은 국가인프라 시스템과 같은 대형 시스템은 저장된 정보의 가치에 비해 보안 시스템 업데이트가 느리다는 점이기 때문에 주요 타겟이 된다.

III. 행위기반 백신 개발

3.1 행위기반 백신 정의

행위기반 탐지기법은 악성코드 탐지 기법 중 하나로 실행 파일의 실행을 통해 이전까지 알려지지 않은 악성 코드를 탐지하는 방법이며, 시스템의 무결성을 보고 판단하는 방법과 시스템의 행위를 감시하는 방법으로 나눌 수 있다.

행위기반 탐지기법의 경우 기존에 주로 사용하던 시그니처 탐지기법과 비교하여 제로데이 공격 등 신규 취약점을 이용한 공격을 탐지할 수 있다.

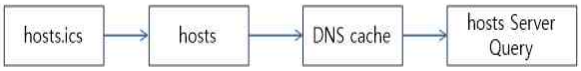
3.2 레지스트리 기반 백신

레지스트리는 윈도우 시스템에서 사용하는 시스템 구성 정보를 저장한 데이터베이스이다. 프로세서의 종류, 주기억장치의 용량, 접속된 주변장치의 정보, 시스템 매개변수, 응용소프트웨어에서 취급하는 파일 타입과 매개변수와 윈도우의 기본 설정 값이 저장되어 있다. 해커는 공격하고자 하는 컴퓨터의 레지스트리 값을 변조하여 악성코드를 정상 프로그램으로 위장하여 실행시킬 수 있는데, 본 행위 기반 백신의 경우 이 점을 이용하여, 해커가 특정 레지스트리 값을 변조 혹은 추가/삭제할 경우 이를 탐지하여 각 특성에 맞는 조치를 수행한다.

```
 varSubkey="SOFTWARE\Microsoft\Windows NT\CurrentVersion"
# 서브레지스트리 목록 지정
 varReg=ConnectRegistry(None, HKEY_LOCAL_MACHINE)
# 루트 레지스트리 핸들 객체 얻기
 varKey=OpenKey(varReg, varSubkey)
# 레지스트리 핸들 객체 얻기
 for i in range(1024):
   try:
        keyname = EnumKey(varKey, I)
# 지정한 레지스트리의 하위 키값 조회
        if("Winlogon" in keyname):
          varSubkey2 = "%s\\%s" % (varSubkey,
keyname)
# 하위 레지스트리 목록 생성 : 상위 레지스트리 목록과
하위 키값 결합
          varKey2 = OpenKey(varReg, varSubkey2) #
레지스트리 핸들 객체 얻기
          SubKey= OpenKey(HKEY_LOCAL_MACHINE,
varSubkey2 ,0,KEY_ALL_ACCESS)
```

3.3 DNS 행위 기반 백신

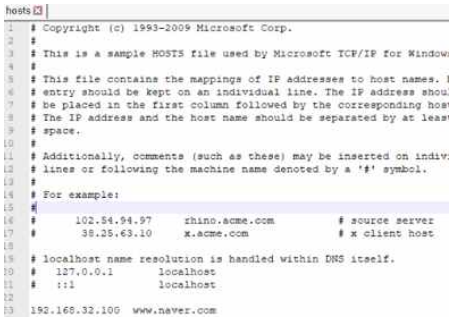
DNS(Domain Name Server)는 호스트의 도메인 이름을 호스트의 네트워크 주소로 바꾸거나 그 반대의 변환을 수행할 수 있도록 개발이 되었다. 오늘날 우리가 사용하는 웹에 대한 주소는 모두 IP로 변경해주는데, 이 기능이 DNS이다. DNS Pharming은 해당 변환 기능의 취약점을 이용한 피싱(phishing) 공격의 유형 중 하나로 해커가 제작한 가짜 사이트로 사용자를 유도시키는 공격 기법이다. DNS Pharming 공격의 특징은 DNS Query 순서에 있다. DNS의 쿼리는 다음과 같이 진행된다.



[그림 2] DNS Query 우선 순위 및 순서

○DNS hosts 파일 변조

hosts 파일 변조는 하단에 IP와 URL이 존재할 경우 해당 DNS Query가 우선순위가 된다. 때문에, 대부분의 악성코드가 피해자 PC를 감염시킨 뒤 아래 그림[3] 와 같이 hosts 파일을 변조하여 공격자의 웹 사이트로 유도할 수 있다.



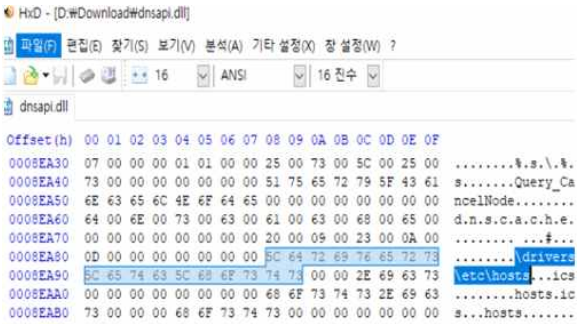
[그림 3] hosts 파일 구조

본 행위 기반 백신은 hosts 파일을 static으로 지정하여 무결성을 유지하는 방식을 채택하였으며 코드는 다음과 같다.

```
try:
    hosts=open("\drivers\etc\hosts"","r")
   if USER_OS=="Windows-10" or USER_OS=="Windows-08" or
USER_OS=="Window-07" :
    hosts=open(HOST_PATH_1+"\hosts","r")
   elif USER_OS_BIT=='64bit': hosts=open(HOST_PATH_2+"\hosts","r")
except:
    print("Parming_hosts Func OS ERROR")

HOSTS=hosts.readlines()
for hosts_data in HOSTS:
   if '#' in hosts_data:continue
   elif hosts_data=='\n':continue
   else:
    HOSTS_IP=".".join(re.findall('\d+',hosts_data)[0:4])
    HOSTS_DNS=".".join(re.findall('\w+',hosts_data)[4:])
    hosts.close()
    return
```

○dnsapi.dll 파일 변조

가장 최신의 DNS Pharming 공격 기법으로 위 DNS hosts 파일에 대한 시스템적 무결성을 유지하는 것을 공격자들이 우회한 형태이다. dnsapi.dll 파일은 hosts 파일의 실행 및 참조에 대한 경로를 지녔으며 DNS Query를 요청할 경우 hosts 파일의 경로명을 파밍 리스트가 담겨진 파일의 경로명으로 변조 할 수 있으며 경로 및 위치는 아래 그림[4]과 같다.



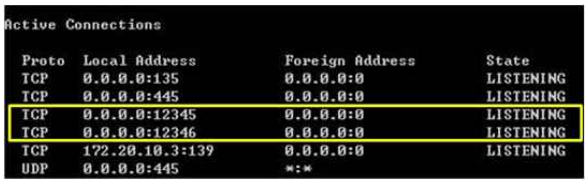
[그림 4] dnsapi.dll 파일 경로

위 그림[4]에 포함된 바이너리 경로를 '(시스템 폴더)\Drives\etc\hosts'가 아닌 '(시스템 폴더)\Drivers\(랜덤)\(랜덤)'으로 변경할 시 해당 바이너리 경로의 파밍 리스트로 DNS Query를 요청한다. 본 행위 기반 백신은 dnsapi.dll 파일에 대한 무결성이 침해되었을 시 탐지하고자 하였다.

```
DNS_API_DLL="C:\Windows\System32\dnsapi.dll"
DNS_API_DLL_PATH="\drivers\etc\hosts"
with open(DNS_API_DLL,"rb") as DNS_API_FILE:
   DNS_FILE_DATA=DNS_API_FILE.read()
   if DNS_API_DLL_PATH not in DNS_FILE_DATA:
        DNS_API_FILE.close()
        return
```

3.3 Backdoor 탐지

Backdoor의 특징은 일부 정상적인 연결은 주요 사용 포트에 LISTENING(접속 대기) 상태인 것에 비해 잘 알려지지 않은 비정상적인 포트가 LISTENING 상태임이 대부분이다. 때문에, 일반적인 사용자 PC에서 사용자가 먼저 소켓(socket) 통신을 요청하기에 시스템이 아닌 일반 사용자 PC 일 경우 열려있는 (LISTENING) 포트에 존재는 없어야 하며, 아래 그림[5]는 Backdoor 포트에 대한 감염 예시이다.



[그림 5] Backdoor 포트 LISTENING 상태

본 행위 기반 백신은 Backdoor의 특징을 이용하여 대중화된 Backdoor 상용 도구 혹은 의심 프로세스를 탐지하도록 설계 하였다.

```
SYSTEM_PORT_LIST=[80,135,443,445,903,913,8673,3306]
User_Process_List=['System','svchost.exe','wininit.exe','
services.exe','spoolsv.exe','lsass.exe']
Malware_Process_List=['nc.exe']

def backdoor_port1():
    while True:
      process_list=psutil.process_iter()
      for process in process_list:
        if process.name() in User_Process_List:
            continue
        elif process.name() in Malware_Process_List:
            return

def backdoor_port2():
   User_Process_List=['SystemIdle Process','csrss.exe',
'System','svchost.exe','winlogon.exe','wininit.exe','smss
.exe','services.exe','Registry','spoolsv.exe','lsass.exe','o
Cam.exe','Idle']
    while True:
      process_list=psutil.process_iter()
        for process in process_list:
         if process.name() in User_Process_List:
             continue
             connect=process.connections(kind='tcp')
         for status in connect:
             try:
               LIP,LPORT=status.laddr
               except ValueError:
                 continue

              if LIP=='0.0.0.0':
               if LPORT not in SYSTEM_PORT_LIST:
                 return
               else:
                 try:
                  RIP,RPORT=status.raddr
                      --생 략--
```

3.4 기타 행위 기반 백신

본 행위 기반 백신에는 앞서 설명한 것 이외에도 취약점 포트(weekness port) 탐지, 비트코인 마이닝 (bitcoin mining)탐지, 랜섬웨어(Ransomware) 차단, 악성 서비스 탐지와 같은 기능이 구현되어있으며 해당 위협 행위의 특징을 기반으로 백신 개발을 수행하였다.

○ Weekness Port

공공 기관에서 사용되지 않는 공유 폴더 포트, 네트워크 드라이브 포트에 대한 사용 여부를 검출한다. 또한 일반적으로 TELNET, FTP 포트와 같은 암호화되지 않는 포트는 사전에 미리 차단 혹은 실시간 스니핑을 통해 네트워크 패킷 정보를 확립하고 위 포트에 사용 여부 탐지 시 검출한다.

○Bitcoin Mining

비트코인 마이닝은 대체적으로 고 CPU 연산 처리를 요구하는 것에 비해 메모리 점유율이 상대적으로 낮다. 대부분의 상용화된 고사양 S/W 같은 경우는 CPU/메모리 점유율이 상관관계를 이루는 것을 이용하여 프로세스 별 CPU/메모리 사용률을 탐지하여 의심 프로세스를 검출한다.

○Service

윈도우즈 서비스(Windows Service)에는 자동 실행 기능이 존재하며, 윈도우즈 부팅 시 악성코드가 실행하도록 하는 매개체 역할을 할 수 있기에 악성코드 개발자들이 주로 활용하는 요소 중 하나이다. 자동 실행 기능을 수행하기 위해선 악성코드에 대한 절대 경로가 요구되며 해당 경로에 대한 악성코드를 분석하는 형식으로 악성코드 자동 실행을 방지할 수 있다.

IV. 결론

본 논문에서는 각종 악성코드에 대한 특징을 고려한 행위 기반 백신에 대해서 설명을 하고, 이에 대한 탐지 및 차단 방법을 설명하였다. 행위 기반 백신은 사용자의 환경을 고려한 엔드포인트 기반이므로 무엇보다도 섬세한 작업이 요구되기에 세밀한 연구가 진행되어야 한다.

[참고문헌]

[1] 유진호, 김민수.(2016) 한국인터넷 진흥원 연구보고서 '악성코드 유포 탐지기술 현황 조사 및 발전모델 연구'

[2] KISA (2016) 한국 인터넷 진흥원 사이버 위협 동향 보고서

Firm-Pot: 악성코드 분석을 위한 대규모 펌웨어 허니팟

김민근, 김은수, 김동관*, 김용대*
KAIST 정보보호대학원
*KAIST 전기 및 전자공학과

Firm-Pot: Large-scale Firmware Honey-Pot for Malware Analysis

Minguen Kim, Eunsoo Kim, Dongkwan Kim*, Yongdae Kim*
Graduate School of Information Security, KAIST
*Department of Electrical and Engineering, KAIST

요 약

최근 다양한 IoT 기기들이 일상생활에서 많은 이로움을 주고 있으며, 그 수는 폭발적으로 증가하고 있는 반면에, 이러한 IoT 기기들의 취약점을 악용하는 공격들도 점점 더 증가하고 있다. 본 연구는 IoT 기기들의 안정성을 높이기 위해 IoT 기기 펌웨어 에뮬레이션 도구인 Firmadyne을 재구성한 대규모 악성코드 분석도구인 Firm-Pot을 제시한다.

Firm-Pot은 기존 Firmadyne의 낮은 성공률을 획기적으로 개선하였으며, 이를 토대로 자동화 및 병렬화기능을 지원하며, 취약점 분석을 위한 디버깅 인터페이스와 네트워크와 시스템콜 모니터링 기능 또한 탑재되어있다. 마지막으로 본 연구에서는 실제 임베디드 기기를 대상으로 하는 악성코드의 동작과정을 Firm-Pot을 이용해 분석해본다.

I. 서론

임베디드 기기는 특정한 목적을 위한 하드웨어와 이를 제어하기 위한 내장된 컴퓨팅 시스템을 지닌 전자 기기들로 다양한 분야에서 사용되고 있다. 특히 IoT 환경에서는 가정용 유무선 공유기를 기반으로, AI 스피커, IP 카메라, 네트워크 프린터, 스마트TV와 같은 많은 임베디드 기기들이 인터넷에 연결되어 활용되고 있으며, 전 세계 인터넷에 연결된 임베디드 기기들의 수는 2020년에 300억개를 넘어설 것이라 예상될 정도로 빠르게 성장하고 있다.

이렇게 임베디드 기기의 수가 늘어나면서, 이를 대상으로 하는 공격 또한 늘어나고 있다. 최근 Mirai botnet과 Bashlight와 같은 IoT 임베디드 기기를 대상으로 하는 악성 프로그램의 공격들이 발생하고 있다. 특히 Mirai botnet의 경우 38만개의 임베디드 기기를 장악하여 수많은 임베디드 기기들이 보안상 취약한 상태이며, 이를 이용한 DDoS 공격으로 그 위험성을 적나라하게 보여주었다.

따라서 수많은 임베디드 기기들에 대해서 보안 취약점에 대한 분석과 botnet과 같은 악성 프로그램에 감염되었을 경우 이를 탐지하고 악성 행위를 분석하기 위한 기술이 필요하다. 이에 본 연구에서는 새로운 임베디드 기기 펌웨어 에뮬레이션 및 악성코드 분석 시스템을 제시하고자 한다.

임베디드 기기들을 에뮬레이션 하는 방법에 대해서 기존에 많은 연구가 있었다. 커스텀 하드웨어를 이용한 에뮬레이션 도구인 Avatar[1]는 임베디드 기기에서 사용하는 다양한 주변장치들을 하나의 하드웨어에 모두 연결하고, 이 하드웨어를 활용해 에뮬레이션된 펌웨어가 필요로 하는 I/O 처리를 대신 수행하도록 했다.

※이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2014-6-00908, 난수발생기 및 임베디드 기기 안전성 연구)

하지만 하드웨어가 필요하기 때문에 대규모 테스트에는 적합하지 않다는 단점이 있었다. 어플리케이션 수준의 방법으로 chroot와 QEMU를 활용하여 펌웨어를 에뮬레이션해 취약점을 찾는 연구[2] 또한 존재하였으나 많은 임베디드 기기들이 NVRAM과 같은 하드웨어에 직접 접근하기 때문에 에뮬레이션에 한계가 있었다.

Firmadyne[3]은 QEMU를 활용하여 임베디드 기기의 펌웨어를 이용해 자동화된 동적 분석을 수행하는 프레임워크로, 많은 임베디드 기기에서 사용되는 하드웨어인 NVRAM의 에뮬레이션을 통해 어떠한 물리적 장치도 필요없이 수많은 임베디드 기기들의 펌웨어를 에뮬레이션을 시도했다. 그 결과 10% 정도의 펌웨어에서 네트워크 연결까지 성공함을 보여줬다.

본 연구에서 구현한 Firm-Pot은 Firmadyne을 기반으로 에뮬레이션 과정을 IoT 환경에서 핵심적인 역할을 하는 유무선 공유기들에 최적화하여 에뮬레이션 성공률을 높였다. 그리고 에뮬레이션 과정을 병렬화하여 대규모 분석이 가능하게하고, 분석을 위한 정보 수집 기술을 강화하였다. 이를 통해 에뮬레이션 된 환경에 대해서 수집된 악성코드나 악성 펌웨어, 취약점에 대한 공격 검증 코드 등을 수많은 펌웨어를 대상으로 테스트 해볼 수 있으며, 각 공격이 어떤 방식으로 작동하고, 어떤 행위를 유발하는지 분석할 수 있다. 또한 여러 개의 에뮬레이션 된 펌웨어들을 외부 네트워크로 공개하여 허니팟을 구성하면 다양한 대상들에 대한 새로운 공격이 발생하는 것도 탐지할 수 있을 것으로 보인다.

II. 시스템 구조 및 구현

Firm-Pot의 구조는 그림 1과 같다. 우선 임베디드 기기의 펌웨어를 인터넷에서 수집한 후, 펌웨어를 대규모로 에뮬레이션 한다. 에뮬레이션된 펌웨어들은 외부 네트워크와 연결되어 허니팟으로 동작하게 되며, 이 허니팟을 대상으로 공격이 수행되면 공격 코드들을 수집할 수 있다. 나아가 분석가는 허니팟을 활용하여 수집된 공격 코드들을 손쉽게 분석을 수행할 수 있다.

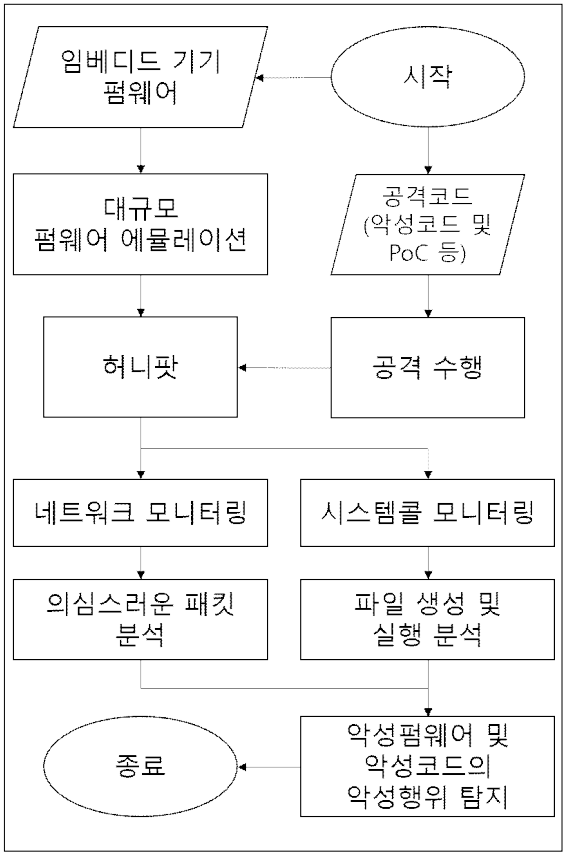


그림 1 임베디드 악성코드 분석 다이어그램

허니팟에서는 송수신되는 네트워크 패킷과 호출되는 주요한 시스템콜들을 모니터링해 다양한 시스템정보들을 수집한 후, 이 정보들을 가공하고 조합하여 악성펌웨어나 악성코드의 기능 및 알고리즘을 파악하게 된다. 파악된 정보들은 추후 악성코드를 자동으로 탐지하기 위해 활용될 수 있다.

2.1 펌웨어 에뮬레이션 최적화

기존 Firmadyne은 다양한 종류의 수많은 펌웨어들을 대상으로 에뮬레이션을 시도했고, 네트워크 연결을 기준으로 에뮬레이션 성공률은 10%정도에 그쳤고, 본 연구에서 사용하는 유무선 공유기의 데이터 셋에서는 웹 서비스의 가용성을 기준으로 기존보다 약간 높은 19%의 성공률을 보여줬다. 본 연구는 Firmadyne의 유무선 공유기 및 IP카메라의 펌웨어들을 대상으로 다양한 에뮬레이션 실패 원인들을 분석하여 에뮬레이션 성공률을 증가시켰다.

가장 먼저 펌웨어가 QEMU에서 올바르게

부팅하고, 알맞은 네트워크 인터페이스를 찾아서 호스트시스템과 네트워크 통신이 가능하도록 만들기 위해, 펌웨어가 초기화하는 전반적인 과정의 커널로그를 분석했다. 첫 번째 문제점은 올바른 초기화프로그램을 선택해주지 않은 것이었다. 기존의 Firmadyne은 커스텀 초기화 스크립트인 preInit.sh을 이용하여 QEMU가 부팅을 시도하도록 설정되어있었는데, 하지만 많은 임베디드 기기들이 rcS 또는 preinit이라는 스크립트 및 프로그램을 이용하여 초기화 하는 것을 발견했고, 이 프로그램들로 QEMU가 부팅하도록 Firmadyne을 개선해줌으로써 기존 Firmadyne의 에뮬레이션 성공률을 증가시킬 수 있었다.

두 번째로는 에뮬레이션된 펌웨어가 하나의 IP에 대해 여러 개의 네트워크 인터페이스로 연결하도록 시도하거나, 하나의 네트워크 인터페이스에 여러 개의 IP를 할당하는 경우가 있었는데, 이 경우 기존의 Firmadyne은 첫 번째 인터페이스로 연결을 시도했다. 우리는 이러한 중복되는 케이스에 대해 하나하나 QEMU로 네트워크 인터페이스에 연결하도록 시도하여 추가적으로 에뮬레이션 성공률을 올릴 수 있었다.

마지막으로 에뮬레이션된 NVRAM library에서 default key value를 초기화해주기 위해 NVRAM 데이터가 존재하는 파일 형태에 대해 분석했고, 기존의 Firmadyne에서 지원하지 못하는 Null-byte 형태로 분리되어있는 파일을 지원할 수 있도록 추가했다.

2.2 docker 기반 병렬화

에뮬레이션 성공률을 증가시킨 것만으로는 대규모의 실험을 진행할 수 없으므로 병렬화 작업 또한 수행하였다. 하지만 IoT 기기의 동적 IP환경 특성상 웹서버에서 페이지를 전달할 때 내부에 저장된 동적 변수 IP를 사용하는 케이스가 있었으며, 이러한 경우 host가 라우팅과 같은 방법을 통해 다른 IP로 에뮬레이션된 펌웨어에 접속했을 경우 IP의 불일치로 인해 더 이상 탐색을 할 수 없는 경우가 발생했고, 이를 위해 독립된 네트워크를 제공해줄 수 있어야했다. 그래서 본 연구에서는 QEMU에 독립된 네트워크 환경을 제공해줄 수 있는 솔루션이 필요했고, OS수준의 가상화 프로그램인 Docker를 사용하기로 했다. Docker의 Container를 이용함으로써 간단하게 QEMU에게 독립된 네트워크 환경을 제공해줄 수 있었다.

2.3 허니팟

이렇게 성공한 대규모로 에뮬레이션된 펌웨어들은 다양한 프로토콜들을 제공하게 되며, 이러한 프로토콜들은 로컬 네트워크를 통해서 직접 공격코드를 전달해 악성코드 분석에 사용될 수 있으며, 외부네트워크에 연결해 인터넷에 공개함으로써 허니팟으로 만들 수 있다. 허니팟은 실제 인터넷 환경에서의 공격을 기다리게 되고, 지속적인 모니터링을 통해 공격이 들어오는 것을 확인할 수 있게 된다.

2.4 공격 및 모니터링

허니팟으로의 공격은 인터넷에서 임베디드 기기를 대상으로 하는 악성코드를 수집해 직접 전송하거나 외부네트워크에 연결한 후 shodan과 같은 서비스에 직접 등록함으로써 외부로부터 적극적으로 공격이 들어오도록 한다.

이러한 공격들은 네트워크 및 시스템콜 모니터링을 통해 다양한 정보의 형태로 저장된다. 네트워크 패킷은 host 시스템에서 tcpdump 프로그램을 통해 모니터링 함으로써 초기 공격의 시작점을 알아낼 수 있도록 한다. 시스템콜은 리눅스커널의 시스템콜 후킹 도구인 kprobe를 활용하여 모니터링하게 되며, 악성코드들이 주로 사용하는 경로들이나 실행되는 프로그램들을 참고한다.

2.5 탐지

분석가는 수집된 네트워크 패킷 및 시스템콜 정보들을 이용하여 실제 공격의 유무를 판단하게 되며, 판단에 활용하기 위한 몇 가지 유용한 정보들이 존재한다. 예를 들어, 네트워크 패킷 모니터링을 통해 공격자가 원격에서 임베디드 시스템에 로그인을 시도하거나, 공격코드를 전송하는 것을 확인할 수 있고, 시스템콜 모니터링을 통해서는 악성 프로그램이 파일시스템에

생성되거나 실행되는 것들을 확인할 수 있다. 분석가는 이러한 인바운드 네트워크 및 시스템콜 호출 정보들의 체인을 잘 연결한다면 실제 공격을 탐지할 수 있다.

III. 실험 및 결과

3.1 데이터셋

에뮬레이션 성공률을 증가시키기 위해 D-Link, tp-link, NETGEAR 3개의 벤더사로부터 유무선 공유기 및 IP카메라로 총 546개의 펌웨어들을 다운로드 받았다. 이후 펌웨어들에 대해 2.1에 작성하였던 에뮬레이션 성공률 외에 벤더 및 펌웨어에 따른 상세한 에뮬레이션 기법들을 추가로 적용을 했고, 이후 비교군을 만들기 위해 추가적으로 동일한 3개의 벤더에서 가장 최신 펌웨어 257개를 다운로드 받아서 추가로 에뮬레이션을 진행했다.

3.2 에뮬레이션 성공률

에뮬레이션 성공률 증가에 사용된 펌웨어들을 대상으로 네트워크 연결성, 웹 서비스 가용성을 기준으로 에뮬레이션 성공률은 기존의 Firmadyne은 각각 33%, 19%의 성공률을 보여주는 반면에 Firm-Pot은 각각 92%, 86%의 성공률을 보여줬다. 또한 추가적으로 다운로드 받은 최신 펌웨어들에 대해서는 기존 Firmadyne은 각각 34%, 15%의 성공률에 대비해 Firm-Pot은 78%, 64%의 높은 성공률을 보여줬다.

3.3 악성코드 테스트

본 연구는 악성코드 테스트를 위해 Mirai botnet을 사용했다. 실제 소스코드를 다운로드 받아 컴파일 하여 펌웨어가 에뮬레이션된 허니팟에 공격을 수행하여 네트워크 패킷 및 시스템콜 정보들을 수집하였고, 네트워크 모니터링으로부터 botnet이 telnet을 통해 유무선 공유기에 직접 연결하는 정보와 ELF형태의 바이너리, 그리고 '/proc/.nippon', '/sys/.nippon', '/tmp/.nippon', '/.nippon'과 같은 추가적인 파일들이 저장되는 폴더 이름들을 패킷으로부터 발견할 수 있었다. 최종적으로 이러한 경로들에 생성된 바이너리가 실행되는 것 역시 open, write, execve 시스템콜들을 모니터링해 확인할 수 있었다.

IV. 결론

Firmadyne을 활용한 대규모 악성코드 분석 허니팟 시스템인 Firm-Pot은 유무선 공유기 및 IP카메라들을 대상으로 기존보다 훨씬 높은 에뮬레이션 성공률을 보여줬고, 네트워크 및 시스템콜 모니터링을 통해 악성코드의 행위를 분석하여 해당 악성코드의 기능 및 알고리즘을 파악할 수 있음을 보여주었다. Firm-Pot은 악성코드 분석가들이 허니팟을 구성하거나 임베디드 기기의 악성코드들을 분석할 때 유용하게 활용될 수 있으며, 유무선 공유기의 제조사에서는 제품의 보안을 높이기 위한 내부 테스트에서도 활용될 수 있다.

[참고문헌]

[1] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti. AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares. NDSS. 2014.

[2] A. Costin, A. Zarras, and A. Francillon. Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces. ASIACCS. 2016.

[3] D. D. Chen, M. Woo, D. Brumley, and M. Egele, Towards Automated Dynamic Analysis for Linux-based Embedded Firmware. NDSS. 2016.

API 별 가중치 기반 스코어링 기법을 적용한 파일리스 악성코드 탐지 기법 연구

양정은*, 배순규**, 오세준***, 이세영****, 최찬우*****, 김재성******,
김기홍*******, 전상현********

*서울여자대학교, **중앙대학교, ***선린인터넷고등학교, ****미림여자정보과학고등학교,
*****서울디지텍고등학교, ******동아대학교, *******(주)세인트시큐리티, ********시큐레터(주)

A Research of Fileless Malicious Code Detection Method Using API Weighting Based Scoring Technique

Jung-Eun Yang*, Sun-Gyu Bae,** Se-Jun Oh***, Se-Yeong Lee****, Chan-Woo Choi*****, Jae-Sung Kim******, Ki-Hong Kim*******, Sang-Hyun Jeon********

*Seoul Women's University,

**Chung-Ang University,

***Sunrin Internet High School,

****Mirim Girls Information Science High School,

*****Seoul Digitech High School,

******Dong-A University,

*******SAINTSECURITY Inc,

********SecuLetter Inc,

요 약

악성코드 탐지 기법에 대한 연구가 깊어질수록, 변종 혹은 신종 악성코드의 출현 또한 극심해진다.[1] 이에 따라, 최근 급증하고 있는 파일리스 악성코드는 파일 형태로 디스크에 쓰지 않고, 메모리에 바로 적재하여 실행하기 때문에, 비교적 탐지가 쉬운 기존 파일 형태의 악성코드보다도 고도화된 탐지 기법을 요구한다. 또한, 패턴 기반의 정적 분석과 가상 환경에서 미리 파일을 실행하여 분석하는 동적 분석만으로는 메모리에서 바로 실행되는 파일리스 악성코드를 탐지하는데 한계점이 존재한다. 이에 본 논문은 PC에서 파일리스 악성코드가 실행되는 순간 호출하는 API의 흐름에 따라 가중치를 달리 산정하여 악성 행위 여부를 식별할 수 있는 스코어링(Scoring) 기법에 대한 연구를 제안하고자 한다.

핵심어 : 파일리스 악성코드, 가중치, 스코어링 기법, API 모니터링

I. 서론

1.1 연구 배경

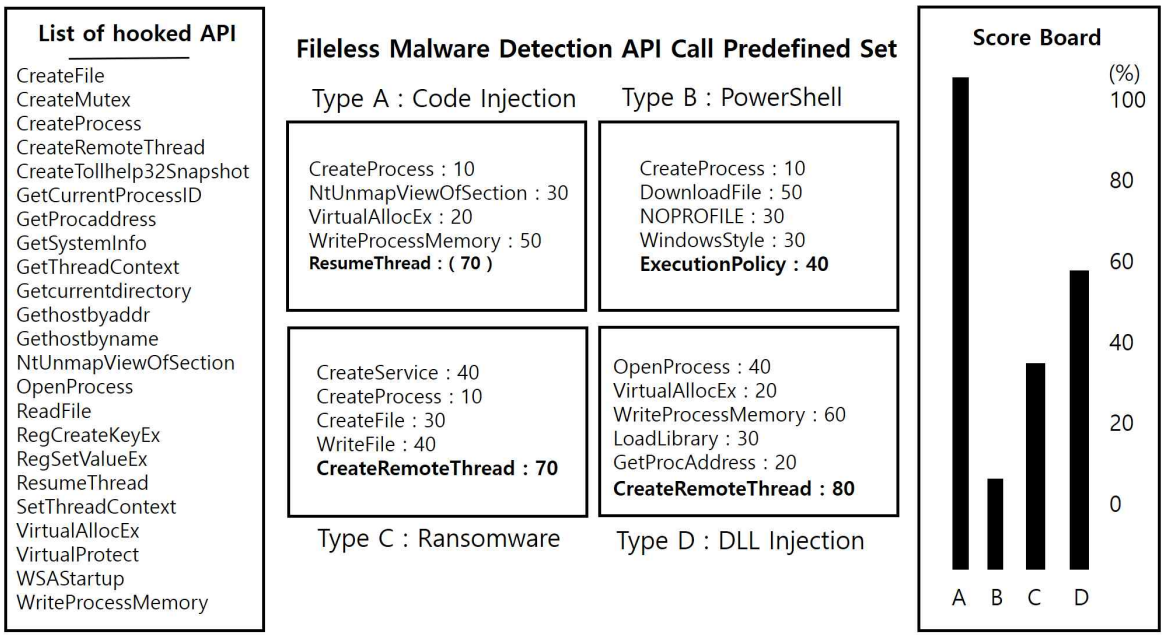
파일리스 악성코드(Fileless Malware)는 메모리에 직접 적재하여 실행하기 때문에, 시그니처(Signature)기반[2] 패턴 비교(Pattern Matching)의 정적 분석으로는 이를 탐지하는데 한계점이 존재한다. 이에 대응 방안으로 가상 환경에서 미리 파일을 실행한 후 정보를 수집하여 위험을 탐지하는 동적 분석이 가능하지만, 악성 행위를 식별하는데 많은 시간이 소요되어 탐지의 어려움이 있을 수 있다. 따라서,

기존 파일 형태의 악성코드를 탐지하던 정적 분석과 동적 분석 방법으로는 파일리스 악성코드를 탐지하기 어렵기 때문에 이에 대응 가능한 새로운 탐지 기법에 대한 연구가 필요하다.

1.2 연구 목적

파일리스 악성코드는 난독화, 패킹 기법으로 수많은 변종 악성코드를 배포하여 손쉽게 백신으로부터의 탐지를 우회할 수 있다. 파일리스 악성코드가 정상 프로세스에 붙어서(Attach) 동작하여 탐지를 우회하거나 회피하기 좋은 형태로 배포되기 때문에 탐지하기는 더욱 더 어렵다. 그렇기 때문에, 정상 프로세스라고 할지라도 API 호출 형태를 보고 악성 행위를 식별해야 할 필요성이 대두된다. 따라서 본 논문에서는 파일리스 악성코드 행위를 분석하여 호출하는 API가 미리 선별한 특정 API들의 호출 형태의 일부 인지를 파악하여 이에 가중치를 부여하고, 누적한 가중치가 특정 임계치 이상이면 악성 행위로 식별하는 스코어링(Scoring) 기법에 대한 연구를 제안하고자 한다.

1.3 가중치 스코어링 기법



[그림1] 가중치 스코어링 기법

Fileless Malware Detection API Call Predefined Set (Type A, B, C, D)

A's API Call Set : Code Injection

B's API Call Set : PowerShell

C's API Call Set : Ransomware

D's API Call Set : DLL Injection

본 논문에서 제안하는 탐지 기법은 악성 행위를 위해 호출(Call)하는 API, 약 150개 정도를 후킹(Hooking)하여 파일리스 악성코드를 탐지하기 위해 해당 API들의 행위를 모니터링 한다.

[그림 1]에서는 그 중 23개의 후킹 된 API 목록(List of hooked API)만 보여주고, 이 중에서 파일리스 악성코드 유형에 따라 API의 호출 형태 묶음(Fileless Malware Detection API

Call Predefined Set)을 미리 정의하였다. 이는, 각 파일리스 악성코드 유형(Type)에 따라 유사한 API 호출 형태(API Call Set)를 묶은 것으로서, 각 각의 다른 파일리스 악성코드 유형이라는 것을 알 수 있다. [그림 1]에서는 총 4개의 API 호출 묶음(API Call Set)의 파일리스 악성코드 유형(Type A: Code Injection, Type B: Powershell, Type C: Rans omware, Type D: DLL Injection)만 보여주었지만, 추가로 더 많은 수의 API 호출 묶음(API Call Set)을 포함하여 탐지율을 높일 수 있다.

더욱 정확한 탐지를 위해 API 호출 묶음(API Call Set)에서 호출 하는 함수의 특정 매개 변수나 호출 전후 상황에 따라 악성 행위에 기여하는 정도를 가중치(단위: %)를 달리 산정하였는데, 이는 파일리스 악성코드의 악성 행위를 탐지하는 기준이 될 수 있다.

자세한 탐지 기법의 동작 방식은 다음과 같다. 후킹 된 API를 악성코드에서 호출하면 해당 API가 어떤 API 호출 묶음(API Call Set)에 정의되어 있는지 탐색하고, 악성 행위 API 호출 묶음의 일부인지 확인한 후, 호출한 API의 가중치를 점수판(Score Board) 내의 해당 API 호출 묶음(API Call Set)에 실시간으로 누적한다. 누적된 가중치가 특정 임계치, 100%를 넘어가게 되면 실행되고 있는 프로세스가 API 세트의 실행 흐름과 유사하여 해당 악성 행위를 하는 프로세스로 탐지하여 이를 사용자에게 알리고 API 호출(Call)을 차단(Block)하는 추후 작업을 수행함으로 탐지는 종료된다.

실제 API 호출 묶음에 가중치를 산정하여 누적하는 방식은 다음과 같다. [그림 1] API 호출 묶음 중, 실제 파일리스 악성코드의 유형인 Code Injection (Target Process 에 원하는 실행 코드를 삽입하는 기술)은 CreateProcess, WriteProcessMemory , ResumeThread 가 필수적으로 호출되기에 이들을 묶어 세트 " A "로 정의하였고, A세트에서 호출되는 CreateProcess 함수의 경우 CREATE_SUSPENDED"(프로세스를 정지 상태로 실행)인자 값을 포함하고 있기 때문에, 가중치 10(%)를 부여한다.

특정 프로그램에서 호출한 API가 후킹 되었다면, 예를 들어 CreateProcess 함수를 악성코드에서 호출하면, A 호출 묶음의 API 함수에 해당하므로, 스코어보드 A호출 묶음에 10(%)을 부여한다. 이어서, NtUnmapViewOfSection 함수를 연속으로 호출한다면, A 호출 묶음에 추가로 30(%)을 누적하여 API 호출 형태와 가중치가 총 40(%) 유사하다는 결론을 얻게 된다. 이렇게 누적한 스코어보드에 가중치가 100%가 넘는 API 세트가 존재한다면, 탐지 대상이 되는 프로세스는 해당 파일리스 악성코드를 실행시키는 악성 행위로 판단하여 탐지됨을 알리고 종료한다.

이러한 탐지 방법을 본 논문에서는 " API 별 가중치 기반 스코어링 기법 "이라고 정의하였고, 각 악성 행위를 실행하기 위해 호출하는 API 호출 흐름과 매개 변수, 전후 상황에 따라 가중치를 달리 부여한 파일리스 악성코드 탐지 방법이라고 할 수 있다.

1.4 실제 적용 결과

PID	Process Name	A	B	C	D	
3956	Dwm.exe	80	50	80	80	
6748	Ctfmon.exe	80	0	0	0	
5948	Dllhost.exe	80	0	40	40	
3768	Notepad.exe	100	80	80	80	DETECTED
4084	Conherence.exe	30	30	20	0	
4192	Chrome.exe	20	80	50	10	
3272	Everything.exe	80	80	80	80	
5564	EXCEL.EXE	80	10	10	10	
1232	WINWORD.EXE	80	80	80	50	
688	Cmd.exe	10	0	0	0	
4232	Svchost.exe	20	10	0	0	
5020	Procexp.exe	20	80	40	40	
6068	Explorer.exe	10	0	0	0	

[표 1] 가중치 스코어링 기법 적용 결과 - 1

PID	Process Name	A	B	C	D	
3056	Cmd.exe	10	80	80	80	
3324	Everything.exe	80	80	80	80	
2928	WINWORD.EXE	80	80	80	80	
816	Cmd.exe	30	80	20	0	
6176	POWERPNT.EXE	0	30	0	0	
4780	Dllhost.exe	0	20	0	0	
2508	Procexp.exe	0	80	20	20	
4748	Explorer.exe	10	40	0	0	
4524	EXCEL.EXE	80	20	50	10	
4264	aaaaa.scr	80	80	100	80	DETECTED

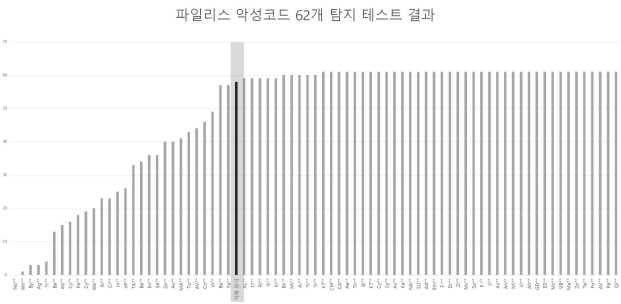
[표 2] 가중치 스코어링 기법 적용 결과 - 2

[표 1]과 [표 2]는 가중치 스코어링 기법

을 실제 적용하여 악성코드를 실행한 후 탐지한 결과이다. 모니터링 하고 있는 각각의 프로세스 아이디(Process ID : PID)별로 악성 행위를 나누어 가중치를 누적하는 방식을 사용하였다. 악성 행위는 4개(Type A, B, C, D)로 나누었으며 이는 앞서 1.3 가중치 스코어링 기법에서 정의한 파일리스 악성 코드 유형에 따라 분류한 API 호출 묶음(API Call Set)을 가리킨다.

[표 1]에서 프로세스 아이디(Process ID : PID)가 3768인 notepad.exe 는 Type A에 해당하는 가중치가 임계값인 100%에 도달하기 때문에, 이를 Code Injection(Type A)에 해당하는 악성 행위를 한다고 판단하였다. 그 외에 해당하는 프로세스에 대해서는 가중치가 100% 미만으로 나타나기 때문에, 악성 행위를 하지 않는다고 판단한다. [표 2]에서도 마찬가지로, 프로세스 아이디(Process ID : PID)가 4264인 aaaaa.scr 는 Type C에 해당하는 가중치가 임계값인 100%에 도달하였기 때문에 aaaaa.scr 는 Ransomware(Type C) 에 해당하는 악성 행위를 한다고 알 수 있다.

1.5 테스트 결과



[그림 2] 스코어링 기법 테스트 결과

본 논문에서 제안하는 탐지 기법을 테스트하기 위해 스코어링 기법을 적용한 윈도우 10 환경에서 총 62개의 파일리스 악성코드를 실행해 보았고, 그 결과는 다음과 같았다.

[그림 2] 에서는 정확한 탐지율 테스트를 위해 본 논문에서 제안하는 스코어링 기법 외에도 상용에서 쓰이는 백신 엔진으로 파일리스 악성코드를 탐지하는 결과를 얻을 수 있었다.

[그림 2]에서 알 수 있듯이 가중치 스코어링 기법을 적용하여 총 62개의 파일리스 악성코드 중 58개가 탐지되는 결과를 얻었고, 나머지 4개의 악성코드는 탐지하지 못하는 결과를 얻었다. 이는 파일리스 악성코드 실행 시, 접속하는 C&C 서버가 닫혔거나, 악성코드 실행 환경이 맞지 않는 오류가 발생하여 파일리스 악성코드가 악성 행위를 제대로 하지 못한 것으로 나타난다.

II. 결론

본 논문에서는 제안하는 " API 별 가중치 기반 스코어링 기법 "은 상용 백신이 탐지할 수 있는 범위 수준, 그 이상의 탐지율을 보임으로써 해당 기법은 충분히 파일리스 악성코드를 탐지할 수 있다는 결과를 도출할 수 있다.

또한, 해당 기법에서 제안하는 API 별 가중치 산정 방식은 파일리스 악성코드가 악성 행위를 위해 호출하는 API의 호출 형태만 미리 선별한 것이 아니라, 매개 변수, 호출 전 후 상황에 따라서 다른 가중치를 부여하기 때문에 고도화된 탐지 기법이라고 말할 수 있다. 해당 1.5 테스트 결과의 높은 탐지율를 통해 이를 입증하므로 향후 변종될 파일리스 악성코드를 탐지하는 데까지도 충분히 대응 가능한 탐지 기법이라고 제안할 수 있다.

[ACKNOWLEDGMENT]

본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best) 지원을 받아 수행되었음.

[참고문헌]

1) 송인수, 이동휘, 김귀남, 2010, " 시각화 기법을 이용한 악성코드 분석 및 분류 연구 ", 경기대학교 산업기술보호특화센터, 1

2) 이승국, 2015, "API 후킹을 통한 시간지연 우회기법이 적용된 악성코드의 탐지에 대한 연구", 숭실대학교 정보과학대학원, 17

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

구두발표 트랙 B

스마트 컨트랙트의 취약한 의사난수 생성 분석 및 대응방안

이진경[1], 김경민[2], 정윤서[3], 이진우[4], 강연호[5], 김규헌[2], 이상섭[6], 김경곤[7]

[1]서울여자대학교 정보보호학과 [2]고려대학교 사이버국방학과 [3]한국디지털미디어고등학교
[4]선린인터넷고등학교 [5]거제제일고등학교 [6]삼성리서치 [7]고려대학교

Analysis of Pseudo Random Number Generation in Smart Contract

Jin-Kyung Lee[1], Kyoung-Min Kim[2], Yun-Seo Jung[3], Jin-Woo Lee[4],

Yeon-Ho Kang[5], Kyu-Heon Kim[2] Sang-Sup Lee[6], Kyoung-Gon Kim[7]

[1]Department of Information Security, Seoul Women's University.

[2]Department of Cyber Defense, Korea University.

[3]Korea Digital Media High School. [4]Sunrin Internet High School. [5]Geoje Jeil High School.

[6]Samsung Research. [7]Korea University.

요 약

최근 블록체인의 활용도가 높아지며 스마트 컨트랙트를 구현하기 위한 플랫폼으로 자리 잡고 있다. 그 중 가장 많이 활용되는 플랫폼은 이더리움이다. 스마트 컨트랙트는 각 명령 코드마다 고유의 암호화폐를 소모하는데 이러한 특징으로 인해 대부분의 내장 함수를 지원하지 않는다. 따라서 개발자가 이를 직접 구현하는데, 그 과정에서 다양한 취약점이 발생한다. 해당 논문은 그 중 의사난수(이하 난수) 생성 함수를 잘못 구현하여 발생하는 예측 가능한 난수 취약점 문제에 대해 다룬다. 또한 여러 스마트 컨트랙트에서 찾은 취약한 난수 함수의 패턴을 분석하고, 난수 생성 함수를 위한 안전한 방안을 제시한다.

I. 서론

이더리움은 블록체인 기반의 스마트 컨트랙트 구현을 위한 플랫폼이다. 이더리움 개발자 Vitalik Buterin은 비트코인에 계약서 등의 추가 정보를 기록할 수 있다는 점에 착안하여, 스마트 컨트랙트 기능을 구현했다[1]. 코드 확장을 통해 무한한 확장성을 가진 탈중앙화 컴퓨팅 환경의 개발을 시행하여 단순한 검증과 연산을 뛰어넘는, 상태 및 함수를 정의하고 상태변이와 데이터 저장이 가능한 튜링 완전성을 갖춘 환경이 가능하도록 하였다[2].

스마트 컨트랙트는 코드에 적힌 계약 조건이 만족되면 그 즉시 계약이 성사되며 어떠한 검열이나 제3자의 간섭 없이 프로그래밍 된 대로 정확히 실행되는 프로그램이다. 또한 블록 내에 거래 기록뿐 아니라 조건문과 반복 명령어 등 실행 코드를 포함하는 것이 가능하여 최근 금융 결제 등 다양한 분야에서 활용되고 있다. 이러한 스마트 컨트랙트 작성을 위해 Gavin Wood와 이더리움 핵심 개발자들이 만든 계약 지향 프로그래밍 언어가 Solidity이다[1]. 본 논문은 스마트 컨트랙트 개발에 가장 많이 활용이 되는 Solidity 언어로 구현된 스마트 컨트랙트를 초점으로 둔다.

난수는 다양한 프로그램에 활용될 수 있으며, 동일하게 스마트 컨트랙트에서도 목적에 따라 난수가 필요할 경우가 발생한다[3]. 이더리움은 별도의 난수 함수가 내장되어 있지 않기 때문에 개발자들이 따로 구현하게 되는 특성을 갖고 있는데, 이는 예측 가능한 난수의 취약점 문

제로 이어질 수 있다.

본 논문에서는 블록체인 기반의 스마트 컨트랙트를 개발할 때 직접 구현하는 난수 함수들의 패턴을 분류하고, 분류된 취약점들에 대하여 분석하였다. 각 취약점들은 저자 구성원들이 찾은 CVE(Common Vulnerabilities and Exposures)들을 예시로 소개한다. 또한 그에 따른 대응 방안 연구에 대해 제시한다.

II. 배경 및 선행연구

2.1 난수 생성 함수의 부재

이더리움에는 난수 생성을 위한 함수가 존재하지 않는데, 이는 블록체인의 어떤 노드가 실행되더라도 같은 결과가 나오도록 트랙잭션을 검증하는 구조이기 때문이다. 스마트 컨트랙트에서는 트랜잭션을 미리 실행시켜보고 자신에게 불리할 경우 네트워크에 전파하지 않는 특성이 있다. 이러한 스마트 컨트랙트의 특성은 확률 요소가 있는 게임이나 복권 컨트랙트를 구현하는데 많은 문제가 된다. 또한 스마트 컨트랙트 내 난수 생성을 위한 더 큰 문제는 높은 가스(gas) 소모 문제를 발생시킨다는 점이다[4]. 이더리움은 전 세계가 하나의 블록체인을 활용하기 때문에 블록체인 사용량만큼 고유 화폐인 이더를 사용하게 된다. 스마트 컨트랙트의 경우 이더로 환산되는 가스를 소모하게 되는데, 난수 함수는 복잡한 알고리즘 때문에 큰 가스 소모가 예상된다. 그 외에도 이더리움 가상 시스템에서는 코드가 서로 다른 시간에 여러 노드에서 실행되므로 같은 난수 결과 값을 가져오기 위해 어려움이 있다[4].

2.2 컨트랙트 내 난수 함수의 취약성

실제로 이러한 스마트 컨트랙트의 무작위성을 구현하기 어려운 점은 악의적인 사용자에 의해 예기치 않은 기능을 초래하기도 한다. 2018년 스마트 컨트랙트 공동체를 위한 보안프로젝트 중 하나인, NCC그룹의 Decentralized Application Security Project (DASP) 에서도 'Bad Randomness', 즉 취약한 난수 문제를 대표적인 9가지 취약점 중 하나로 선정하였다.

III. Solidity 난수 생성방식의 문제점

본 연구를 진행하며 크게 다음 세 가지 형태의 취약한 난수 생성 방식을 확인했다. 첫 번째는 seed로 블록 변수(block variables)를 활용하는 것이며, 두 번째는 과거 블록의 블록해시를 활용하는 것, 마지막으로 과거 블록의 블록해시와 private 변수로 선언된 seed를 활용하는 것이다. 다음 소주제에서는 각 난수 생성 방식을 분석하고, 취약한 이유를 소개한다.

3.1 블록 변수를 활용한 난수 생성 함수

이더리움 solidity에서 제공하는 블록 변수는 다음과 같은 종류가 있다.

Block Variables	Description
block.coinbase	현재 블록을 채굴한 miner의 주소
block.difficulty	블록 채굴 난이도
block.gaslimit	트랜잭션 최대 가스 소비 제한
block.number	블록의 높이
block.timestamp	블록 채굴 시간

[표1. Solidity 기본 블록 변수]

우선, 블록 변수는 miner(채굴자)가 조작할 수 있다. 이러한 난수를 활용한 스마트 컨트랙트 중 miner가 충분한 보상을 주는 계약이 있다면 채굴 시간을 늦추는 등의 블록 변수를 조작할 수 있게 된다.

더 중요한 것은 블록 변수가 동일한 블록 내에서 공유된다는 것이다. 따라서 공격자의 컨트랙트가 내부 메시지를 통해 해당 컨트랙트를 호출하면 두 계약에서 동일한 난수결과를 산출한다. 아래는 CVE-2018-17968의 랜덤함수이다.

```
function random() private view returns (uint256) {
    return uint256(uint256(keccak256(block.timestamp, block.difficulty))%6);
}
```

[그림1. CVE-2018-17968의 랜덤함수]

3.2 블록 해시를 활용한 난수 생성 함수

이더리움 블록체인의 각 블록에는 블록 해시가 있다. Solidity에서 block.blockhash() 함수를

통해 이러한 블록 해시를 가져올 수 있다. 이 함수는 블록 번호를 인자로 받아온다. 본 연구를 통해 난수 생성 함수 구현에서 block.blockhash()를 오용하는 방식이 크게 두 가지임을 확인하였다.

첫 번째는 현재 블록 번호를 활용하는 것이다. 코드 상에서는 block.blockhash(block.number)를 활용하는 것이다. miner가 컨트랙트 코드를 실행하는 트랜잭션을 선택하면 향후 블록의 block.number가 알려지고 컨트랙트에서 해당 값에 안정적으로 접근할 수 있기 때문에 안전해 보인다. 하지만 Solidity를 구동시키는 Ethereum Virtual Machine(EVM)에서 이 컨트랙트를 실행될 당시에는 블록 생성이 되어있지 않아 항상 0으로 결정된다.

두 번째는 최근 마지막에 채굴된 블록 번호를 활용하는 방식이다. 코드로는 block.blockhash(block.number-1)을 통해 구현한다. 마지막으로 채굴된 블록의 블록 해시를 가져오는 것인데, 이는 앞에 3.1장에서 블록 변수를 활용했던 방식과 마찬가지로, 공격자는 내부 메시지를 활용해서 대상 계약을 호출하고 같은 난수 생성 함수 결과를 얻어낼 수 있다.

3.3 private 변수를 활용한 난수 생성 함수

난수의 무작위성을 높이기 위해 컨트랙트들 중 일부는 private 변수로 저장된 값을 활용하였다. Solidity에는 Visibility(가시성)이라는 지정자가 존재한다. 그 중 private 변수는, 파생되거나 외부 호출된 컨트랙트가 아니라 정의된 컨트랙트에서만 접근 가능하다. 하지만 컨트랙트 내부에 있는 변수는 모든 외부 사용자에게 공개된다. private 가시성이라도 다른 컨트랙트가 정보에 접근하고 수정할 수 없지만 여전히 블록체인 외부의 모든 사람들이 볼 수 있다. 이러한 특징은 블록체인의 투명성을 보여준다[2].

이더리움 블록체인에 접근하는 API인 web3를 통해서 컨트랙트의 변수를 저장하는 영역에 접근할 수 있다. 하지만 일부 스마트 컨트랙트 개발자들은 private변수를 읽을 수 없는 권한으로 오해하여, 알려지지 않아야하는 값을 저장함으로써 취약한 seed를 활용하게 된다. 이는 예측 가능한 난수 취약점 외에도 정보 누출(information leakage) 취약점이 발생할 수 있다. 다음 그림3은 이러한 취약점을 설명한다. 당첨번호인 maxTickets를 블록체인에서 검색할 수 있어 공격자는 항상 당첨될 수 있게 된다.

```
uint256 private maxTickets;
uint256 numtickets;

function PlayGame() {
    numtickets++;
}
function PayWinner( address winner ) private {
    require(numtickets == maxTickets);
    winner.transfer(winnerPrice);
}
```

[그림2. CVE-2018-15552의 의사 코드]

IV. 대응 방안

해당 챕터에서는 안전한 방식으로 난수를 생성하고 활용하는 방법에 대해서 제시한다. 이는 어느 정도 구조적인 개발 방법으로 해결될 수 있으며 크게 세 가지의 대응방안이 존재한다.

첫 번째는 미래의 블록 번호를 활용하는 것이다. 다음과 같은 도박 스마트 컨트랙트의 시나리오를 가정한다.

- 컨트랙트 이용자는 베팅값을 보내고, 컨트랙트는 해당 트랜잭션의 블록번호만 저장한다.
- 다음 컨트랙트에 요청할 시 이용자는 당첨번호를 부르는 함수를 호출한다.
- 당첨 함수에 저장된 블록번호를 검색하고 블록해시를 가져와 난수를 생성하는데 활용한다.

이러한 방식은 실제 블록넘버의 블록해시를 저장하고 이후에 활용한다는 점에서 난수성을 유지할 수 있다. 하지만 한 가지 제약조건이 있는데 EVM에서 블록 해시 검색은 확장성 문제로 최근 256개까지만 접근할 수 있으며, 256개의 블록 이전의 해시를 검색할 때는 0이 된다. 따라서 두 번째 호출이 256개의 블록 이내에 호출하지 않게 되면 0이 된다. 이는 약 51분 정도이며 이 시간 내에 컨트랙트 호출이 발생하지 않으면 예측 가능한 값이 된다.

그 외에는 컨트랙트의 내부 호출을 막는 방법이다. 앞에서 설명했듯이 공격자의 계약이 취

약한 컨트랙트의 내부 함수를 호출시켜 같은 난수 생성 함수 값을 가져오게 만드는 방식이 존재한다. 따라서 이러한 문제를 해결하기 위해 컨트랙트의 접근을 방지하는 방법이 존재한다. 이는 바로 'require (tx.origin==msg.sender);'를 활용하는 것이다. tx.origin은 트랜잭션의 원래 보낸 주소이고 msg.sender는 즉각적으로 보낸 주소이다. 따라서 트랜잭션을 컨트랙트 주소는 접근 불가하고, 일반 지갑 사용자 계정만 가능하게 된다. 이렇게 되면 내부 호출을 통한 난수 생성 함수 값 예측이 불가능해진다. 하지만 이는 반대로 컨트랙트의 접근을 불가능하게 만들어 가용성의 문제가 생길 수 있다. 단적인 예로, 컨트랙트로 개발된 다중 서명 지갑(multi-signature)은 해당 스마트 컨트랙트에 접근할 수 없게 된다.

다른 방법은 외부 API를 활용하는 것이다. 일부 컨트랙트를 개발할 때 블록체인에 제공되지 않은 정보에 의존해야할 경우가 존재한다. 예를 들어 스포츠 행사 결과, 항공편 상태 등과 같은 경우인데, 안전한 난수 함수를 외부적으로 빌려 활용할 때도 활용할 수 있다.

이러한 방식은 기존의 검증된 난수 함수를 활용하여 안전한 결과를 가져온다는 장점이 있다. 하지만 여기에는 중요한 제약조건이 존재한다. 바로 정보 제공자를 신뢰 할 수 있어야 한다는 점이다. 블록체인은 중앙 관리가 없기 때문에, 합의 알고리즘과 같은 개념으로부터 신뢰를 구축해야한다. 외부에서 알게 된 정보에 의존할 때 해당 정보에 대한 신뢰를 쌓을 수 있는 방법을 찾아야한다.

이러한 점에 있어서 탈중앙화된 블록체인의 이념과 대립적으로 하나의 독립주체를 믿어야 한다는 문제점이 발생된다. 또한 단일 취약점(Single Point of Failure)의 가능성이 있다. 해당 외부 API의 난수 함수가 문제를 일으킬 경우, 혹은 제공되는 데이터의 형태가 변화가 있을 경우 컨트랙트를 파기 시켜야할 수도 있다. 블록체인에 컨트랙트가 한 번 배포되면 수정할 수 없다는 불변성 때문에 새로운 컨트랙트를 개발해야 한다.

V. 결론

본 논문에서는 스마트 컨트랙트의 취약한 난수 생성방식의 패턴을 분석하고 더 안전한 난수 생성 방안을 제시하였다. 하지만 이더리움 블록체인에서 난수 생성 함수를 안전하게 구현하는 것은 여전히 어려운 과제이다.

난수 생성 함수를 디자인 할 때 개발자는 각 구현 요소들이 예측 가능한 값인지 고민을 해야 하고, 그에 따른 적절한 방법을 선택해야한다. 이를 위해 본 논문은 세 가지 대응방안들을 제시하였다. 해당 대응방안들에도 몇 가지 제약조건들이 존재하며, 이러한 제약조건을 해결 가능한 환경 속에서 채택해야 한다.

[참고문헌]

[1] V Buterin, A next-generation smart contract and decentralized application platform, white paper, 2014

[2] Wood, Gavin, Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper, 151, 2014.

[3] Nicola AtzeiMassimo, Bartoletti, Tiziana Cimoli, A Survey of Attacks on Ethereum Smart Contracts (SoK), Conference paper, 28 March 2017.

[4] D.S.Milojicic, V.Kalogeraki, R.Lukose,K.Nagaraja, J.Pruyne, B.Richard, S.Rollins and Z.Xu, Peer to Peer Computing, HP Laboratories Palo Alto HPL-2002-57, March, 2002.

블록체인 기반 인증 체계 구축 및 위협 분석

이가영, 이연수, 이혜지, 염흥열

*순천향대학교 정보보호학과

A Blockchain based Authentication system and its threat analysis

Ga-Young Lee, Yeun-Su Lee, Hye-Ji Lee, Heung Youl Youm*

*Department of Information Security Engineering, Soonchunhyang University.

요 약

최근 블록체인 기술을 활용한 서비스들이 다양한 분야에서 적용되고 있다. 그 중 금융 분야에서는 기존 금융 인증 서비스의 한계점을 개선하고자 블록체인 기술을 활용하여 새로운 인증 서비스인 공동인증서비스가 등장하였다. 하지만 여전히 블록체인의 기술적, 관련 법규에 대한 개선점이 요구되고, 공동인증서비스에 대한 보안 검토가 부족하기 때문에 금융 서비스에 피해가 발생할 수 있다. 따라서 본 논문에서는 공동인증서비스의 시스템 환경을 가정 및 설계하여 그에 따른 취약점을 분석하였다.

I. 서론

최근 블록체인 기술은 많은 사람들에게 주목을 받아 계속 발전해가고 있으며 다양한 분야에서 적용되고 있다. 특히 금융 분야에서 블록체인의 안정성과 효율성 때문에 더욱 활발하게 사용되고 있다[1]. 그 예로 최근에 국내 은행 기업들이 다수 참여하고 있는 공동인증서비스 애플리케이션이 출시되었다[2]. 공동인증서비스는 기존 금융 인증 서비스의 한계점을 블록체인 기술을 활용하여 개선한 시스템으로 다양한 장점 및 특징을 가진다[2]. 하지만 블록체인 기술이 급속도로 발전하여 보안 검토가 요구되고 있지만 블록체인의 보안 위협에 대한 연구는 활발하게 이루어지지 않고 있다[3]. 또한 금융 분야는 중요 데이터를 많이 다루고 있어 보안 위협이 발생하면 그 피해가 막대하기 때문에 블록체인 기술을 활용한 서비스들에 대한 보안 검토가 필요하다[3].

따라서 본 논문에서는 블록체인을 활용한 공동인증서비스 시스템을 설계하고 그에 따른 보안 위협에 대해 연구를 진행하였으며 다음과 같이 구성된다. 2장에서 공동인증서비스에 대한 정의 및 원리를 소개하며, 3장에서는 공동인증서비스 시스템을 설계하였다. 그리고 4장에서 설계에 따라 발생할 수 있는 공동인증서비스의 취약점을 분석하였고, 5장에서 본 논문에 대한 결론으로 마무리 짓는다.

II. 관련 연구

2.1 공동인증서비스 정의

온라인 거래, 자금 이체 등을 이용하기 위한 용도인 공동인증서비스는 블록체인의 원리 및 특성을 활용하여 인증서와 인증 데이터를 컨소시엄 블록체인 인프라에 저장하는 서비스를 가리킨다[4]. 해당 인증 서비스는 컨소시엄 블록체인 네트워크에 인증서와 인증 데이터가 저장되는데, 이때 데이터 위·변조 위험 감소 및 제 3자 없이도 사용자 인증이 이루어지는 시스템 구조를 지니고 있다[5]. 또한 블록체인 네트워크는 인증서와 인증 데이터를 관리하는 중앙기관

이 불필요하므로 유지 및 생성 비용이 감소하는 특징을 갖고 있다[4].

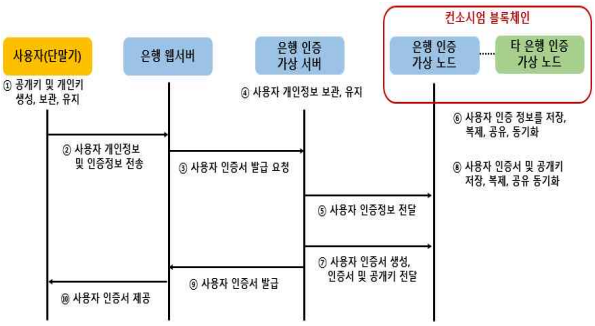
2.2 공동인증서비스 모델

공동인증서비스 모델은 사용자(단말기), 은행 웹 서버, 은행 인증 가상 서버, 블록체인 네트워크로 이루어져있다. 또한 공동인증서비스의 각 시스템은 다음 [표 1]과 같은 역할을 수행한다.

[표 1] 공동인증서비스 각 시스템 역할[4]

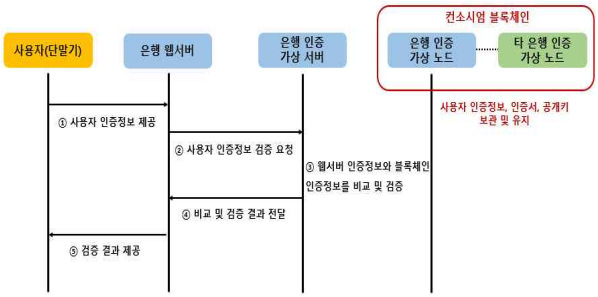
구분	역할
사용자 (단말기)	단말기에서 개인키 및 공개키 생성
	단말기 내, 개인키 보관 유지
은행 웹 서버	인증 가상 서버에게 사용자 인증정보 및 개인정보 전달
	사설인증서 발급 요청
은행 인증 가상 서버	사용자 개인정보 자체적으로 보관·유지
	사용자 사설인증서 생성
블록체인 네트워크	사용자 사설인증 및 공개키를 저장·복제·공유·동기화

공동인증서비스를 이용하기 위해서 사용자는 인증서 생성 및 등록을 필수적으로 거쳐야한다. 인증서 생성은 은행 인증 가상 서버에서 이루어지며 생성된 인증서는 컨소시엄 블록체인 네트워크에 저장된다. 이에 대한 자세한 과정은 [그림 1]과 같다.



[그림 1] 공동인증서비스의 인증서 생성 및 등록 과정[5]

상기의 과정을 거친 후, 사용자는 공동인증서비스 이용 전 등록과정에서 등록한 인증정보에 대한 인증 과정을 거쳐야한다. 사용자 인증 절차 과정에서는 은행 인증 가상 노드에 보관되어 있는 인증정보와 사용자가 제공한 인증정보를 비교 및 검증하는 과정을 포함한다. 이에 대한 자세한 과정은 [그림 2]과 같다.



[그림 2] 공동인증서비스의 사용자 인증 과정[5]

3장에서는 공동인증서비스 모델의 시스템 환경을 가정하고 설계했으며, 이에 대한 근거를 설명하고자 한다.

III. 시스템 설계

본 논문에서는 공동인증서비스의 취약점 분석에 앞서 공동인증서비스의 시스템 환경을 가정하여 설계하였다.

3.1 공동인증서비스 설계

공동인증서비스는 블록체인을 활용한 P2P 기반 금융 서비스이다. 이러한 블록체인 유형은 참여하는 네트워크의 관리 주체, 성격 등에 따라 퍼블릭, 프라이빗, 컨소시엄 등으로 구분할 수 있다[6]. 이 중에서도 공동인증서비스에서는 컨소시엄 블록체인을 활용하고 있다. 그에 대한 근거는 다음 설명과 같다.

컨소시엄 블록체인은 퍼블릭 블록체인과 달리 허가받은 사용자만 접근을 허용하여 보안을 강화할 수 있고, 한 소유자가 모든 권한을 가지게 되는 프라이빗 블록체인과는 다르게 미리 정해진 노드들이 권한을 가지게 할 수 있다. 또한 퍼블릭 블록체인은 안전성이나 최종성과 같은 문제점 때문에 P2P 기반 금융 서비스인 공동인증서비스엔 적합하지 않다[7]. 퍼블릭 블록체인에서의 안전성, 최종성 문제에 대한 설명은 다음과 같다[7].

-안전성에 의한 문제점은 퍼블릭 블록체인이 중앙 서버나 인증된 기관이 존재하지 않고

누구나 참여하여 통제나 관리하기가 어렵다는 점에서 발생한다.

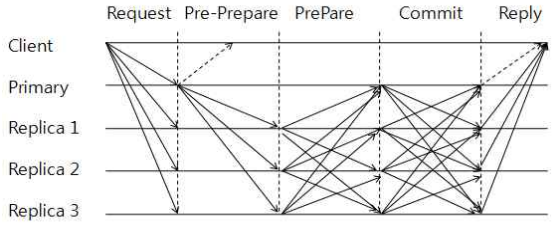
-최종성에 의한 문제점은 퍼블릭 블록체인에서 거래를 확정할 때, 일정 시간이 지나야 확정할 수 있다는 점에서 생긴다[7]. 이 일정 시간 동안 네트워크 분기가 생길 수 있기 때문에 즉각적인 처리와 정확한 데이터가 필요한 금융 분야에서는 적합하지 않다.

따라서 본 논문에서는 공동인증서비스가 컨소시엄 블록체인을 활용함과 동시에 PBFT 합의 알고리즘을 사용한다고 가정하였다. 그에 대한 근거는 다음과 같다.

퍼블릭 블록체인에서 채택하고 있는 분산합의 알고리즘인 PoW(Proof of Work)와 PoS(Proof of Stake)은 블록 생성 및 검증하는 과정에서 비용이 발생한다[7]. PoW는 채굴(mining)과정에 참여한 네트워크 이용자에게 보상을 지급하고, PoS는 PoW와는 다르게 새로운 블록 생성에 대한 보상을 지급하지 않지만 거래 비용이 발생한다[8]. 하지만 금융 서비스인 공동인증서비스에서는 이러한 비용을 필요로 하지 않는다[7]. 반면에 컨소시엄 블록체인에서 사용하는 PBFT 합의 알고리즘은 블록 생성 및 검증하는 과정에서 비용이 발생하지 않는다[7]. 또한 PBFT는 BFT(Byzantine Fault Tolerance)계열의 분산합의 알고리즘으로서 네트워크 분기를 허용하지 않기 때문에 퍼블릭 블록체인의 최종성 문제점이 발생하지 않는다. PBFT 합의 알고리즘에 대한 설명은 3.2와 같다.

3.2 PBFT 합의 알고리즘

PBFT(Practical Byzantine Fault Tolerance) 합의 알고리즘은 비잔틴 장애(Byzantine Faults)에 내성을 갖는 복제 알고리즘으로[8], BFT 합의 알고리즘과 다르게 분산 시스템에서 비동기식 네트워크 방식으로 발전된 합의 알고리즘이다[7]. PBFT 합의 알고리즘은 트랜잭션을 요청할 책임을 지는 Primary Node(주 노드) 한 개와 Replica Node 여러 개로 구성되어 있다[7][8]. 또한 전체 프로세스는 Request, Pre-Prepare, Commit, Reply 단계로 구분된다[7]. 각 단계에서 노드는 모든 노드의 2/3 이상에서 투표를 받은 경우 다음 단계로 진입하기 때문에 PBFT는 모든 노드가 네트워크에 알려져 있어야 한다[8]. 다음 [그림 3]는 오류가 없는 정상적인 PBFT 합의 알고리즘이 진행되는 것을 보여준다[9]. Replica 0이 기본이며, Replica 3은 결함을 나타낸다[9].



[그림 3] PBFT 합의 알고리즘

본 논문에서는 상기와 같은 여러 근거들을 바탕으로 공동인증서비스의 시스템 환경을 설계하였고, 이에 따라 발생하는 취약점을 다음 4장에서 설명할 예정이다.

IV. 취약점 분석

공동인증서비스는 블록체인 기술을 활용하여 나타나는 다양한 특징과 장점에도 불구하고 몇 가지 위협적인 보안위협이 존재한다. 4장에서는 본 논문 3장에서 설계한 시스템 환경을 바탕으로 공동인증서비스를 이용할 때 발생할 수 있는 보안 위협에 대해 다룬다.

4.1 PBFT 합의 알고리즘에 의한 취약점

PBFT(Practical Byzantine Fault Tolerance) 합의 알고리즘의 각 노드는 다른 노드들이 보내준 블록 유효성 검증 결과 값을 취합하고, 전체의 2/3이 초과하여 동일한 결과 값을 보냈을 경우 해당 결과 값을 참으로 인식한다. 따라서 만약 전체의 1/3이 악의적인 노드일 경우, 모든 블록의 결과 값을 참으로 조작이 가능하다는 위협점이 존재한다.

4.2 개인키 인증정보 유출

모든 사용자는 공동인증서비스를 이용하기 위해 각자 개인키를 보관하고 있으며 사용자가 가지고 있는 개인키가 해커의 악의적인 공격을 통해 유출이 될 수 있다. 개인키 유출 시 블록

체인 인프라에 접근할 수 있는 사용자 권한을 가질 수 있어 블록체인 내의 개인정보 등을 도난당할 수 있다. 또한 블록체인의 특성상 범죄자 행동 추적이 현실적으로 어렵다.

4.3 BGP Hijacking 공격

BGP(Border Gateway Protocol)는 IP 패킷이 전달되는 방식을 규제하는 라우팅 프로토콜이다[10]. BGP Hijacking 공격은 해커가 BGP 라우팅을 조작하여 블록체인 네트워크 트래픽 차단 및 블록의 전파 속도를 지연시키는 공격이다[10]. 인터넷 기반으로 연동되어 있는 BGP를 공격할 경우 모든 기업에 영향을 줄 수 있으며 라우팅 프로토콜에 문제가 발생하는 경우는 네트워크 관리자의 실수가 대부분 차지한다[11]. 또한 거래 수가 많은 금융 분야에서 사용되는 블록체인 서비스일수록 해당 공격에 대한 피해가 크다. 2018년 4월 전자지갑서비스인 'MyEtherWallet'이 Hijacking 공격에 당한 사례가 있다[12]. 해당 공격의 공격자는 이용자와 서비스를 연결하는 BGP와 DNS 취약점을 이용하였다[12]. 공격자는 BGP 메시지를 위조하여 DNS를 하이재킹한 뒤, 가짜 사이트로 접속을 유도하여 공격을 실행하였고, 당시 이 공격으로 인한 피해금액이 15만 달러에 달했다[12]. 이와 같이 금융 분야의 블록체인 서비스에 대한 BGP Hijacking 공격은 거래 정보에 직접적인 피해를 입힐 수 있다.

V. 결론

최근 주목받는 기술 중 하나인 블록체인은 안정성과 효율성을 기반으로 금융 분야에서 활발히 활용되고 있다. 하지만 지속적으로 블록체인의 보안 위협에 대한 문제점이 제기되고 있어 이에 따른 확실한 대응방안이 필요하다.

따라서 본 논문에서는 금융서비스 분야의 새로운 인증 방식인 공동인증서비스의 시스템 환경을 가정 및 설계하였고, 이에 따른 PBFT 합의 알고리즘에 의한 취약점, 개인키 도난 취약점, BGP Hijacking Attack 취약점을 분석하였다.

향후 연구로 해당 취약점에 대한 대응방안을 연구하고 공동인증서비스에서의 취약점을 추가로 분석하고자 한다.

[참고문헌]

[1] 박정호, '블록체인 산업 현황 및 동향', 정보통신산업진흥원, 2018.04

[2] 은행연합회 IT부, '뱅크사인 은행공동인증서비스 추진 현황', 2018.06

[3] 박준한, 김유성, 공수재, '블록체인 구현측면 정보보안 동향 및 시사점', 정보통신기술진흥센터, 2018.08

[4] 박근덕, '클라우드 기반의 분산원장기술을 활용한 전자금융기반시설 간의 공동인증 서비스에 대한 보안 프레임워크 제안', 순천향대학교 대학원, 2018.02.

[5] 이가영, 이혜지, 이연수, 염흥열 '블록체인을 활용한 공동인증서비스 기술 분석', 순천향대학교, 2018.05

[6] 홍승필, '금융권 블록체인 활용 방안에 대한 정책 연구', 전자금융과 금융보안 제6호, pp. 12-18, 2016.04.

[7] 박종혁, '개선된 PBFT 블록체인 기반의 On-Line P2P 금융거래용 플랫폼 설계', 한양대학교 공학대학원, 2018.08.

[8] 염흥열, '블록체인 보안과 프라이버시', TTA Journal Vol.177, 2018.05.

[9] Miguel Castro, Barbara Liskov, 'Practical Byzantine Fault Tolerance', Massachusetts Institute of Technology, 1999.

[10] 서신석, 홍성철, 홍원기, 'BGP 보안 위협 요소와 대처방안', 포항공과대학교, 2008

[11] 전성복, BGP 라우팅 프로토콜의 취약성 분석 및 개선방안, 서강대학교, 2007

[12] Joon Ian Wong, 'The internet's core infrastructure was hacked to steal ethereum from a popular wallet service', 2018.04

블록체인을 이용한 마이크로그리드 기기의 인증 기법 설계

염정철*, 이영석*, 나인호*
*군산대학교 컴퓨터정보통신공학부

A Design of Authentication Scheme using Blockchain in Microgrid Devices

Jeong-Cheol Yeom*, Young-Seok Lee*, In-Ho Ra*
*School of Computer, Information and Communication Engineering
Kunsan National University.

요 약

블록체인은 거래정보를 기록한 장부를 중앙서버에 저장하지 않고 네트워크상에 분산하여 저장하는 분산장부 기술로서 참여한 노드들에게 신뢰성을 보장해줄 수 있다. 하지만 블록체인에 참여하지 않은 노드에게는 인증이나 신뢰성을 제공하지 못하는데, 기본적으로 블록체인에 참여하는 노드는 일정수준 이상의 성능을 가져야 하기 때문에 사물인터넷, 마이크로그리드처럼 저전력, 저성능을 가지는 노드들이 다수 존재하는 환경에서 모든 노드가 참여하기는 어렵다. 본 논문에서는 마이크로 그리드 환경에서 블록체인에 참여하지 않은 노드에 대한 인증방안을 제시한다.

I. 서론

에너지 프로슈머의 유형은 전기요금 상계거래, P2P전력거래, 분산자원 중개시장을 이용한 도매거래의 3가지 유형으로 나눌 수 있다[1]. 이중 P2P 전력거래의 경우 계산, 정산이 복잡하고 수요자와 공급자간 매칭이 어렵다는 단점이 있어 꺼려져 왔다. 하지만 에너지 체인 기술이 도입됨에 따라 투명한 에너지 거래 시스템의 구축이 가능해지고 거래비용 감소 및 빠른 정산이 가능해졌다. 또한 스마트 계약을 통해 복잡한 거래 인증절차를 간소화 되어 새로운 P2P 전력거래 비즈니스 모델들이 출시되기 시작하였다[2].

그렇지만 현재 적용된 사례의 에너지 블록체인의 경우 기존 블록체인의 형태를 그대로 사용하기 때문에 검증과정이 비효율적이고 블록체인에 참여하지 않는 노드에 대해서는 보안성을 제공하지 못한다.

본 논문에서는 상기 언급한 문제점의 해결을 위해 마이크로 그리드 시스템에 적합한 블록체인의 적용 및 인증방안에 대하여 제시한다.

II. 관련연구

2.1 디바이스 인증 기술

디바이스 인증기술은 크게 공개키 기반 구조의 인증과 하드웨어 특성 기반 인증 둘로 나눌 수 있다.

하드웨어 특성 기반 인증은 하드웨어 기기 정보를 기반으로 디바이스를 인식하기 때문에 이기종간의 상호 연동 및 보안성이 취약하다는 보안문제점이 노출된다[3].

또한, 공개키 기반 구조(PKI)는 인증서를 사용하여 높은 보안 안전성과 확실한 인증이 가능하다. 하지만 인증정보에 하드웨어 정보가 포함되지 않아 복제, 위조 등에 취약하다는 단점이 존재한다.

2.2 블록체인 기반 인증 기술

블록체인은 분산원장 기술로 데이터를 중앙 서버가 아닌 시스템에 참여하는 모든 구성원들이 공유되는 데이터를 저장한다. 하지만 기술의 특성상 시스템 내에 발생한 정보를 모든 참여자가 확인할 수 있기 때문에 높은 투명성을 가지고, 서로 알 수 없는 노드

끼리 다른 참여자의 신뢰도를 확인시켜주는 합의(Concensus) 알고리즘을 통해 참여자들 및 저장데이터에게 높은 수준의 무결성을 포함하기 때문에 신뢰성을 가지는 보안기술로도 사용이 가능하다[4].

블록체인을 이용한 인증방식의 경우 사용자 모두가 데이터를 소유, 검증하기 때문에 기존 인증방식보다 강력할 뿐 아니라 위변조 방지가 가능하고 인증절차가 간소화된다는 장점이 있지만 블록체인에 직접적으로 참여한 노드에게만 보장이 가능한 한계점이 존재한다.

III. 제안연구

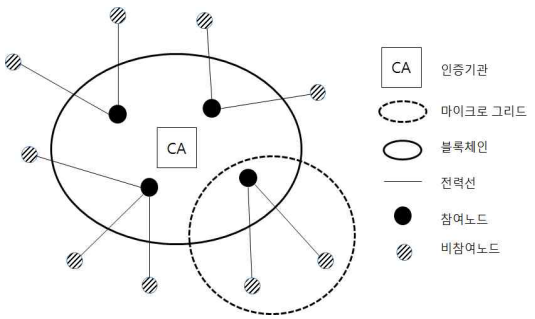
3.1 요구사항

본 논문에서 마이크로 그리드 환경에 블록체인을 적용하는 방식은 다음과 같은 몇 가지 요구사항이 필요하다.

1. 고유 값은 디바이스에게 초기에 부여된 값이며, 대표노드만이 알고 있다.
2. 각 대표노드가 가지고 있는 2개의 관리구조는 블록체인에 참여한 노드들에게 서로 공유되며, 마이크로 그리드에 새로운 디바이스를 추가하거나 변경하는 경우에 항상 최신의 데이터를 보존한다.
3. CA(Certification Authority)는 블록체인에 참여하는 노드들의 키를 관리하며 안전하게 보호한다.

3.2 설계

본 논문에서 제안하는 블록체인의 적용 형태는 다음의 [그림 1]과 같다.



[그림 1] 블록체인의 적용 형태

마이크로 그리드 내에서 하나의 대표 디바이스만 블록체인에 참여한다. 이 디바이스는 블록체인의 참여노드인 동시에 마이크로 그리드를 관리하는 역할을 가진다.

다음의 [표 1]은 대표노드가 마이크로 그리드 디바이스를 관리하기 위한 구조이다.

[표 1] 마이크로 그리드 관리 구조

No	ID	이름	고유값	hash
1	a1a1	sola sel	abc123	f1568d
2	a1b2	ESS	def456	123fd5
3	a1c3	Switch	bcd345	15cd83

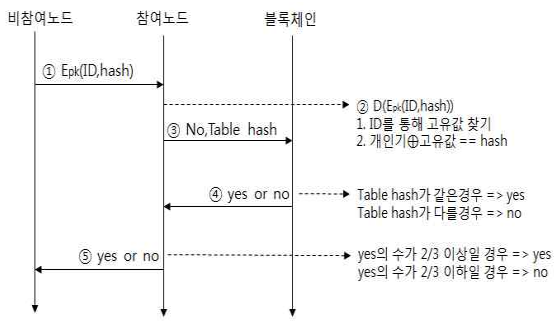
No는 참여노드가 디바이스를 관리하는 번호를 나타내고, ID필드에는 디바이스의 ID가 들어간다. 이름은 기기의 역할을 나타내며 고유값은 마이크로 그리드 내 구성원들이 공통적으로 각자 가지고 있는 변경이 불가능한 값이다. 해시값은 고유값과 대표노드의 개인키를 XOR 연산한 뒤 해시함수를 이용하여 구한 값이다. 또한 블록체인에 참여한 노드들은 다음의 [표 2]를 가지고 있어야 한다.

[표 2] 해시관리 구조

블록체인 ID	No	table hash
a1b2c3	1	a1a2a3
	2	c1c2c3
b2c3d4	1	a1b2c3
	2	b2c3d4
abc123	1	b2c3a1
	2	f5a1c3

블록체인 ID는 각 노드가 가지고 있는 지갑의 공개키 값이며 No는 해당 대표노드에서 인증을 요청하는 노드의 번호이다. table hash는 [표 1]에서 각 No의 데이터를 모두 이어붙인 값을 해시함수를 통해서 구한 별도의 해시 값이다.

3.3 인증방안



[그림 2] 블록체인을 이용한 인증

본 논문에서 제안하는 인증방안은 위의 [그림 2]와 같으며 다음의 순서로 수행된다.

① 인증을 원하는 노드 A는 대표노드에게 ID와 hash를 대표노드의 공개키로 암호화 하여 인증요청 메시지를 전달한다.

② 인증요청 메시지를 자신의 개인키로 복호화 한 뒤 [표 1]에서 ID에 맞는 고유값을 찾는다. 고유값과 개인키를 XOR연산하여 과정①의 hash값과 비교한다.

③ 두 값이 일치할시 [표 2]에서 해당 No와 ID에 해당하는 talbe hash를 블록체인에 브로드캐스팅 한다.

④ 각 노드들은 수신한 데이터를 자신의 테이블과 비교하고, 두 데이터가 일치할 경우 yes를 일치하지 않을 경우는 No를 송신한다.

⑤ 받은 yes의 수가 블록체인 참여 노드의 2/3 이상 일 경우 인증을 허가한다.

IV. 결론

본 논문에서는 마이크로 그리드 환경에서 하나의 대표노드가 블록체인에 참여하여 효율적으로 블록체인을 구성하는 방식에서 블록체인에 참여하지 않는 비참여노드에게 블록체인을 이용하여 인증하는 방안을 제시하였다.

본 논문에서 제시한 인증방식의 경우 인증을 요청하는 디바이스는 초기의 인증요청메세지의 송신에서 생기는 암호화 과정 한번을 제외하고는 별다른 연산이 요구되지 않아 저사양 디바이스에게도 모두 적용이 가능하며, 악의적인 노드가 대표노드가 가지고 있는 테이블을 공격해도 블록체인에 있는 참여노드의 2/3에 해당하는 데이터를 짧은 시간 내에 바꾸지 못한다면 이상을 감지할 수 있다는 장점이 있다.

ACKNOWLEDGMENT

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.20180005080012002, 마이크로그리드 보안 및 운영 효율성을 위한 블록체인 기반 임베디드 기기 및 플랫폼 개발)

[참고문헌]

[1] 김민경, 이윤혜, 에너지 프로슈머, 새 전력수급 주체로 분산자원 중개시장 이용해 활성화 필요, 2018

[2] 우청원, 에너지 블록체인 도입방안 연구, 2018.04

[3] 이성훈, 김광조, 블록체인을 이용한 스마트그리드 시스템의 기기 인증 방안, 2016

[4] 현무용, 김태훈, 김재희, 김종화, "스마트그리드 기기의 인증 및 키 관리 보안 동향", 2013. 10

5G 환경을 위한 블록체인 기반 분산형 이동성 관리 기술

김이진, 최형기*
성균관대학교

Blockchain-based Distributed Mobility Management for 5G Networks

Ejin Kim and Hyoung-Kee Choi
Sungkyunkwan University

요 약

이동성 관리 기술은 모바일 노드가 움직임에 따라 모바일 노드의 데이터 트래픽이 중앙 네트워크를 거치지 않고 네트워크 가장자리에서 처리되도록 하는 분산형 이동성 관리 기술을 통해 낮은 지연 시간과 신뢰성 높은 통신을 지향하는 5G 네트워크의 요구사항을 충족시키고자 한다. 그러나 핸드오버 시 발생하는 시그널링 메시지는 여전히 중앙 네트워크를 통해 처리되고 있으며, 이동성 관리 기술에서 사용하는 인증 방식은 비효율적이고 안전하지 않다. 본 논문에서는 이동성 관련 정보를 안전하고 효율적으로 관리하기 위한 블록체인 기반의 분산형 이동성 관리 기술을 제안한다. 제안하는 기법에서는 이동성 관련 정보를 네트워크 가장자리에서 위변조 불가능한 방식으로 공유하며, 노드간 수행되는 인증에 대한 보안성과 성능을 향상시킨다.

I. 서론

Fifth generation(5G) 네트워크에서는 증강 현실이나 자율 주행과 같이 실시간으로 처리해야 하는 모바일 애플리케이션에 대한 수요가 증가할 것으로 예상된다. 이러한 흐름과 일치하게 이동성 관리 기술 또한 분산된 방식으로 구조로 발전하고 있다. 중앙 집중형 이동성 관리 기술로 잘 알려진 MIPv6(mobile IPv6) [1]를 변형하거나 확장하는 방식으로 5G의 새로운 요구사항을 만족시키는 이동성 관리 기술에 대한 연구가 이루어지고 있다. 분산형 이동성 관리(Distributed Mobility Management: DMM) 기술 [2][3]은 응답 지연 시간을 축소시키기 위해 중앙에서 관리되던 이동성 라우팅 기능을 제공하는 데이터 평면의 트래픽을 액세스 네트워크 단으로 분산한다. 이를 통해, CMM에서 발생하는 단일 장애 지점이나 최적화되지 않은 라우팅 경로(suboptimal routing) 등의 문제를 해결한다.

DMM은 분산된 구조의 데이터 평면과 달리 위치 관리 기능을 제공하는 제어 평면에 대해서는 여전히 중앙 집중형 구조를 가진다. 중앙 집중형 제어 평면으로 인해 이동성 관리는 중앙 네트워크를 반드시 거쳐야 하는 문제가 발생한다. 본 논문에서는 5G에서의 이동성 관리를 위해 수행되는 DMM의 지연을 최소화하는 것을 목표로 한다. 이를 위해, 블록체인을 기반으로 제어 평면에 대해 분산형 구조를 가지며 종단간 인증을 효율적으로 수행하는 이동성 관리 기술 제안하고자 한다.

II.문제 정의

모바일 노드(Mobile Node: MN)가 새로운 네트워크로 이동하면 MN이 연결되어 있는 액세스 라우터(Serving MAAR: S-MAAR)는 MN의 모든 활성화된 세션의 앵커 MAAR

* 교신저자(Corresponding author)

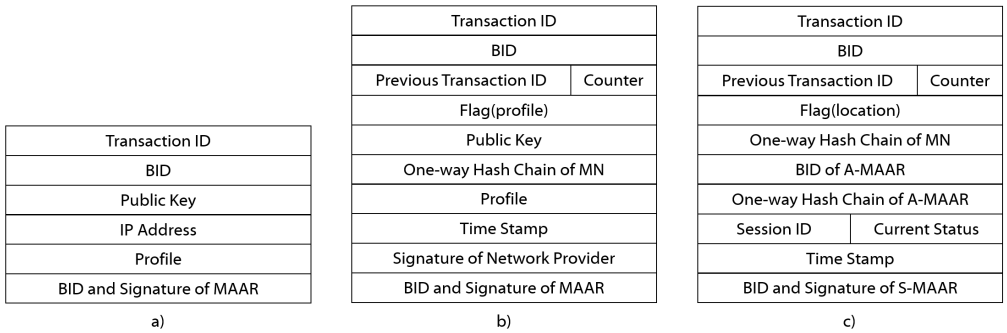


Fig. 1. Transaction format of proposed DMM: a) Transaction format about MN(profile); b) Transaction format about MN(location); c) Transaction format about MAAR.

(Anchored MAAR: A-MAAR)을 파악하여 바인딩 업데이트를 진행한다. A-MAAR은 활성화되어 있는 세션이 종료될 때까지 연결을 유지한다. 네트워크 기반 분산형 이동성 관리 기술에서는 중앙 이동성 데이터베이스(Central Mobility Database: CMD)라는 중앙 데이터베이스에 MN의 이동성 관련 정보를 저장한다. S-MAAR은 CMD에게 활성화된 세션에 대한 쿼리를 전송하여 A-MAAR의 위치를 파악한 후, 바인딩 업데이트 절차를 수행한다. 또한, S-MAAR은 MN이 새로운 세션을 열 때 BCE를 CMD에 등록하여 MN의 이동성 관련 정보를 업데이트해야 한다. CMD는 네트워크 내의 모든 MAAR이 접근하기 용이하도록 네트워크 중앙에 위치한다. MN의 이동성 관련 정보를 파악하기 위해 CMD에 쿼리하는 과정은 CMD까지의 거리에 따라 바인딩 업데이트 절차에 추가적인 지연을 야기한다. 또한, 중앙 집중 형태의 제어 평면인 CMD는 단일 장애 지점에 취약하며, CMD의 보안성이 곧 DMM 전체의 보안성과 동일하게 되어 CMD가 공격의 대상이 된다.

완전 분산형 N-DMM(fully distributed N-DMM) 기술에서는 이동성 관리를 위한 중앙 집중 형태의 네트워크 노드를 제거하고 데이터 평면과 함께 제어 평면의 역할을 네트워크 가장자리로 분산시킨다. 완전 분산형 N-DMM 기술을 도입하여 부분 분산형 N-DMM (semi-distributed N-DMM)에서 발생할 수 있는 위협을 차단하고, DMM의 성능을 향상시킬 수 있다.

III.블록체인 기반 DMM

본 연구에서는 [5]에서 제안한 방식과 같이 중앙 집중 형태로 이동성 관련 정보를 관리하는 CMD의 기능을 블록체인을 통해 MAAR에 재배치시켜 완전 분산형 N-DMM 구조를 구성하였다. 분산 형태로 데이터를 공유함에 있어 제 3의 기관의 개입없이 네트워크 제공자 간의 신뢰 관계를 효율적으로 구축할 수 있다는 장점을 가진다. 또한, 중앙을 거쳐 진행되던 이동성 관리와 노드 간 인증을 인증 주체인 MN의 가까이에서 수항함으로써 지연을 최소화하였다.

블록체인을 통해 공유하고자 하는 데이터는 MN 및 MAAR에 대한 외부로 노출되어서는 안되는 정보로, MAAR 간에만 공유되어야 한다. 블록체인 네트워크는 원장에 대한 읽기 및 쓰기 권한이 제한되는 허가형 블록체인으로 구성한다.

본 논문에서 제안하는 바인딩 업데이트 프로토콜은 MN의 이동성 정보를 블록체인으로 공유하는 구조이다. MN이 다른 네트워크로 이동하는 속도가 MN의 이동성 정보에 대한 트랜잭션이 원장에 탑재되는 속도보다 빠를 경우, 바인딩 업데이트 절차가 정상적으로 진행될 수 없다. [4]에서는 일반적인 MN의 이동 속도를 0 km/h(kilometer per hours)에서 120 km/h로 정의한다. Tendermint와 같이 속도를 개선한 합의 알고리즘은 약 10,000 TPS(Transaction Per Seconds)의 속도를 지원한다. 이러한 합의 알고리즘을 사용할 경우, MN의 이동 속도보다 빠르게 트랜잭션을 처리하는 것이 가능하다. 따라

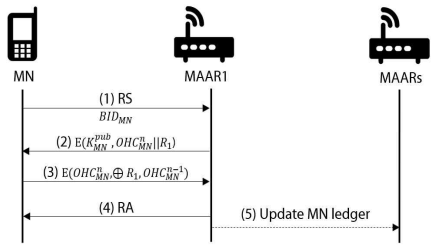


Fig. 2. Binding Update Protocol for the MN without On-going Sessions.

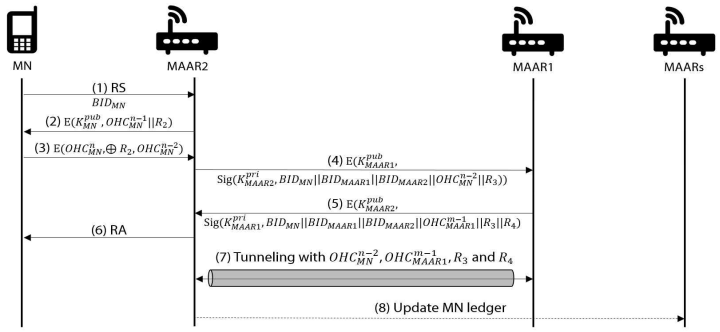


Fig. 3. Binding Update Protocol for the MN with On-going Sessions.

서 MN의 이동속도보다 빠르게 트랜잭션을 처리할 수 있는 속도를 가지는 합의 알고리즘을 사용해야 한다.

3.1 블록체인 네트워크의 구성요소

제안하는 블록체인 기반 DMM에는 MAAR이 관리하는 두 종류의 허가형 블록체인 원장이 존재하며, 각 원장에 탑재되는 트랜젝션은 Fig. 1과 같은 형식을 가진다.

• MAAR 원장: DMM 내에서 MN의 이동성을 관리하는 MAAR에 대한 정보가 탑재된 원장으로, MAAR 간의 인증과 MAAR에 대한 정보를 공유하기 위해 사용된다. MAAR이 블록체인 네트워크에 참여 노드로 추가될 때나 MAAR의 정보를 변경할 할 때 업데이트된다.

• MN 원장: MN이 통신 사업자에 가입할 때 등록되어 인증에 사용되는 정보와 바인딩 업데이트 절차를 통해 갱신되는 이동성 관련 정보를 포함한다.

3.2 블록체인 네트워크의 동작 방식

모든 MAAR이 트랜잭션을 생성할 수 있는 권한을 가진다. 연결을 시도하는 MN에 대해 바인딩 업데이트 절차를 수행한 후, MN의 location 트랜잭션을 생성한다. 합의 알고리즘에 따라 선출된 검증 노드가 아직 검증되지 않은 트랜잭션을 수신하여 검증을 수행한다. 정상적으로 검증이 이루어진 트랜잭션을 모아 블록을 생성하고, 블록을 체인에 탑재할 것인지에 대해 다른 검증 노드와 합의를 수행한다. 검증 노드에 문제가 발생하는 경우, 검증 노드 선정 방식을 통해 MAAR 중 새로운 검증 노드를 선발한다.

검증 노드는 트랜잭션에 담겨있는 S-MAAR의 서명과 MN의 일방향 해시 체인이 올바른 해시 체인을 이루고 있는지 파악한다. MAAR에 대한 서명은 MAAR 원장을 통해 올바른 MAAR의 서명인지를 파악할 수 있으며, MN의 일방향 해시 체인에 대한 검증은 이전 트랜잭션 ID 필드를 통해 효과적으로 탐색할 수 있다.

3.3 블록체인 기반 DMM의 바인딩 업데이트

제안하는 블록체인 기반 DMM은 Fig. 3과 Fig. 4와 같은 방식으로 바인딩 업데이트를 수행한다. 제안하는 바인딩 업데이트 절차를 통해 다음과 같은 보안 이슈를 만족할 수 있다.

• 세션 소유권: 제안하는 DMM에서는 바인딩 업데이트 시 S-MAAR이 일방향 해시 체인을 통해 MN에 대한 인증을 수행한다. MN 원장에 탑재된 MN의 공개키 K_{MN}^{pub}을 이용하며, 재전송 공격을 막기 위하여 랜덤값 R_1(Fig. 2-(2)) 또는 R_2(Fig. 3-(2))을 함께 전달한다. 합법적인 MN만이 S-MAAR이 생성한 R_1 또는 R_2를 확인할 수 있으며, 이 값을 통해 이전 단계의 일방향 해시 체인값을 안전하게 검증한다. S-MAAR이 수신한 일방향 해시 체인 값은 A-MAAR에게 전달된다(Fig. 3-(4)). A-MAAR은 기존에 유지하던 자신과의 활성화 세션에 참여하던 MN인지 검증하여 안전성을 보장한다. S-MAAR은

MN의 location 트랜잭션에 포함된 A-MAAR의 일방향 해시 체인을 확인하여 MN과 세션을 유지하던 MAAR임을 인증한다(Fig. 3-(5)).

• 인증되지 않은 로그(rogue) MAAR 탐지: MN 원장을 공유하는 합법적인 MAAR만이 OHC_{MN} 값을 알고 있으므로 블록체인 네트워크에 참여하지 않는 노드는 MN에게 올바른 OHC_{MN}을 전달할 수 없어 바인딩 업데이트 절차를 진행하는 것이 불가능하다. 또한, 바인딩 업데이트 절차에서 MAAR 간의 통신은 MAAR 원장에서 공유되는 MAAR 공개키를 통해 이루어진다(Fig. 3-(4), (5)). 이를 통해 네트워크 제공자 간 별도의 인증 프로토콜 없이 로그 MAAR에 대한 탐지를 수행할 수 있다.

• 이동성 정보의 무결성: MN의 이동성 정보는 바인딩 업데이트 이후 S-MAAR에 의해 MN 원장에 추가된다. 원장에 탑재된 MN의 이동성 정보는 블록체인의 보안성에 의해 변조가 불가능하다. 또한, location 트랜잭션에는 MN의 일방향 해시 체인이 포함되므로 S-MAAR은 MN이 실제로 연결되어 있지 않은 상태에서 단독으로 MN의 이동성 정보에 대한 가짜 트랜잭션을 생성하는 것이 불가능하다.

• MAAR 간의 효율적인 암호화 통신: 바인딩 업데이트 절차 이후 S-MAAR과 A-MAAR 간의 통신은 OHC_{MN}과 OHC_{MAAR1}을 통해 도출한 세션키를 통해 암호화된다. OHC_{MN}과 OHC_{MAAR1}은 MN 원장에 탑재되는 값으로, 다른 MAAR이 두 MAAR 간의 암호화 통신을 확인할 수 있는 문제가 발생한다. S-MAAR과 A-MAAR이 인증값을 주고받을 때(Fig. 3-(4), (5)) 랜덤값 R_3, R_4를 함께 전달하여 이를 세션키를 도출하는 입력값으로 사용하는 방식으로 다른 MAAR에게 세션키가 노출되는 문제를 해결하였다. IPSec과 같은 별도의 암호화 프로토콜을 사용하지 않고 추가 비용을 최소화하는 방식으로 암호화 통신을 수행한다.

IV.결론

본 논문에서는 MN의 인증과 이동성 정보를 블록체인 기반으로 수행하는 DMM 기술을 제안하였다. 제안하는 DMM은 네트워크 제공자 간에 자원을 공유하는 환경에서 추가적인 보안 메커니즘없이도 MN의 이동성 정보에 대한 무결성을 보장한다.

제안하는 DMM에는 몇 가지 한계점이 존재한다. MN의 이동성 정보를 모든 MAAR이 공유해야 하므로 저장 공간의 낭비와 부족에 관한 문제가 발생할 수 있다. 또한, 사용하는 합의 알고리즘에 의존하여 DMM의 성능이 결정된다. MN의 이동 속도가 합의 알고리즘보다 빠를 경우 바인딩 업데이트 절차가 정상적으로 진행되지 않는다. 속도와 보안성 면에서 안정적인 합의 알고리즘과 저장 공간의 낭비에 대한 연구를 추가적으로 진행하여 제안하는 DMM의 성능을 향상시키고자 한다.

ACKNOWLEDGMENT

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2016R1D1A1B03936211).

[참고문헌]

[1] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in IPv6," IETF RFC 6275, Jul. 2011.

[2] D. Liu et al., "Distributed mobility management: Current practices and gap analysis," IETF RFC 7429, Jan. 2015.

[3] J. Lee et al., "Distributed IP mobility management from the perspective of the IETF: Motivations, requirements, approaches, comparison, and challenges," IEEE Wireless Communications, vol.20, no.5, pp.159-168, Nov. 2013.

[4] 5G Forum, "5G vision, requirements, and enabling technologies (v.2.0)," Mar. 2016.

[5] V. Sharma et al., "Secure and energy-efficient handover in fog networks using blockchain-based DMM," IEEE Communications Magazine, vol.56, no.5, pp. 22-31, May 2018.

효율적 바이너리 코드 재조립 시스템의 설계*

최지원, 차상길
한국과학기술원

On Designing an Efficient Binary Reassembler

Jiwon Choi, Sang Kil Cha
KAIST

요 약

바이너리 계측(instrumentation)은 악성코드 분석이나 취약점 탐지 등에 있어 핵심적인 기술이다. 최근에는 정적으로 역어셈블(disassemble)된 바이너리에 계측 코드를 삽입한 후 재조립(reassemble)하는 기술이 대두되고 있으나, 대부분 계측의 정확도 또는 효율이 현저히 낮아지는 문제를 가지고 있다. 본 논문에서는 최신 바이너리 재조립 기술들이 갖는 문제점에 대해 고찰하고, 정확도와 효율성 모두를 높일 수 있는 새로운 바이너리 재조립 기술을 제안한다.

I. 서론

소프트웨어의 동작을 관찰하고 측정하기 위한 기술인 바이너리 계측(instrumentation)기술은 악성코드 분석이나 메모리 오류 탐지, 바이너리 취약점 패치 등 다양한 소프트웨어 보안에 필요한 핵심적인 기술이다[1].

바이너리 계측 방식은 크게 동적 계측과 정적 계측 방식으로 나뉜다. 동적으로 코드 캐시를 변형하는 동적 계측 방식은 정적 계측에 비해 속도가 느리다는 단점이 있지만 동적으로 생성되는 코드를 계측할 수 있다. 한편, 정적 계측 방식은 속도가 빠르긴 하지만 동적 코드를 다룰 수 없을 뿐 아니라 역어셈블의 부정확성에 의해 계측의 정확도가 낮다는 단점을 갖는다. 최근에는 이러한 정적 계측의 문제를 개선한 Superset Disassembly 기술[3]이 소개되었는데, 모든 메모리 접근을 동적으로 계측함으로서 정확도가 획기적으로 개선된 반면, 속도 및 공간 효율이 크게 떨어지는 문제를 갖는다.

본 논문에서는 이러한 정적 바이너리 계측 기술이 갖는 문제점들에 대해 진단하고, 이를 경감하기 위한 기술들을 제안한다. 좀 더 구체적으로는 정적 값 분석(value analysis)과 동적 메모리 계산 방식을 결합한 하이브리드 형태의 심볼화(symbolization)를 개발하여, 이를 통해 실제 바이너리 코드를 안정적으로 재조립하는 기술을 제안한다.

우리는 제안하는 방법의 효율성을 검증하기 위해 GNU coreutils의 108개의 스트립(strip)된 바이너리들에 대해 재조립 실험을 진행하였다. 그 결과 우리는 정적 계측이 갖는 속도상의 우월성을 유지한 채 공간 오버헤드 또한 평균 300배 가까이 줄일 수 있었다.

II. 관련 연구

정적 바이너리 계측기술은 최근 들어 바이너리 재작성 기술을 통해 그 실효성이 높아지는 추세이다. 바이너리 재작성의 가장 큰 어려움은

*이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.B0717-16-0109, 바이너리 코드 분석을 통한 자동화된 역공학 및 취약점 탐지 기반기술 개발).

역어셈블된 코드의 재조립시 코드와 데이터들이 재배치된다는 점이다. 즉, 원본 바이너리에서 코드나 데이터가 특정 위치에 고정 할당된 경우 이를 재조립한 바이너리에서는 실행 시 해당 코드나 데이터를 참조할 수 없는 문제가 발생한다. 과거 연구에서는 이러한 문제를 효과적으로 핸들링하기 위해 크게 두 가지 방법론이 제시되어 왔는데, 메모리 심볼화[2][4]와 전역 매핑 테이블의 운용[3]이 이에 해당한다.

메모리 심볼화는 바이너리 내부에 명시된 값이 메모리 참조 값인지 여부를 다양한 휴리스틱을 사용해 판단한다. 그리고 메모리 참조 연산들에 대해서는 메모리 주소 대신 심볼(라벨)을 활용하여 해당 영역을 참조할 수 있도록 변경한다. 만약 심볼화 작업이 완벽할 수 있다면, 이후 코드가 추가되거나 삭제되어 재배치가 일어나더라도, 정상적인 동작이 가능하다.

전역 매핑 테이블 운용 방식은 Superset Disassembly[3]에서 처음으로 제안되었으며, 모든 메모리 참조 명령어가 실행되기 이전에 전역 매핑 테이블을 통해 원본 코드의 위치를 참조하도록 하는 방식이다. 이 방법은 근원적으로 메모리 재배치 문제를 해소하지만, 모든 메모리 참조를 계측함에 따라 높은 런타임 오버헤드를 갖는다. 또한 추가적인 테이블과 원본 역어셈블 이미지에 대한 메모리를 요구하므로, 최소 2배에서 많게는 200배에 이르는 공간 오버헤드 증가를 갖는다.

III. 기존 시스템의 한계점

우리 시스템을 소개하기에 앞서 기존 시스템의 한계점을 크게 두 가지로 정리하였다.

첫째, 심볼화 기반 바이너리 재조립 기술은 기본적으로 PIE 바이너리(Position Independent Executable)를 다루지 못한다. 왜냐하면 바이너리 실행 이전에는 계산된 코드 포인터의 주소를 정적으로 결정할 수 없기 때문이다. 따라서 과거의 UROBOROS[2]나 Ramblr[4]의 경우 상용 바이너리에 적용되기 어려운 한계점을 가지고 있었다. 실제 Ubuntu 18.04 기준 /usr/bin의 98%가 PIE 바이너리이다.

둘째, 기존의 기술에서는 코드 재조립 이후에 발생할 수 있는 코드 및 데이터의 정렬(alignment) 문제를 고려하지 않았다. 코드 실행에 있어서 캐시라인에 정렬되도록 코드를 배치하는 편이 성능 상 유리하며, 특히 SIMD 명령어의 경우에는 정렬이 고려되지 않는다면 명령어의 실행 자체가 불가능한 경우가 발생한다. 따라서 원본 바이너리의 정확도와 효율을 유지하기 위해서는 재조립된 바이너리에서도 정렬을 보존해야 한다.

IV. 시스템 설계 및 구현

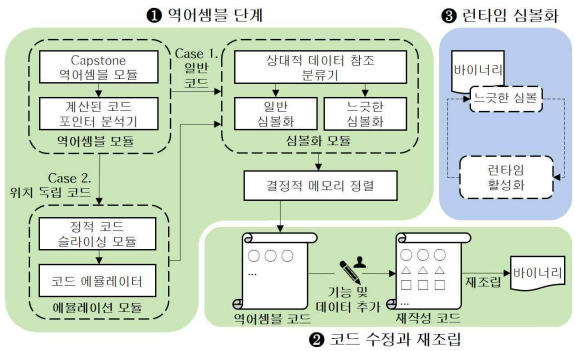


Fig. 1 System overview

우리의 시스템은 바이너리 파일을 입력받아 역어셈블과 재조립 과정을 거친 새로운 바이너리 파일을 생성한다. Fig. 1은 시스템의 전체적인 구조를 나타내는데, 앞 절에서 말한 두 가지 문제를 각각 해소할 수 있는 기술들이 각 단계별로 적용된다. 먼저 PIE 바이너리 문제의 경우 정적 코드 슬라이싱을 통해 코드 포인터의 주소를 계산하게 되며 (섹션 4.1), 미처 정적으로 다 계산되지 못한 심볼들은 느긋한 심볼화 과정을 통해 동적으로 메모리 주소를 해소하는 과정을 거치게 된다 (섹션 4.2). 또한 코드 정렬 문제의 경우에는 원본 바이너리의 정렬을 최대한 보존하는 결정적 메모리 정렬 기술을 사용하여 해소한다 (섹션 4.3).

4.1 정적 코드 슬라이싱

정적 코드 슬라이싱 모듈은 전위 슬라이스(forward slice)방식을 이용해 결정 가능한(deterministic) 레지스터 값을 분석한다. 슬라이

싱은 PIE 바이너리에서 실행 주소를 리턴하는 get_pc_thunk를 기점으로 시작되며 선택적으로 추출된 코드 슬라이스에서 레지스터 실제 값을 evaluation한다. 이때 결정 가능한 값들만을 계산하며, 계산된 값이 메모리 참조 값일 경우에 한해 이를 심볼화한다. 물론 정적으로 결정되지 못한 심볼들이 존재할 수 있는데, 이는 다음 절에서 설명하는 느긋한 심볼화 과정을 통해 동적으로 메모리 주소를 해소한다.

4.2 느긋한 심볼화

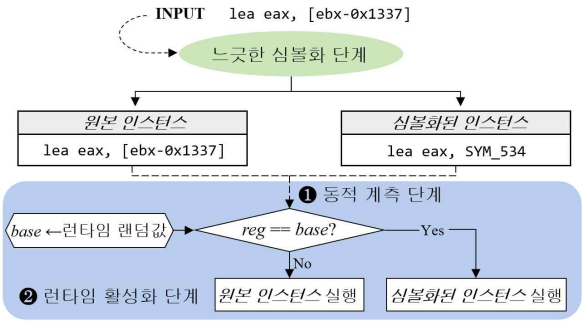


Fig. 2 Lazy symbolization

PIE 바이너리에서는 데이터 참조 시 바이너리의 특정 지점을 기준으로 하는 상대적인 주소를 사용한다. 기준점의 위치는 실행 시 랜덤하게 결정되기 때문에 기존의 기술이나 앞 절에서 논의한 정적 슬라이싱을 활용하더라도 추출이 불가능하다. 우리는 느긋한 심볼화(lazy symbolization) 기술을 통해 이를 해소한다.

먼저, 역어셈블된 코드 중에서 메모리 포인터 관련 명령어(LEA등)를 추출한다. 해당 명령어들은 데이터를 접근하기 위한 주소의 계산일 수도 있고, 단순 메모리 계산 명령일수도 있다. 하지만 후자를 심볼화하는 경우 잘못된 데이터 값을 사용하게 되는 문제가 존재한다. 따라서 우리는 동적 계측을 이용하여 두 경우를 판별한다. GCC 컴파일러의 경우 메모리 연산의 베이스 메모리 주소로 항상 고정된 위치(예를 들면 GOT)를 사용하는 점에 착안하여 동적으로 베이스 레지스터의 값을 확인하고 해당 값이 알려진 베이스 포인터 값이면 메모리 주소 연산으로 판별하도록 하였다.

따라서 Fig. 2에서 보는 바와 같이 메모리 관련 명령어들에 대한 동적 계측 코드를 추가적으로 삽입한다. 동적으로 계산된 베이스 주소가 GOT 등의 고정된 위치 값을 갖는 경우에는 심볼화된 인스턴스를 실행하며, 그 외에는 원본 인스턴스를 실행한다. 심볼화된 인스턴스가 사용할 심볼의 주소를 계산하기 위해 정적으로 베이스 레지스터의 값을 GOT주소라 가정하고 임의의 심볼을 부여한다. 단, 이때 가정이 잘못되었을 경우에도 의미 없는 심볼이 추가될 뿐 코드의 시멘틱을 변화시키지 않기 때문에 재조립된 바이너리는 정상적으로 동작하게 된다.

4.3 결정적 메모리 정렬

코드 재조립 이후에 코드 및 데이터 정렬이 보존되지 않는다면 바이너리의 성능이 떨어질 뿐 아니라 실행 자체가 불가능해질 수 있다. 이를 해소하기 위한 방법으로는 SIMD 명령어에서 참조하는 메모리 주소를 모두 조사하는 것이지만 이는 바이너리의 모든 실행 흐름을 복원할 수 있어야만 가능하다는 문제점이 있다.

결정적 메모리 정렬 기술은 원본 바이너리에 존재하는 SIMD명령어의 연산 대상 주소를 활용하여 해당 데이터를 정렬한다. 이때 정렬의 크기는 가장 큰 32바이트에서부터 2바이트에 이르기까지 내림차순으로 원본 주소의 정렬 크기를 근사하여 결정하는데, 이를 통해 정렬이 요구될 가능성이 있는 모든 데이터들을 정렬할 수 있으며, Superset Disassembly[3] 방식에 비해 적은 공간 오버헤드로도 구현이 가능하다.

4.4 구현

본 논문에서 제안하는 시스템은 Python을 사용하여 약 2,500줄로 구현하였다. 우리의 시스템은 크게 바이너리의 메타데이터 분석을 위한 Pyelftools, 바이너리 역어셈블을 위한 Capstone, 그리고 재조립시 기계어의 유효성 검증을 위한 Keystone 모듈로 구성된다.

V. 평가

제안하는 시스템의 효과를 확인하기 위해 바이너리 데이터 셋에 대한 실험을 수행했다. 실험은 8개의 Intel i7-7700K 4.20GHz 코어 및

4GB RAM을 갖는 Ubuntu 16.04 32bit 가상 시스템에서 진행하였다.

5.1 데이터 셋(Data Set)

제안하는 시스템의 성능 측정을 위해 GNU coreutils 8.30.10 에 포함된 108개의 스트립된 리눅스용 바이너리에 대해 테스트를 진행했다. 상용 바이너리에서의 다양한 컴파일 특성을 나타내기 위해 우리는 O0부터 O3에 이르는 4단계의 최적화 단계별로 생성했으며, 그 결과 총 432개의 바이너리를 얻을 수 있었다.

5.2 바이너리 동작의 정확도

우리는 재조립된 바이너리의 정확도를 검증하기 위해 다양한 입력을 정상적으로 실행할 수 있는지 테스트하였다. 이를 위해 coreutils에 포함된 608개의 유닛테스트 스크립트를 이용하였으며, Table 1의 두 번째 열에서 확인할 수 있듯이 모든 재조립된 바이너리에서 100%의 실행 정확도를 보였다.

최적화 단계	정확도	전체 크기 (원본 바이너리)	전체 크기 (재조립 바이너리)	증가율
-O0	100%	5009592 bytes	6040456 bytes	20.57%
-O1	100%	4751072 bytes	5315480 bytes	11.87%
-O2	100%	4966708 bytes	5550780 bytes	11.75%
-O3	100%	7016680 bytes	7524020 bytes	7.23%

Table 1 Spatial overhead

5.3 재조립된 바이너리의 크기 증가

Table 1의 3-5열에서 볼 수 있듯, 재조립된 바이너리의 크기 증가는 원본 바이너리에 비해 평균 12.85% 수준이었다. 최신 기술인 Superset Disassembly[3]에서 최소 2배의 사이즈 증가가 있었던 것을 감안할 때, 크기증가분이 획기적으로 감소한 것을 확인할 수 있다.

5.4 실행 오버헤드

재조립된 바이너리의 실행 오버헤드를 측정하기 위해 우리는 각 바이너리별로 100회에 걸쳐 coreutils test suite를 실행하였고, 그 결과를 Fig. 3에 정리하였다. 네 가지 최적화 수준에 대하여 평균 0.0788%의 오버헤드를 보였으며, 이는 무시할 수 있는 수준의 오버헤드이다. O1이나 O2 바이너리들에 대해서는 오히려 오버헤드가 감소하는 경향이 나타났는데, 이는 우리의 정적 코드 슬라이싱 기술을 통해 일부 간접 분기문이 직접 분기문으로 변화하면서 생긴 현상으로 수동 검증을 통해 해당 변환이 정상적임을 입증하였다.

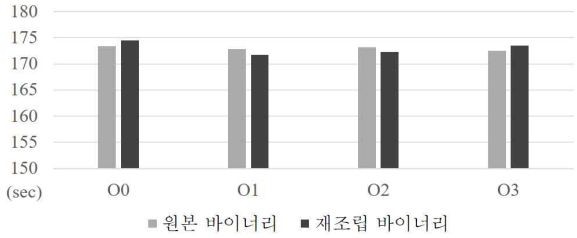


Fig. 3 Performance overhead

		UROBOROS [2]	Ramblr [4]	MULTIVERSE [3]	Suggested System
역어셈블과 재조립 가능	일반 바이너리	O	O	O	O
	PIE 바이너리	✗	✗	O	O
	라이브러리	✗	✗	O	O
기능 수정 데이터 수정 모두 가능	일반 바이너리	*partially*	*partially*	O	O
	PIE 바이너리	✗	✗	O	O
	라이브러리	✗	✗	O	O
w/o 모든 메모리 참조를 계측		O	O	✗	O
w/o 전역 맵핑 테이블 필요		O	O	✗	O
w/o 모든 분기문 재작성 필요		O	O	✗	O
x2 미만의 공간 오버헤드		O	O	✗	O

※ *partially* : SIMD 명령어가 데이터참조를 사용하지 않는 경우 부분적으로 가능

Table 2 Compare with existing system

VI. 결론

본 논문에서는 정확도와 효율성 모두를 높일 수 있는 바이너리 재조립 기술을 제안하였다. 끝으로 우리 시스템의 우수성을 Table 2를 통해 정리하였다.

[참고문헌]

[1] 차상길, "소프트웨어 보안과 바이너리 분석", 정보과학회지 제36권 제3호(통권 제346호), pp. 11-16, 2018.

[2] S. Wang, P. Wang, "Reassembleable Disassembling", in proc. of the USENIX security, 2015.

[3] E. Bauman, Z. Lin, "Superset Disassembly :Statically Rewriting x86 Binaries Without Heuristics", in proc. of the NDSS 2018.

[4] R. Wang, Y. Shoshitaishvili, "Ramblr : Making Reassembly Great Again", in proc. of the NDSS, 2017.

An effective way to identify similar functions by performing pre-processing for binary analysis

Yungi Jo[1], Jimin Kim[2], Cheongmin Ji[2] and Manpyo Hong[1*]

Dept of Cyber Security, Ajou University[1]

Dept. of Computer Engineering, Ajou University[2]

ABSTRACT

Comparing similarity between binary functions is essential for identifying malicious code, 1-day vulnerability detection, and reverse engineering analysis of binary files with symbol table removed. Recently, Studies of analyzing binary similarity have been focused on How to functions are similar in semantic aspect when binaries have syntactically different. Our experimental results show the ratio of false positive increases if there are differences in compile options between binaries. In this paper, we propose a new method to reduce the structural differences between functions generated by compiler options and code obfuscation techniques through binary pre-processing in order to solve the problem and improve the results of similarity identification method of existing studies.

Keywords: Binary diffing, Binary similarity, Reverse engineering

I. Introduction

Binary function similarity comparison reduces the time required to analyze binary executable by identifying similar functions or commonly used library functions in a target binary during reverse engineering. It can be used to detect malicious application and find vulnerable codes. However, there are many difficulties in identifying a binary function through binary similarity analysis. In a similarity analysis task, syntactic information is often used as an element to compare functions. But the syntactic information is extracted at function's instruction level. So syntactic information is more focused on opcodes of functions rather than overall function features. The semantic information related to characteristic of the function can also be used to overcome the syntactic information's limitation. But there are many difficulties to extract the semantic information from functions. For example, what information can be extracted? and how? So recent studies have focused on the way to extract more comprehensive syntactic information or easier and more accurate semantic information[1, 2, 5].

In this paper, we propose a method to increase recall and reduce false-positive of the existing diffing algorithms and heuristics by removing predictable changes in advance. The predictable changes include the change of instruction assignment register, instruction relocation and branch inversion. They occur at the compilation stages. Different compile options cause these changes and come as an obstacle in the similarity analysis task.

The proposed method is composed of two stages. First, we collect patterns from instructions that do not affect the operation of functions and inserted during the compilation. Second, we remove the instructions matched by the pattern

In this paper, we use the Diaphora[3] as a similarity analysis tool to verify the proposed method. The tool has many verified algorithms and heuristics.

II. Related work

As mentioned in the introduction, existing studies are divided into two classes in terms of analyzing the similarity with the syntactic information and the semantic information.

If architectures of two binaries to be analyzed is different, there is a difference between the opcodes used. Even if binaries are executed on the

MOV	register	Canary value
MOV	?	register

Fig. 1. SSP pattern for initializing canary value part (clang4.0, i386)

MOV	register1	Canary value
MOV	register2	?
CMP	register1	register2
JNZ	?	

Fig. 2. SSP pattern for checking canary value part (clang4.0, i386)

same architecture with same operations, instructions can be different because of various compilation environments. Therefore, when analyzing similarity between functions based on syntactically acquired information, the similarity analysis does not perform well due to the fundamental differences in their syntax[4]. In order to solve this problem, several studies have focused on the semantic information[1, 6, 7, 8].

III. Pre-processing task

3.1 Collecting the pattern of instruction

There are some predictable changes to instructions inserted by compile option. We find out which ones are inserted instructions and eilminate differences caused by inserted instructions. To find the differences, patterns of the inserted instructions are used. We compare the binaries compiled with different options from the same source code and collect the instruction blocks of differences between two binaries. The collected instructions are not always same even if they are inserted through the same option, because each addresses or variables they use are different. So we substitute each register operands into simplified notation such as register1, register2 and etc. And using this pattern, we can find instructions inserted at the compilation steps. For evaluation, we choose SSP(Stack Smashing Protector) and BCF(Bogus Control Flow) as compile option that interfere similarity analysis.

3.1.1 Stack smashing protector

SSP option is used for preventing stack overflow attacks. Instructions performing the SSP option are inserted at compilation step. The instructions set a specific value, called canary, at the bottom of the function stack frame. Next, functions' argument and local variables are moved under canary value. If stack overflow occurs, canary value will be changed. Before the function ends, the application can detect whether stack overflow had occurred or not. Figure 1 and 2 are pattern instructions for SSP.

```
# x : 1 byte
# y : random integer
if y < 10 || (x-1)*(x & 1) == 0 do
        break or jump
```

Fig. 3. Bogus control flow pass example(always true case)

MOV	register1 ,	data segment1
MOV	register2 ,	data segment2
MOV	register3 ,	register1
SUB	register3	1
IMUL	register1	register3
AND	register1	1
CMP	register1	0
SETZ	bl	
CMP	register2	0x0A
SETL	bh	
OR	bl	bh
TEST	bl	1
JNZ	?	
JMP	?	

Fig. 4. BCF pattern example (clang4.0, i386)

3.1.2 Bogus Control Flow

Bogus control flow is an obfuscation technique that changes control flow by inserting a conditional branch that becomes unconditionally true or false. An example of a conditional expression is shown in Fig. 3. When $X = 1$, $X - 1 = 0$ is always satisfied. When $X = 0$, $X \ \& \ 1 = 0$ is always satisfied. Therefore, $(X-1) \bullet (X \ AND \ 1) = 0$ conditional expression always has true, regardless of X value. Figure 4 is an example of pattern for BCF.

3.2 Substitution of code block

In order to prevent the inserted instructions affecting similarity analysis result, deleting the instruction is not the best method because it damages the function, data reference and branch instruction related to the memory address. Rather than deleting instructions, we replace instructions with NOP instructions.

SSP	total	before pre-processing				after pre-processing				variation	
		TP	precision	recall	f-score	TP	precision	recall	f-score	TP	f-score
cpio	347	199	0.995	0.572	0.727	263	1	0.758	0.862	64	0.135
ftp	243	159	0.994	0.653	0.788	205	0.995	0.847	0.915	46	0.127
hello	84	22	0.955	0.253	0.4	39	0.974	0.458	0.623	17	0.223
ping	175	89	0.989	0.511	0.674	135	0.993	0.776	0.871	46	0.197
sed	246	146	0.986	0.598	0.745	217	0.973	0.904	0.937	71	0.192
tar	1167	558	0.988	0.481	0.647	858	0.968	0.753	0.847	300	0.2
telnet	301	177	0.994	0.59	0.741	246	0.988	0.826	0.899	69	0.158
time	30	20	0.952	0.69	0.8	22	0.957	0.759	0.846	2	0.046
wget	440	264	0.992	0.603	0.75	370	0.995	0.862	0.924	106	0.174
average	337	158	0.980	0.530	0.676	236	0.985	0.747	0.841	78	0.165

Fig. 5. comparison with pre-processed binary on stack smashing protector option

BCF	total	before pre-processing				after pre-processing				variation	
		TP	precision	recall	f-score	TP	precision	recall	f-score	TP	f-score
cpio	347	121	0.992	0.35	0.517	187	0.995	0.534	0.695	66	0.178
ftp	243	83	1	0.345	0.509	147	1	0.579	0.733	64	0.224
hello	84	40	1	0.476	0.645	51	1	0.607	0.756	11	0.111
ping	175	60	1	0.343	0.511	103	1	0.589	0.741	43	0.23
sed	246	54	1	0.22	0.36	108	1	0.439	0.61	54	0.25
tar	1167	351	0.992	0.302	0.462	633	0.992	0.538	0.698	282	0.236
telnet	304	99	1	0.329	0.495	174	1	0.572	0.728	75	0.233
time	30	17	1	0.567	0.723	20	1	0.667	0.8	3	0.077
wget	440	126	1	0.286	0.445	213	1	0.484	0.652	87	0.207
average	337	89	0.95	0.35	0.50	134	0.95	0.52	0.67	76	0.194

Fig. 6. comparison with pre-processed binary on bogus control flow pass(obfuscation techniques)

Substituting the predictable changes with NOP instructions helps reduce differences unrelated to the feature of the function. It improves similarity analysis performance of both the semantic and syntactic information similarity analysis.

IV. Evaluation

4.1 Experimental environment

To verify our proposed method, we use nine binaries of GNU source codes(cpio, ftp, hello, ping, sed, tar, telnet, time and wget) with SSP option and BCF obfuscation technique respectively and the Diaphora as a similarity analysis tool. we compare the two cases. In one case, we perform a similarity analysis between binaries which options are not applied and binaries which SSP or BCF are applied without pre-processing. In another case, we do pre-processing to the binaries and perform the similarity analysis again. The results of the Diaphora include reliability value. Among the results from the Diaphora, only the matching pairs having reliability value 0.6 or higher are considered. Because a matching pairs that meet this confition is likely to be actually true pairs and does not reduce recall significantly.

name	SSP	BCF
cpio	554	2544
ftp	401	2530
hello	125	365
ping	300	1209
sed	419	2821
tar	1949	14080
telnet	548	2591
time	29	126
wget	790	5004

Fig. 7. The number of detected instruction block

4.2 Result of pre-processing

After pre-processing was performed, Figure. 7. is a number of instruction blocks replaced. For example, cpio has 117,767 bytes of text section. We belive that a block of SSP pattern is roughly 12 bytes. We can see that among the area of

117,767 bytes, about 6,648bytes is changed. It shows that a considerable number of code blocks inserted by compiler were replaced with NOP instructions.

4.3 Comparison

In the entire binaries, the precision value is high both before and after pre-processing[Fig. 5, Fig. 6]. If the diaphora reliability value is 0.6 or more, the matching result shows high accuracy. The recall increase 21.7% for SSP option and 17% for BCF obfuscation technique with pre-processing. This shows that the more functions can be identified through pre-processing. And the F-score, which is used to measure performance for classification model, increases by 0.165 and 0.194 for each options respectively

V. Conclusion

The similarity analysis has obstacles because of the changes inserted at compilation steps. But among the changes, some changes are predictable because instructions are inserted in a similar form. In this paper, we propose a method to reduce differences by replacing the changes with NOP instructions in advance of similarity analysis, and verify the method through comparison. In the experiment, with pre-processing, F-score increase by 17%, 21.7%.

VI. Acknowledge

This research was supported by the Korea Ministry of Science and ICT. It was also supported by the Korea National Security Research Institute(Project No.: 2018-059)

[Reference]

[1] Yikun Hu, Yuanyuan Zhang, Juanru Li, Dawu Gu, "Cross-Architecture Binary Semantics Understanding via Similar Code Comparison", IEEE 23rd International Conference, 2016

[2] Jiang Ming, Meng Pan and Debin GAO, "iBinHunt: Binary Hunting with Inter-Procedural Control Flow"

[3] joxeankoret, diaphora. A diffing tool, https://github.com/joxeankoret/diaphora

[4] Mauel Egele, Maverick Woo, Peter Chapman, David Brumley, "Blanket Execution: Dynamic Similarity Testing for Program Binaries and Components", Open access to the Proceedings of the 23rd USENIX Security Symposium, August. 2014

[5] Jiang Ming, Dongpeng Xu, Yufei Jiang, and Dinghao, "BinSim: Trace-based Semantic Binary Diffing via System Call Sliced Segment Equivalence Checking"

[6] Jannik Pewny, Behrad Garmany, Robert Gawlik, Christian Rossow, Thorsten Holz "Cross-Architecture Bug Search in Binary Executables", IEEE Symposium on Security and Privacy, 2015

[7] Jiang Ming, Dongpeng Xu, and Dinghao Wu, "Memoized Semantics-Based Binary Diffing with Application to Malware Lineage Inference", IFIP International Information Security Conference, 2015

[8] Yikun Hu, Yuanyuan Zhang, Juanru Li, Dawu Gu, "Cross-Architecture Binary Semantics Understanding via Similar Code Comparison", IEEE 23rd International Conference, 2016

핸들러 기능 매핑을 사용한 VMProtect 가상화 코드내의 비교 연산자와 조건문 분석

전하람
서울과학기술대학교 글로벌 융합산업공학과

Analysis of Comparative Operator and Conditional statement in Virtualized code with VMProtect using Handler function mapping

Department of Industrial and Information Systems Engineering, Seoul National University of Science and Technology

요 약

소프트웨어 개발자는 자신이 만든 소프트웨어의 분석을 어렵게 하기 위해 Anti-Reversing 기법인 코드 가상화를 사용한다. 관련 소프트웨어 VMProtect에선 가상의 스택과 가상의 레지스터를 기반으로 핸들러를 호출해가며 코드를 실행시킨다. 본 논문에선 VMProtect로 가상화 된 코드에서 비교 연산자와 조건문이 실행되는 방법을 핸들러 기능 매핑을 통해 분석하여 조건문 우회 방법을 제시한다.

I. 서론

코드 가상화는 패킹의 한 종류로 원래의 프로그램 코드를 바이트 코드로 만들고, 프로그램에 삽입된 가상 머신이 이를 실행한다[1]. 가상 머신 내에선 Virtual CPU가 가상의 스택과 가상의 레지스터를 대상으로 동작하는 핸들러를 호출한다. 핸들러는 핸들러 테이블을 참조하여 호출되며 각각 mov, add, push, pop 등의 명령어 기능을 갖고 있다[2].

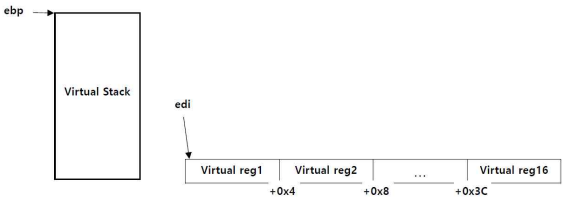
상용 패커 VMProtect에선 보호하고 싶은 코드 시작과 끝에 VMProtectBeginVirtualization과 VMProtectEnd를 사용하여 code virtualization기능을 적용시킬 수 있다. 대부분의 패커들은 소프트웨어의 코드를 암호화 하고 시작지점에서 복호화 작업을 수행하지만, code virtualization기능은 암호화와 복호화 없이 동작한다. 또한, 두 개의 피연산자들을 비교하는데 아무런 명령어를 갖고 있지 않으며 조건문 또한 갖고 있지 않다[3].

본 논문에서는 if(a==b)와 같은 간단한 조건문을 가상화 시켜 핸들러들을 기능별로 매핑하고, 가상 스택을 기반으로 연산되는 일련의 과정을 표로 나타내었다. 이 표를 통해 가상화 코드 내에서 데이터를 비교하는 방법과 조건문에서 실행의 흐름이 바뀌는 방법을 나타내고 조건문의 우회 방법을 제시하여 추후 연구에 도움을 주고자 한다.

II. 관련 연구

2.1 가상 스택과 가상 레지스터

가상화 코드 내에서 사용되는 가상 스택과 가상 레지스터는 가상화가 시작되며 주소가 정해진다. [그림 1]과 같이 실제 레지스터 EBP가 가상 스택의 ESP를, 실제 레지스터 EDI가 가상 레지스터 VM_Context의 주소를 나타내지만[4], VMProtect의 버전이 올라가며 다른 레지스터들도 나타낼 수 있게 되었다.

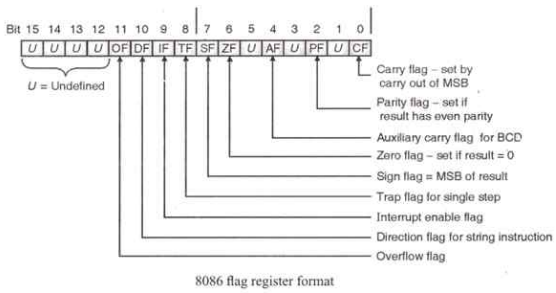


[그림 1] 가상 스택과 가상 레지스터

본 논문에선 EBP와 EDI를 가상 스택과 가상 레지스터의 주소로 사용하였다.

2.2 상태 레지스터(eflag register)

상태 레지스터란 [그림 2]과 같이 다양한 산술 연산 결과의 상태를 알려주는 플래그 비트들이 모인 16비트 레지스터다. 값이 같은지를 확인하는 ==연산자는 비교 연산 후 상태 레지스터의 Zero flag 값을 0 또는 1로 바꾼다. 가상화 코드 내에서도 이러한 플래그 값을 사용하지만 비트 마스킹(bit masking)을 통해 직접 검출하고 검출 값에 따라 핸들러의 호출 순서가 달라져 조건 분기가 실행된다.



[그림 2] 상태 레지스터

2.3 핸들러 기능 매핑

핸들러들은 가상 스택과 가상 레지스터를 대상으로 하는 가상 명령어와 1:1 매칭이 될 수 있다[5]. 또한 핸들러는 VM_instruction_pointer를 참조하여 다음 핸들러를 호출하는데, ESI와 같은 특정 레지스터가 가리키고 있으며 이 값에 따라 핸들러의 호출 순서가 결정된다[4].

하나의 핸들러에서 가상 스택(ESP)과 가상 레지스터(EDI)에 접근하여 연산하는 주요 어셈블리어만 골라 가상 명령어로 바꾸면 [표 1]과 같다. (직관성과 표기의 편의를 위해 문법에 맞지 않은 어셈블리어도 있으며 some reg는 특정 인덱스의 가상 레지스터이다.)

asm in Handler	Virtual Instruction

mov edx, [ebp]
add ebp, 0x4
mov [edi+eax], edx
	pop some reg

mov edx, [edi+eax]
sub ebp, 0x4
mov [ebp], edx
	push some reg

mov edx, ebp
sub ebp, 0x4
mov [ebp], edx
	push esp

mov edx, [ebp]
mov ecx, [edx]
mov [ebp], ecx
	mov [esp], [[esp]]

mov edx, [ebp]
mov cl, [ebp+4]
sub ebp, 2
shr edx, cl
mov [ebp], edx
	mov A, [esp]
	shr A, n
	sub esp, 2
	mov [esp+4], A

mov edx, [ebp]
mov ecx, [ebp+4]
add edx, ecx
mov [ebp+4], edx
pushfd
pop [ebp]
	add [esp+4], [esp]
	mov [esp], eflag register

mov edx, [ebp]
mov ecx, [ebp+4]
not edx
not ecx
and edx, ecx
mov [ebp+4], edx
pushfd
pop [ebp]
	or [esp+4], [esp]
	not [esp+4]
	mov [esp], eflag register

[표 1] 핸들러 기능

하나의 핸들러 내부엔 더미(dummy) 어셈블리어와 VM_instruction_pointer를 참조하여 다음 핸들러를 호출하는 부분도 있지만, 기능을 파악하는데 불필요하여 생략하였다. [표 1]에 나온 핸들러 이외에도 다양한 핸들러들이 있고 VMProtect 버전에 따라 '메인 핸들러'가 존재하기도 한다. 메인 핸들러는 가상 스택과 가상 레지스터에 접근하여 연산을 하지 않으며 대신 VM_instruction_pointer를 참조하여 다음 핸들러를 호출하는 기능만을 갖고 있다. 따라서 가

상 스택과 가상 레지스터에 접근하여 연산을 하는 서브 핸들러와 다음 서브 핸들러를 호출하는 메인 핸들러가 번갈아 가며 호출된다. 이 경우에도 서브 핸들러들에서 가상 스택과 가상 레지스터에 접근하는 어셈블리어를 골라 가상 명령어를 매핑시키면 분석이 가능하다.

III. 분석

3.1 분석 방법

if(Data_X == Data_Y) 조건문을 가상화한 뒤 가상 스택과 가상 레지스터에 관련된 부분을 함께 출력되도록 트레이싱했다. 트레이싱 결과에서 가상 스택을 기반으로 이뤄지는 비교 연산과 조건문 내의 참, 거짓에 따른 핸들러 호출 순서를 중점적으로 관찰 하였다.

3.2 비교 연산

Virtual Instruction	Virtual Stack
-	Data_X Data_Y
push esp	esp+4 Data_X Data_Y
mov [esp], [[esp]]	Data_X Data_X Data_Y
or [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy eflag !Data_X Data_Y
pop some reg	!Data_X Data_Y
add [esp+4], [esp] mov [esp], eflag register	eflag_1 !Data_X+Data_Y
pop some reg	!Data_X+Data_Y
push esp	esp+4 !Data_X+Data_Y
or [esp+4], [esp] not [esp+4] mov [esp], eflag register	eflag_2 !(!Data_X+Data_Y)
pop some reg	!(!Data_X+Data_Y)
pop some reg	-

[표 2] 비교 연산

[표 2]는 Data_X와 Data_Y가 가상화 코드 내에서 비교 연산되는 시작과정을 핸들러의 호출 순서에 따라 나타낸 것이다. Virtual Instruction은 핸들러의 기능을 나타낸 것이고 Virtual Stack은 해당 기능이 실행된 후의 가상 스택의 모습이다.

비교 연산 대상을 가상스택에 위치시키고 !Data_X+Data_Y와 !(!Data_X+Data_Y)의 연산 후 각각의 상태레지스터 eflag_1과 eflag_2를 가상 레지스터에 저장한다. Data_X와 Data_Y가 같다면 !Data_X+Data_Y값이 0xFFFFFFFF이고 !(!Data_X+Data_Y)값은 0이므로 연산 후의 Zero flag가 1인 상태로 eflag_2가 저장된다. 두 값이 같지 않을 경우 !(!Data_X+Data_Y)의 값은 0이 아니므로 Zero flag가 0인 상태로 eflag_2가 저장된다.

eflag_1의 용도는 비교 연산자가 >, >=, <=, <일 경우 !Data_X+Data_Y의 연산 결과에서 Overflow flag와 Carry flag를 저장하는데 사용된다.

3.3 플래그 마스크

3.2에서 가상 레지스터에 저장되었던 eflag_1과 eflag_2를 각각 비트 마스킹한 후 하나의 상태레지스터로 만든다.

Virtual Instruction	Virtual Stack
push some reg	eflag_1
push memory data	0x815 eflag_1
and [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy eflag !(0x815 && eflag_1)
pop some reg	!(0x815 && eflag_1)
push esp	esp+4 !(0x815 && eflag_1)
mov [esp], [[esp]]	!(0x815 && eflag_1) !(0x815 && eflag_1)
and [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy eflag masked_eflag_1
pop some reg	masked_eflag_1
push some reg	eflag_2 masked_eflag_1
push memory data	0xfffff7ea eflag_2 masked_eflag_1
and [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy eflag !(0xFFFFF7EA && eflag_2) masked_eflag_1
pop some reg	!(0xFFFFF7EA && eflag_2) masked_eflag_1
push esp	esp+4

	!(0xFFFFF7EA && eflag_2) masked_eflag_1
mov [esp], [[esp]]	!(0xFFFFF7EA && eflag_2) !(0xFFFFF7EA && eflag_2) masked_eflag_1
or [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy eflag masked_eflag_2 masked_eflag_1
pop some reg	masked_eflag_2 masked_eflag_1
add [esp+4], [esp] mov [esp], eflag register	dummy eflag final_eflag
pop some reg	final_eflag

[표 3] 비트 마스킹

[표 3]에서 final_eflag는 비트 마스킹이 완료된 masked_eflag_1과 masked_eflag_2의 합으로 만들어 진다. 비트 마스킹을 위해 사용되는 값은 메모리에 암호화 되어 있다가 특정 핸들러(push memory data)가 복호화해서 가상 스택에 저장한다. 연산과정이 복잡하지만 간단히 정리하면 eflag_1와 0x815를 and연산 하여 Overflow flag와 Carry flag를 마스킹하고, eflag_2와 0xFFFFF7EA를 and연산 하여 Zero flag와 Sign flag를 마스킹한다. masked_eflag_1과 masked_eflag_2를 더해 만들어진 final_eflag에서는 사용된 비교 연산자가 무엇이냐에 따라 후에 검출되는 값이 달라진다.

3.4 Zero flag 검출 및 분기

Virtual Instruction	Virtual Stack
push memory data	0xFFFFFFBF final_eflag
or [esp+4], [esp] not [esp+4] mov [esp], eflag register	dummy flag !(0xFFFFFFBF || final_eflag)
pop some reg	0x40 && (!final_eflag)

mov A, [esp]
shr A, n
sub esp, 2
mov [esp+4], A
	dummy data
	0 or 1

[표 4] Zero flag 검출

[표 4]에서 !final_eflag는 0xFFFFFFBF와 NOR 연산되어 Zero flag가 검출된다. 검출된 값은 [표 4]의 마지막 핸들러에서 오른쪽으로 6번 시프팅되어 Data_X와 Data_Y가 같다면 0이, 다르다면 1이 된다. 최종적으로 연산된 이 0과 1은 여러 연산 과정을 통해 새로운 VM_instruction_pointer을 만들고 VM_instruction_pointer를 가리키는 레지스터를 바꾸는 핸들러가 호출된다. 그 뒤로 실행 흐름인 핸들러의 호출 순서가 서로 달라졌기 때문에 결과적으로 조건 분기가 실행된 것이다.

IV. 우회 방법

우회 방법은 연산 된 결과의 상태 레지스터를 조작하는 것이다. 본 논문에선 비교 연산자 중 ==을 대상으로 분석하여 Zero flag가 검출되었지만 >, <, >=, <=와 같은 연산자들의 경우 비교 대상의 자료형이 signed/unsigned에 따라 jge, jbe, ja, jl등의 어셈블리어로 컴파일 되고 그에 맞게 검출되는 플래그도 바뀌게 된다. 하지만 검출되기 전 일련의 과정을 거쳐 final_eflag가 만들어 지는 것 까지는 동일하기 때문에 해당 부분을 찾아 상태 레지스터를 조작하여 조건문을 우회할 수 있다. 비교 연산자에서 사용되는 상태 레지스터는 Overflow flag, Sign flag, Zero flag, Carry flag 4가지이므로 사용된 연산자를 모르는 상태에서 단순히 조건문 우회가 목적이라면 final_flag의 위 4가지 flag를 바꿔가며 실행시키는 방법도 있다.

또한 비교 연산 과정이 시작되는 부분에서 가상 스택의 비교되는 데이터 값을 조작하여 우회하는 방법도 있지만 데이터가 실행할 때마다 바뀌는 경우 직접 비교연산을 한 뒤 조작해야하는 불편함이 있다.

V. 결론

본 논문에선 상용 소프트웨어 VMProtect로 가상화 된 코드 내 조건문과 비교 연산자가 실행되는 과정을 핸들러 기능 매핑과 가상 스택을 표로 나타내어 분석하고 우회 방법을 제시하였다. 이를 통해 Virtual CPU가 값을 비교하는 과정이 가상 스택을 기반으로 상태 레지스터를 검출하고 조건 분기하며 이는 Real CPU의 방법과 유사하다는 것을 알 수 있었다.

하지만 몇 가지 부족한 요소가 있으며 추가 연구가 필요하다. 첫째로 핸들러의 기능 매핑

과정이다. 기존의 방법들은 메인 핸들러를 중심으로 호출되는 서브핸들러를 분석하고[6], 가상 스택, 가상 레지스터와 VM_instruction_pointer을 대상으로 접근하는 특정 레지스터들을 통해 기능을 매칭하였지만, VMProtect의 버전이 올라가며 다른 레지스터도 사용되고 동일한 기능의 핸들러를 여러개 만들어 놓는 등 분석이 어려워져 매핑 자동화에 대한 연구가 필요하다. 둘째로 핸들러 기능 매핑 후 특정 부분을 찾는 과정이다. 비교 연산 부분의 경우 본 논문에서 소개한 핸들러의 호출 과정이 순차적이기 때문에 이를 토대로 전체 과정에서 검색하면 효과적일 것으로 추정된다. 하지만 함수 호출과 같은 다른 기능의 핸들러 호출 분석과 추가적인 검색방법에 대한 연구가 필요하다.

[참고문헌]

[1] 목성균・전현구・조은선 (2017). "역난독화를 위한 바이너리 프로그램 슬라이싱" 정보보호학회논문지

[2] 박지훈 (2012). "Code Virtualized"

[3] 박용수・김성호 외 2인 (2012). "코드 가상화 기법이 적용된 악성코드 분석 방법 연구." 한국인터넷진흥원

[4] Samuel Chevet (2015). "Inside VMProtect"

[5] 이성호・한태숙 (2014). "가상 머신 기반으로 난독화된 실행파일의 구조 및 원본의미 추출 동적 방법." 정보과학학회논문지

[6] 원지혜, 한태숙 (2014). "가상화 기법으로 난독화된 실행 파일의 동적 분석을 통한 핸들러 간략화" 한국정보과학회

정적 바이너리 계측을 활용한 메모리 취약점 탐지기술 연구*

김형석†, 김현태*, 이승엽*, 차상길*
†국가보안기술연구소, *한국과학기술원 정보보호대학원

Binary-Only Address Sanitizer using Static Binary Instrumentation

Hyugseok Kim†, Hyuntae Kim*, Seungyeop Lee*, Sang Kil Cha*
†National Security Research Institute, NSR.
*Graduate School of Information Security, KAIST.

요 약

메모리 취약점은 권한 상승, 임의 코드 실행, 권한 탈취 등 다양한 사이버 상의 공격을 야기한다. 이러한 메모리 취약점을 탐지하기 위한 연구들이 활발히 진행되고 있으며 일부는 도구화되어 사용되고 있다. 그러나 대부분의 기존 연구들은 소스코드가 없는 바이너리 프로그램에는 적용이 불가능한 한계를 가지고 있으며, 동적 계측을 이용한 메모리 취약점 분석은 높은 오버헤드 문제를 가지고 있다. 본 논문에서는 바이너리 코드를 대상으로 효율적인 메모리 취약점 탐지를 구현하기 위한 도전 과제들을 정의하고, 이에 대한 해결 방법을 제시하고자 한다. 그리고 정적 바이너리 계측을 활용한 취약점 탐지 도구인 BiO-ASan을 제안하고, 바이너리 기반 메모리 취약점 탐지의 가능성을 재고하고자 한다.

I. 서론

메모리 취약점은 시스템 크래쉬나 정의되지 않은 프로그램의 실행을 야기한다. 최근 CVE 통계에 의하면, Heap Out-of-Bound (OOB) Read, Use-After-Free, Heap Corruption 등 힙(Heap) 메모리에서 발생하는 취약점이 대다수를 차지하고 있다.[9] 이러한 취약점은 공격자의 권한 상승, 임의 코드 실행, 정보 탈취 등 다양한 공격에 활용되고 있다.

취약점을 탐지하기 위해 Sanitizer에 대한 연구가 활발히 진행되었으며, 일부는 도구화되어 사용되고 있다. Sanitizer는 비정상적 메모리 접근을 실행시간에 탐지하는 실용적인 도구들을 통칭하는 용어이며, 메모리 취약점 탐지 도구(Address Sanitizer), Thread Sanitizer 등이 있다.

메모리 취약점 탐지를 위한 기존 연구들은 정교한 Sanitizing을 수행하기 위해 프로그램 언어[6][7][8]나 중간언어 수준[1][4][5]의 계측(instrumentation)을 수행한다. 하지만 이는 소스코드를 필요로 하기 때문에 상용 소프트웨어를 대상으로는 적용이 불가능한 한계를 가진다. 또한 Valgrind[3] 등의 바이너리 기반의 탐지 도구들의 경우 대개 동적 계측을 수행하기 때문에 큰 오버헤드가 따른다.

본 논문에서는 바이너리 프로그램을 대상으로 효율적인 취약점 탐지 도구를 구현하기 위한 도전 과제들을 정의하고, 이에 대한 해결 방법을 제시하고자 한다. 그리고 소스코드가 없는 레거시 프로그램에서도 적용 가능한 메모리 취약점 탐지 도구인 BiO-ASan을 제안한다. BiO-ASan은 일반 바이너리 프로그램을 대상으로 힙 메모리 취약점 탐지를 위한 정적 바이너리 계측을 수행한다.

★ 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.B0717-16-0109, 바이너리 코드 분석을 통한 자동화된 역공학 및 취약점 탐지 기반기술 개발).

본 논문의 성과는 다음과 같다.

- 첫째, 바이너리 기반 메모리 취약점 탐지 도구를 구현하는데 있어서의 한계와 가능성을 정의한다.
- 둘째, 바이너리 수준에서 적용 가능한 새로운 메모리 취약점 탐지 기술을 제안한다.
- 셋째, 구현된 제안한 기술에 대한 테스트를 통해 향후 연구 가능성을 제시한다.

본 논문 구성은 다음과 같다. 2장에서 바이너리 프로그램에 대한 힙 메모리 취약점 탐지 도구인 BiO-ASan의 설계를 소개하고 3장에서 Bio-ASan의 구현 결과를 소개한다. 그리고 4장에서 본 연구의 결론과 future work를 제시한다.

II. 설계

메모리 공간에 대한 메모리 취약점 탐지 도구를 구현하기 위해서는 메모리 객체의 경계 정보와 메모리 접근에 대한 문맥(semantics) 정보가 필요하다. 하지만 바이너리에서는 메모리 타입 정보와 소스코드 레벨의 추상화된 문맥이 없기 때문에 메모리 취약점 탐지 도구를 구현하는데 있어서 한계가 존재한다. 본 논문에서는 힙 메모리에 청크 할당시 메모리 객체 주변에 레드존을 삽입하고, 어셈블리 명령어가 내포하는 주소와 사이즈 정보를 활용하여 레드존 침범을 체크하는 계측 방법을 소개한다.

바이너리 프로그램에 메모리 취약점 탐지 루틴을 삽입하기 위한 BiO-ASan의 디자인은 다음과 같다. 첫 번째, 동적으로 생성된 메모리 공간에 shadow 메모리 생성한다. 두 번째, 동적 메모리 생성 함수를 후킹하여 할당된 메모리 앞뒤로 레드존을 삽입한다. 세 번째, 바이너리 재조립 도구를 활용하여 바이너리 파일을 어셈블리 코드로 변환하고, 메모리 접근 명령을 식별해 계측 코드를 삽입한다.

2.1 Shadow 메모리 생성/접근

기존 메모리 취약점 탐지도구인 ASan[1]에서는 실행 메모리 상태를 shadow 메모리에 투영하여 표현한다. 프로그램 실행시 계측 수행으로 shadow 메모리 정보를 조회하고 레드존 침범 여부를 판단한다. ASan에서 shadow 메모리는 고정된 주소에 위치한다. 그 결과 macOS와 같은 일부 플랫폼에서는 ASLR를 비활성화해야 하는 제약이 발생한다. 또한 절대 주소를 사용하는 바이너리 프로그램에서 고정된 위치의 shadow 메모리를 사용할 경우, 일부 메모리 주소가 오버랩 되는 문제가 발생할 수 있다. 이러한 문제를 해결하고자 본 연구에서는 동적으로 shadow 메모리를 할당하고 세그먼트 레지스터를 이용해 shadow 메모리에 접근하는 방법을 제안한다.

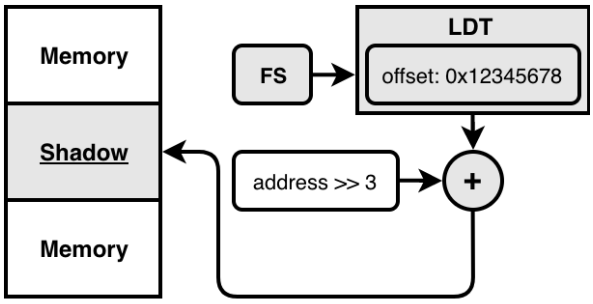


그림1. 세그먼트 레지스터를 이용한 shadow 메모리 접근

그림1은 Local Descriptor Table(LDT)에 등록된 shadow 메모리 주소를 세그먼트 레지스터를 이용해 접근하는 과정을 나타내고 있다.

2.2 동적 메모리 생성 함수 후킹

동적으로 할당된 메모리 객체 앞뒤로 레드존을 삽입하기 위해, 본 연구에서는 동적 메모리 생성 함수를 후킹하여 레드존을 삽입한다. 후킹 함수에서는 레드존을 포함하는 메모리를 할당하고, 메모리 객체 앞뒤로 레드존을 삽입한다. 그리고 shadow 메모리에 상태정보를 설정한 뒤, 유효 주소 공간을 반환한다.

2.3 메모리 접근 코드에 대한 계측 코드 삽입

바이너리 코드에 레드존 체크 루틴을 삽입하기 위해, 본 연구에서는 바이너리 재조립 도구인 Ramblr[2]를 이용해 바이너리 프로그램에 대한 어셈블리 코드를 생성한다. 그리고 정규표현식을 이용해 어셈블리 코드에서 메모리 접근 명령어를 식별하고 계측 코드를 삽입한다. 이 때 어셈블리 코드 레벨에서 식별가능한 메모리 주소 정보와 사이즈 정보를 계측 코드에 전달한다. 컴파일 과정을 거쳐 생성된 새로운 바이너리 프로그램에서는 계측 코드를 통해 레드존 침범 여부를 체크한다.

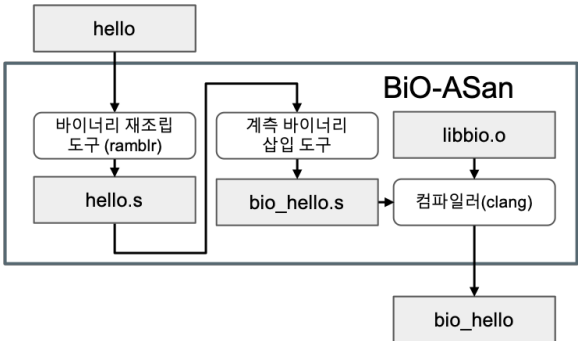


그림2. BiO-ASan의 Sanitizer 생성 과정

III. 구현 결과

BiO-ASan은 일반 바이너리 프로그램에서 힙 메모리 취약점 탐지 코드가 포함된 바이너리 프로그램을 생성한다. 그림2는 BiO-ASan을 통해 테스트 바이너리 프로그램 hello에 힙 메모리 취약점 탐지 코드가 삽입되는 과정을 나타낸다. BiO-ASan은 바이너리 재조립 도구를 이용해 어셈블리 코드 hello.s를 얻은 뒤, 계측 코드를 삽입한 bio_hello.s 파일을 생성한다. 그리고 bio_hello.s 파일과 shadow 메모리 생성, 동적 메모리 함수에 대한 후킹 함수를 포함하는 라이브러 파일(libbio.o)을 함께 컴파일하여 새로운 바이너리 프로그램인 bio_hello를 생성한다.

BiO-ASan을 통해 생성된 바이너리 프로그램은 레드존 침범을 체크하여 OOB 접근을 탐지한다. OOB 접근이 발생한 경우, 해당 메모리의 주소와 shadow 메모리 값, 그리고 접근에 필요한 shadow 메모리 값 정보를 출력한다.

3.1 힙 메모리 Address Sanitizing

힙 메모리 취약점 탐지 확인을 위해 아래와 같은 샘플 코드 hello.c를 작성하였다. 샘플 코드 hello.c에서는 8번째 줄과 13번째 줄에서 OOB 접근이 발생한다.

```
1 int main(void) {
2     int *a; char *buf; int i;
3     puts("a=malloc(), buf=malloc()");
4     a = (int *)malloc(2);
5     buf = (char *)malloc(SIZE);
6
7     puts("assign 7 to a");
8     *a = 7; //Out-of-Bound!!
9
10    puts("fill buf with 'A'");
11    for (i = 0; i < SIZE - 1; i++)
12        buf[i] = 'A';
13    buf[SIZE]='\0'; //Out-of-Bound!!
14
15    free(a);   free(buf);
16    return 0;
17 }
```

아래 결과는 BiO-ASan을 적용하지 않은 바이너리의 실행 결과이다. 코드 실행 중 OOB 접근이 발생하였지만 특별한 증상 없이 프로그램이 정상 종료되었다.

```
1  $ ./hello
2  a = malloc(), buf = malloc()
3  assign 7 to a
4  fill buf with 'A'
```

다음은 BiO-ASan을 통해 생성된 bio_hello를 실행한 결과이다. 4번째 줄에서 샘플 코드의 첫 번째 OOB 접근(hello.c의 8번째 줄)에 대한 탐지 메시지가 출력되었다. 주소 공간 0x8550430에 대한 shadow 메모리 값은 2인데, 접근에 필요한 값은 4 이상이라서 레드존 침범이 탐지 되었다. 6번째 줄에서는 샘플 코드의 두 번째 OOB 접근(hello.c의 13번째 줄)이 탐지 되었다. Shadow 메모리 값이 0xff로 설정된 레드존 영역의 주소 공간 0x8550498에 접근하면서 OOB 접근이 탐지되었다.

```
1  $ ./bio_hello
2  a=malloc(), buf=malloc()
3  assign 7 to a
4  state(0x8550430): 02 04:: OOB!!
5  fill buf with 'A'
6  state(0x8550498): ff 01:: OOB!!
```

3.2 기존 바이너리 재조립 도구의 한계

BiO-ASan에서는 바이너리 재조립 도구를 이용하여 바이너리 프로그램에 대한 어셈블리 코드

를 생성하고, 생성된 어셈블리 코드를 바탕으로 계측을 수행한다.

바이너리 재조립은 바이너리 코드를 재프로그래밍이 가능한 형태의 어셈블리 코드로 변환하는 연구로 현재 활발히 진행 중이다. 그러나 현존하는 바이너리 재조립[2][10] 도구들은 어셈블리 재조립을 수행하는데 있어 추론적인(heuristic) 방법을 사용하고 있어 정확도 면에서 한계를 가지고 있다.

본 논문에서는 coreutils 어플리케이션에 대해 BiO-ASan을 적용하기 위해 Ramblr를 이용하여 어셈블리 코드를 생성하였다. 그러나 Ramblr의 추론적인 데이터 타입 예측의 실패로 잘못된 어셈블리 코드가 생성되었다. 그 결과 coreutils에 대한 테스트 진행시 다수의 오류상황이 발생하였다. 테이블1은 Ramblr로 생성한 바이너리 77개에 대해 coreutils의 테스트 프로그램(test suite)을 진행한 결과이다. 35% 이상의 테스트에서 실패/에러가 발생한 것을 확인할 수 있다. 바이너리 재조립 도구를 이용한 정적 계측을 위해서는 바이너리 재조립 도구의 기능상의 한계를 개선할 필요가 있다.

PASS	FAIL	ERROR	TOTAL
304	119	47	470

테이블1. Ramblr로 생성한 coreutils 바이너리에 대한 테스트 프로그램 수행 결과

IV.결론

본 논문에서는 소스코드가 없는 바이너리 프로그램에 대한 메모리 취약점 탐지 도구의 한계와 가능성을 정의하였다. 그리고 동적 메모리 할당 함수에 전달되는 사이즈 정보를 활용하여 주변에 레드존을 삽입하는 방법과 어셈블리 코드 레벨에서 확인할 수 있는 메모리 주소와 사이즈 정보를 활용해 계측을 수행하는 방법을 소개하였다.

또한 본 논문에서는 바이너리 코드에 대한 메모리 취약점 탐지 도구인 BiO-ASan을 제안하였다. 실험을 통해 BiO-ASan으로 생성된 바이너리 프로그램에서 힙 메모리의 OOB 접근 탐지가 가능함을 확인하였다. 이와 함께 본 연구에서는 바이너리 프로그램에 대한 정적 계측을 성공적으로 수행하기 위해서는 바이너리 재조립 기술에 대한 연구가 필요함을 확인하였다.

본 연구를 통해 바이너리 기반 메모리 취약점 탐지의 가능성을 살펴보았다. 앞으로 우리는 본 연구결과를 확장하여 소스코드가 없는 바이너리 프로그램에서 스택 및 전역 변수에 대한 메모리 취약점 탐지 도구에 대한 연구를 진행할 계획이다.

[참고문헌]

[1] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "AddressSanitizer: A fast address sanity checker," in USENIX ATC'12, 2012.

[2] R. Wang, Y. Shoshitaishvili, A. Bianchi, A. Machiry, J. Grosen, P. Grosen, C. Krügel, and G. Vigna, "Ramblr: Making reassembly great again," in NDSS, 2017.

[3] N. Nethercote and J. Seward, "Valgrind: A frameworkfor heavyweight dynamic binary instrumentation." in PLDI, 2007.

[4] Intel, "Intel pointer checker," https://software.intel.com/en-us/node/522703, 2017.

[5] W. Han, B. Joe, B. Lee, C. Song, and I. Shin, "Enhancing memory error detection for large-scale applications and fuzz testing," in NDSS, 2018.

[6] LLVM Developers, "Undefined behavior sanitizer," http://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html, 2017.

[7] I. Haller, Y. Jeon, H. Peng, M. Payer, C. Giuffrida, H. Bos, and E. Kouwe, "TypeSan: Practical type confusiondetection," in ACM CCS, 2016.

[8] LLVM Developers, "TySan: A type sanitizer," https://lists.llvm.org/pipermail/llvm-dev/2017-April/111766.html, 2017.

[9] D. Weston, M. Miller, "Windows 10 Mitigate Improments," 17p. in BlackHat USA 2016.

[10] S. Wang, P. Wang, and D. Wu, "Reassembleable disassembling", in USENIX Security Symposium, 2015.

네이버 음성 CAPTCHA의 취약점 Case Study 분석

전소윤** 우지혜* 최지원* 김형식*** 우사이먼*

한국뉴욕주립대학교 컴퓨터과학과*, 응용수학통계학과**
성균관대학교 소프트웨어학과***

Case Study of Evaluating and Breaking Naver's Audio CAPTCHA using Off-the-Shelf Speech-to-text APIs

Jihye Woo* Soyoon Jeon** Ji Won Choi* Hyoungshick Kim***

Simon S. Woo*

*Computer Science, SUNY Korea

**Applied Mathematics & Statistics, SUNY Korea

***Sungkyunkwan University Korea

요 약

CAPTCHA는 컴퓨터와 인간을 구별하기 위한 테스트로 주로 Bot의 자동화된 공격을 차단하기 위하여 자주 사용된다. 이미지 CAPTCHA와 더불어, 음성 CAPTCHA는 사람은 알아듣기는 쉽지만, Bot은 이해하기 힘든 음성을 생성하여 사람과 컴퓨터를 구별하는데 사용된다. 하지만 현재 많은 회사들이 우수한 성능의 Speech-to-Text APIs를 제공하고 있으며, 이런 APIs들이 Bot의 자동화 공격에 사용될 수 있다. 본 연구에서는 네이버의 한국어 기반의 음성 CAPTCHA off-the-shelf Google, IBM, 네이버의 Speech-to-Text APIs에 대한 인식률을 비교, 평가 및 분석하였다. 연구 결과 API들의 인식률이 휴리스틱 규칙 적용 시 최대 74%로 나타났고 이를 통해 네이버 CAPTCHA의 취약점을 밝혔다.

I. 서론

CAPTCHA(Completely Automated Public Turing tests to tell Computers and Humans Apart)는 컴퓨터와 사람을 구별하기 위해 개발되었다[4]. 최근, 오디오 기반의 CAPTCHA 시스템도 많은 인터넷 서비스에서 시각 장애인을 위해 사용되기 시작하였다. 그러나 최근 음성인식 및 기계 학습 시스템의 비약적인 발전으로 음성 CAPTCHA는 기계와 사용자 간의 구별이 어려워지고 있다. 이러한 상황은 더 악화 될 것으로 보인다. 만약 음성 CAPTCHA가 컴퓨터와 사람을 구별하지 못한다면, 즉 Bot이 자동으로 음성 CAPTCHA를 인식할 수 있다면, 음성 CAPTCHA의 보안이 큰 위협을 받을 것이다.

현재 영어 기반 음성 CAPTCHA의 문제점은 다른 연구들을 통해 밝혀졌으나[5], 네이버에서 제공하는 한국어 기반의 음성 CAPTCHA에 대해서는 관련 연구가 미비하다. 따라서 본 연구에서는 off-the-shelf 음성 인식(Speech-to-Text) APIs를 사용하여 네이버의 음성 CAPTCHA에 관한 자동 인식률을 조사하였다. 특히 Google, IBM 그리고 네이버의 자동 음성인식 APIs를 사용한 결과 최대 74%가 자동으로 인식되어 네이버 CAPTCHA의 취약함이 드러났다.

II. 관련 연구

현재 음성 CAPTCHA API를 제공하는 웹사이트는 국내에서는 다음(카카오)과 네이버가 있으며, 국외로는 Google, eBay 등이 있다. 이미 Google, eBay 등 국외의 음성 CAPTCHA는 다양한 논문에서 문제점 및 새로운 대안이 제시되었다. Google, Digg, reCAPTCHA의 음성 CAPTCHA들은 각각 67%, 71%, 45%의 확률로

깨졌으며[4]. 위의 해당 CAPTCHA들을 무력화 시킬 수 있는 UnCaptcha까지 개발된 상황이다. eBay 음성 CAPTCHA 역시 75%의 확률로 깨졌다[5]. 또한, Google, eBay 등의 음성 CAPTCHA는 VoIP에 적절하지 않다는 것이 밝혀졌다. 이에 따라, 숫자마다 다른 잡음을 적용하는 등 VoIP에 적합한 새로운 CAPTCHA가 제안되었다[6].

자동음성인식기술(Speech-to-Text)은 마이크와 같은 소리 센서를 통해 얻은 음향학적 신호를 단어나 문장으로 변환시키는 기술을 말한다. Google, IBM, Baidu, 네이버 등 다양한 기업들은 자동음성인식기술을 오픈 API로 제공하고 있다[2]. 딥러닝으로 인해 최근에 음성기술은 비약적인 발전을 거듭하고 있으며 앞으로도 많은 양의 데이터와 Computer Power로 인식률이 더 높아질 것이라 예상된다[3]. 그러므로 공격자가 이러한 높은 인식률을 가진 API를 활용한다면, 음성 CAPTCHA는 반대로 큰 위협이 된다.

III. 실험 방법 및 설계

3.1 실험 환경

본 논문에서는 한국어 음성 인식이 지원되는 네이버의 Clova Speech Recognition API, Google의 Speech-to-Text API, IBM의 Speech-to-Text API를 활용하여 네이버의 한국어 음성 CAPTCHA를 인식해보았다.

네이버 Clova API의 경우 Android Studio에서 샘플코드를 이용하여 모바일 어플리케이션으로 컴파일한 후 잡음이 적은 같은 공간에서 네이버의 음성 CAPTCHA를 재생하여 인식 결과를 수집하였다. Google과 IBM의 경우, 웹서버를 통해 제공된 Speech-to-Text API를 활용하여 음성 인식 결과를 수집하였다.

총 50개의 네이버 음성 CAPTCHA에 대한 음성 인식 실험을 해보았다. 각각의 음성 CAPTCHA는 1부터 9까지(하나, 둘, 셋, 넷, 다섯, 여섯, 일곱, 여덟, 아홉)의 숫자 중 무작위의 6개 숫자를 담고 있다. 본 논문에서는 기존의 음성 인식 API를 활용하여 네이버 음성 CAPTCHA의 인식률과 취약한 보안 문제를 분석하고자 한다.

CAPTCHA	네이버	IBM	Google
하나 여덟 셋 일곱 여덟 셋	x 배달 x 일곡 여덟 x	그 여덟 테 일고 여덟 세	283783

[표 1] 음성인식 결과 예시

위의 [표 1]은 실험 결과의 예시이다. 50개의 결과를 두 가지 규칙에 따라 인식률을 분석하였다.

3.2 Strict 규칙 (Exact Matching)

Strict 규칙은 음성인식 결과와 정확한 숫자의 발음이 일치할 경우를 세었다. 즉 "하나, 둘, 셋, 넷, 다섯, 여섯, 일곱, 여덟, 아홉"과 일치할 경우만을 인정하였다.

	1	2	3	4	5	6	7	8	9	Total
Naver	64%	23%	0%	8%	0%	12%	14%	63%	68%	26%
IBM	100%	42%	7%	0%	0%	6%	5%	46%	20%	28%
Google	100%	60%	38%	62%	86%	73%	68%	67%	71%	72%

[표 2] Strict 규칙 적용 시, 숫자별 인식률

[표 2]는 각 숫자에 대한 인식률이다. Google이 72%의 정확도로 가장 높은 인식률을 보였고, 그 다음으로는 IBM이 28%, 네이버가 26%의 인식률을 보여주었다. 네이버와 IBM의 경우 "셋, 넷, 다섯, 여섯"의 인식률이 Google에 비해 낮았다. CAPTCHA가 인간과 컴퓨터를 구분하기 위해서는 컴퓨터가 음성 CAPTCHA를 인식해서는 안 된다. 그러나 Google은 72%, IBM은 28%, 그리고 네이버는 26%의 유의미한 인식률을 보여주었다. 이는 음성 CAPTCHA의 치명적인 문제점으로 개선될 필요가 있다.

3.3 휴리스틱 규칙

다음으로 휴리스틱(heuristic)을 이용하여 인식률을 분석하였다. 즉, 음성을 인식할 때 발생하는 여러 변수를 고려하기 위해 휴리스틱 규칙을 만들고, 음성인식 API로 텍스트 화된 음성 CAPTCHA에 적용하였다.

음성 CAPTCHA의 다양한 변수(발화 속도,

억양, 연음 등)를 고려하여 본 연구에서는 아래와 같은 휴리스틱 규칙을 세웠다.

인정되는 숫자 발음은 현실 발음을 고려하여 '세(셋), 네(넷), 다서(다섯), 여서(여섯), 일고(일곱), 여덜(여덟), 아옵(아홉)'까지 포함하였다.

방법 \ 자리 / 힘		입술소리	잇몸소리	입천장소리	목청소리
터짐소리	예사소리 된소리 거센소리	ㅂ ㅃ ㅍ	ㄷ ㄸ ㅌ	ㄱ ㄲ ㅋ	
붙갈이소리	예사소리 된소리 거센소리			ㅈ ㅉ ㅊ	
갈이소리	예사소리 된소리		ㅅ ㅆ		ㅎ
콧소리		ㅁ	ㄴ	ㅇ	
흐름소리			ㄹ		

[표 3] 한국어 자음과 분화 조건

자음은 [표 3]와 같이 조음 방법에 따라 터짐소리, 붙갈이소리, 갈이소리, 콧소리, 흐름소리로 분류한 후 같은 범주 안에 있는 자음끼리는 차이를 허용한다[2]. 예를 들어, '다섯'을 '타서'로 읽은 경우 'ㄷ'과 'ㅌ'은 같은 터짐소리이므로 '타서' 또한 인정하였다.

혀의 높낮이 \ 조음부 위치 / 입술의 모양	혀의 앞		혀의 뒤	
	안 둥근 소리	둥근 소리	안 둥근 소리	둥근 소리
높은 소리	ㅣ	ㅟ	ㅡ	ㅜ
중간 소리	ㅔ	ㅚ	ㅓ	ㅗ
낮은 소리	ㅐ		ㅏ	

[표 4] 한국어 단모음과 분화 조건

모음은 [표 4]와 같이 전설(혀의 앞) 평순(안 둥근 소리) 모음, 전설 원순(둥근 소리) 모음, 후설(혀의 뒤) 평순 모음, 후설 원순 모음으로 나누어 같은 범주 안의 모음끼리는 차이를 허용하였다[2]. 예를 들어, '일곱(일고)'을 '일구'로 읽은 경우 'ㅗ'와 'ㅜ'는 같은 후설 원순 범주에 속하므로 '일구' 또한 인정해주었다.

또한, 본 연구에서 정한 휴리스틱 규칙에 맞지 않으나 사람이 들었을 때 음성 CAPTCHA와 비슷하게 들리는 결과도 포함하였다. 예를 들어, '하나'의 경우 '아나'라고 읽는 경우가 존재했는데 'ㅎ'은 갈이소리, 'ㅇ'은 콧소리로 같은 범주에 속하지 않는다. 그러나 음성 CAPTCHA의 발음이 유사하게 들려서 예외로 처리하였다. 그리고 사람의 발음과 음성 CAPTCHA 발음이 비슷한 경우 포함하기로 하였다.

이와 반대로, 본 연구에서 정한 휴리스틱 규칙에는 포함이 되지만 음성 CAPTCHA의 오디오 파일과 비슷하지 않은 경우는 제외했다. '여섯(여서)'의 경우 '여사'라고 읽는 경우가 존재했는데 'ㅓ'와 'ㅏ'는 둘 다 후설 평순 모음으로 지정한 휴리스틱에는 부합하나 '여사'와 CAPTCHA를 오디오 파일의 발음이 달라 제외하기로 했다. 이를 바탕으로 총 8개의 휴리스틱 규칙을 정하였다.

하나	둘	셋/세	넷/네	다섯/다서	여섯/여서	일곱/일고	여덟/여덜	아홉/아옵
1	2	3	4	5	6	7	8	9
아나	돌	pet	맵	타사	여서	일곡		아오
않아	굴	해	내	타서		읽고(일꼬)		아호
	들	헷	멧			일구		아우
	툴	헤	매			일고		
		pat	mat			일거		
		페	네			읽어		

[표 5] 휴리스틱 규칙 적용 시, 인정된 목록

IV. 실험 결과 및 분석

휴리스틱 규칙을 적용한 결과 CAPTCHA 인식률이 네이버는 48%, IBM은 55% 그리고 Google은 74%로 상승했다.

	1	2	3	4	5	6	7	8	9	Total
Naver	94%	40%	35%	58%	0%	12%	73%	63%	80%	48%
IBM	100%	46%	7%	96%	20%	55%	55%	46%	43%	55%
Google	100%	67%	38%	69%	86%	76%	68%	67%	71%	74%

[표 6] 휴리스틱 규칙 적용 시, 각 숫자별 인식률

위의 [표 6]은 휴리스틱 규칙을 적용시킨 각 숫자에 대한 인식률이다. Google은 strict 규칙 적용 시에 높은 인식률을 보였기 때문에 휴리스틱 규칙을 적용했을 시에 인식률의 변화가 적었다. 그러나 네이버 Clova의 경우 오인식률이 높았던 "셋, 넷"이 높은 인식률을 보여주었다. IBM의 경우 마찬가지로 "넷, 다섯, 여섯"에 대해 인식률이 높아졌다.

네이버와 IBM은 다수의 오답이 정답과 유

사하였기 때문에 휴리스틱 규칙을 적용할 때 네이버와 IBM의 정확도가 Strict 규칙대비 각각 1.84배, 1.92배로 매우 증가하였다. 공통적으로 휴리스틱 규칙을 적용한 뒤 음성인식 기술의 인식률이 높아지면서 같은 음성 CAPTCHA를 기계와 인간이 인식했을 때의 차이가 줄어든다는 사실을 알 수 있다.

또한, Google은 Strict 규칙 적용 시 72%, 휴리스틱 규칙 적용 시 75%로 세 가지의 음성인식 기술 중 가장 정확하다. 휴리스틱 규칙을 기준으로 전체 6개의 숫자 배열 중 숫자 5~6개를 인식한 비율은 55.1%이다(5개 인식 : 30.6% / 6개 인식 : 24.4%). 전체 숫자 배열 중 반 이상의 숫자들을 대부분 인식했다고 볼 수 있다.

이러한 Google의 높은 인식률은 음성인식 기술로 네이버 CAPTCHA를 쉽게 공격할 수 있는 가능성을 제시해주기 때문에 한국어 음성 CAPTCHA에 대한 새로운 디자인 및 보안이 필요하다.

V. 결론

본 연구는 아직 50개의 네이버 음성 CAPTCHA 샘플로 테스트하였기 때문에 결과에는 한계점이 존재한다. 앞으로 인식률을 정확하게 측정하기 위하여 더 많은 샘플로 연구를 진행할 예정이다.

본 논문에서는 Google, IBM 그리고 네이버 음성 인식 API를 통해 네이버 CAPTCHA를 텍스트 화하고 Strict 규칙과 휴리스틱 규칙을 적용하여 각각의 인식률을 산출하였다.

Strict 규칙을 적용한 결과 Google, 네이버, IBM의 인식률이 각각 72%, 26%, 28%로 나타났고, 휴리스틱을 적용했을 때는 각각 74%, 48%, 55%로 더 높은 정확도를 보여주었다. 결과적으로 휴리스틱 규칙을 적용하게 되면 음성 API로 네이버 음성 CAPTCHA를 깰 수 있는 확률이 매우 높아지게 된다. 이 결과는 현 네이버 CAPTCHA에 취약한 보안 문제가 있다는 점을 보여준다. 따라서 현 네이버 음성 CAPTCHA 시스템은 개선이 필요하다.

[참고문헌]

[1] 국어 국립 원, 새 국어 생활, 2015

[2] 최승주, 음성인식 오픈 API의 음성인식 정확도 비교 분석, 2017

[3] 노희경, 구글, 네이버, 다음 카카오 API 활용 앱의 표준어 및 방언 음성인식 기초 성능평가, 2017

[4] Carnegie Mellon University CyLab, The reCAPTCHA Project - Carnegie Mellon University CyLab". www.cylab.cmu.edu, 2017

[5] Elie Bursztein, Decaptcha: Breaking 75% of eBay Audio CAPTCHAs

[6] Yannis Soupionis, Audio CAPTCHA: Existing solutions assessment and a new implementation for VoIP telephony, 2010

Return Oriented Programming 공격에 대한 방어 기법 동향 분석*

장혜린* 박문찬* 진홍주* 이동훈*
*고려대학교 정보보호학과

Analysis of Defense Techniques for Return-Oriented Programming Attack

Hye Rean Jang*, Moonchan Park*, Hongjoo Jin*, Dong Hoon Lee*
*Graduate School of information security, Korea University.

요 약

ROP(Return Oriented Programming) 공격은 프로그램에 적용된 DEP 기법을 우회하여 공격자가 원하는 코드를 실행시키는 공격 기법이다. ROP 공격은 오늘날 가장 많이 이용되는 공격 중 하나이며, 시스템 함수를 이용하여 DEP를 비활성화시켜 Code-Injection 공격을 다시 가능하게 만드는 것도 가능하다. ROP 공격의 방어 기법에 관한 연구는 꾸준히 이루어지고 있으며, ROP를 막기 위해 코드 재작성, 하드웨어 지원, 메모리 격리(isolation), ASLR(Address Space Layout Randomization) 등 많은 방법이 사용되고 있다. 따라서 본 논문에서는 이러한 ROP 공격의 방어 기법에 관한 연구들이 어떤 메커니즘으로 공격을 막는지 비교 분석하고, 각 연구의 한계점에 관해 서술한다.

I. 서론

데이터영역의 실행을 방지하는 DEP(Date Execution Prevention) 기술이 현대의 운영체제에 내장되면서 데이터영역에 코드를 삽입하여 실행하는 Code-Injection 공격은 매우 어려워졌다. 그러나 공격자들은 새로운 방어 기법에 맞서, 코드를 삽입하는 것이 아닌 프로그램 내부에 존재하는 코드를 사용하는 기법을 고안하여 DEP를 우회하였다.

코드 재사용 공격은 오늘날까지도 빈번히 사용되고 있으며, 특히 코드 재사용 공격의 한 종류인 ROP(Return-Oriented Programming) 공격은 가장 많이 사용되고 있는 공격 중 하나이다[1]. 이는 프로그램 내부에 존재하는 짧은 코드 조각을 return 명령어를 통해 엮어 악의적인 실행을 하는 페이로드를 만드는 기법으로, 프로그램의 제어 흐름을 탈취하거나, DEP의 기능을 비활성화시켜 다시 Code-Injection 공격을 가능하게 만드는 등, 프로그램에 치명적인 공격으로 이어진다[6]. 이에 따라 ROP 공격의 방어 기법에 관한 연구는 꾸준히 진행됐으며 현재까지 많은 방어 기법들이 제안되었다[2][3][4][5]. 본 논문에서는 이러한 ROP 공격의 방어 기법에 대하여 분석하고, 각각의 한계점에 관하여 다루려고 한다.

* 이 성과는 2018년도 과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1A2B3009643).

II. 배경지식

2.1 Return Oriented Programming

ROP 공격은 코드 재사용 공격의 한 종류로 짧은 코드 조각들을 엮어 공격자가 원하는 명령을 수행하는 공격이다. 공격 과정은 먼저 ret 명령어로 끝나는 짧은 코드 조각들을 엮어 공격 페이로드를 만들고, 생성한 페이로드를 프로그램에

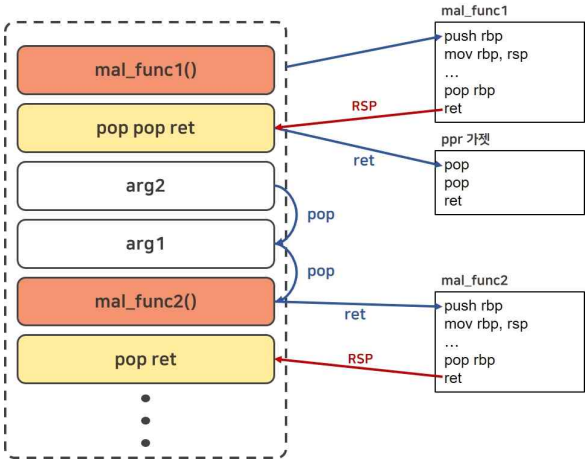


Fig. 1. ROP 공격의 페이로드

삽입한 뒤 스택 포인터를 조작하여 삽입된 페이로드가 실행되도록 한다. 이때 사용되는 짧은 코드 조각은 gadget이라 하며 스택 포인터의 값을 조작하여 페이로드를 실행시키는 것을 stack pivoting이라고 한다.

ROP 공격의 페이로드 생성 과정에서 각 gadget은 ret 명령어를 통해 차례로 실행된다. 즉, gadget의 반환 주소 위치에 다음으로 실행될 gadget의 주소를 넣어 ret 명령어가 실행되었을 때 다음 gadget이 실행되는 것이다. 또한, 페이로드 상에서 함수를 호출했을 때 사용한 인자는 ppr gadget을 이용하여 정리한다. ppr gadget은 pop-pop-ret 을 수행하는 코드 조각으로 필요에 따라 pop의 개수를 늘리고 줄일 수 있다.

일반적인 ROP 공격의 페이로드는 Fig. 1.과 같다. stack pivoting을 통해 페이로드를 실행시키면 가장 첫 번째로 mal_func1()이 수행되며 mal_func1()에는 두 개의 인자값 arg1, arg2가 스택을 통해 전달된다. mal_func1()의 반환 주소에는 ppr gadget의 주소가 들어있다. 따라서 mal_func1()이 종료되고 ret 명령이 실행되면 ppr gadget이 실행된다. arg1과 arg2은 두개의 pop 명령어에 의하여 스택에서 정리되고, ret 명령을 통하여 mal_func2()가 실행된다. ROP 공격에서는 이러한 일련의 과정을 통해 페이로드가 실행되며, 따라서 공격자는 자신이 원하는 것을 획득할 수 있다.

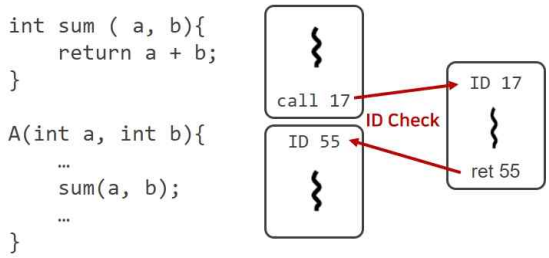


Fig. 2. CFI 기법의 기본 원리

III. 반환지향형 프로그래밍에 대한 방어 기법

3.1 제어 흐름 무결성 기법 (Control Flow Integrity, CFI)

CFI 기법은 프로그램의 유효하지 않은 제어 흐름을 제한하여 ROP 공격을 막는 기법이다 [2]. CFI 기법은 두 단계로 구성이 된다. 첫 번째 단계에서는 제어 흐름의 유효성에 관한 판단 기준을 정하기 위하여 정적분석을 통해 제어 흐름 그래프(Control Flow Graph, CFG)를 생성한다. 두 번째 단계에서는, 앞서 구한 CFG를 기반으로 분기문의 목적지에 ID를 삽입하고 분기분이 발생하는 지점에는 ID 검사 코드를 삽입한다. 따라서 Fig. 2와 같이 분기문이 발생하여 대상 주소로 제어 흐름이 이동하기 전에 대상 주소의 ID를 확인한다. ID가 일치하면 제어 흐름이 이동되지만, ID가 일치하지 않는다면 프로세스를 중단시켜 프로그램의 제어 흐름에 대한 무결성을 보장한다.

ROP 공격에서는 공격자가 반환 주소를 조작한다. 이 때문에 ret 명령어가 실행되면 정상적인 실행의 반환 주소가 아닌 공격자가 원하는 곳의 주소로 프로그램의 제어 흐름이 넘어가게 된다. 따라서 CFI 기법이 적용된 프로그램에서 변조된 주소는 ID 값이 주지 않았거나, 주어졌더라도 일치하지 않기 때문에 분기문이 발생하는 지점의 검사 코드에 의하여 공격이 탐지된다.

3.2 스택 포인터 무결성 기법 (Stack Pointer Integrity, SPI)

스택 포인터 무결성(Stack Pointer Integrity,

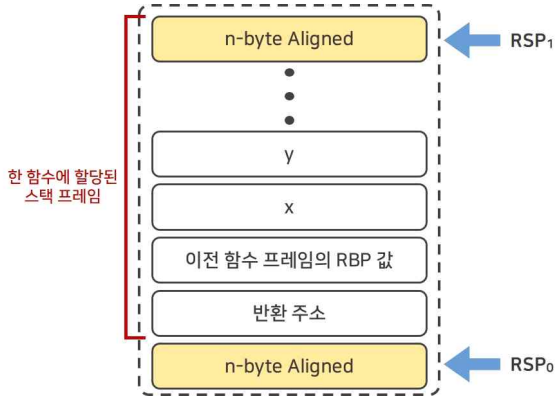


Fig. 3. SPI기법이 적용된 프로그램의 스택구조

SPI) 기법의 기본 아이디어는 다음과 같다[3]. 먼저 함수가 실행되었을 때 할당되는 스택 프레임의 크기를 n 바이트의 배수로 제한한다. 따라서 Fig. 3.과 같이 스택 포인터는 항상 n의 배수인 주소만을 가리키게 되며 반환 주소는 n의 배수인 주소 바로 상단에 위치하게 된다. 그러므로 ret 명령어가 실행되면 스택 포인터의 값을 검사해 올바른 위치(n의 배수인 주소) 상단에 있다면 ret 명령을 수행하고 아니라면 ROP 공격으로 판단하여 프로그램을 종료시킨다.

ROP 공격은 반복적인 ret 명령어를 통해 페이로드를 수행한다. Fig. 1.과 같은 페이로드로 공격을 한다고 가정하였을 때, 페이로드의 시작을 n의 배수인 주소에 삽입한 경우 stack pivoting은 가능하다. 하지만 다음 gadget을 실행시키기 위해 ret 명령어가 수행될 때는 스택 포인터가 n의 배수 상단에 위치하지 않기 때문에 SPI 정책에 의하여 공격이 탐지되어 프로그램이 종료된다. 공격자는 ROP 공격을 탐지할 수 없도록 페이로드에 더미값을 삽입하여 ret 명령어가 발생할 때의 위치를 n의 배수 상단의 값으로 정렬시킬 수 있다. 이때, n의 배수로 정렬된 다음 gadget을 실행시키기 위해서는 삽입한 더미값들을 스택 내에서 꺼내야 한다. 따라서 더미값의 개수만큼 pop 명령어를 포함하는 gadget을 찾아야 하지만 이러한 gadget을 찾기는 매우 어렵다. 그러므로 위와 같은 방법으로 SPI를 우회하는 것은 현실적이지 않다.

3.3 KBouncer

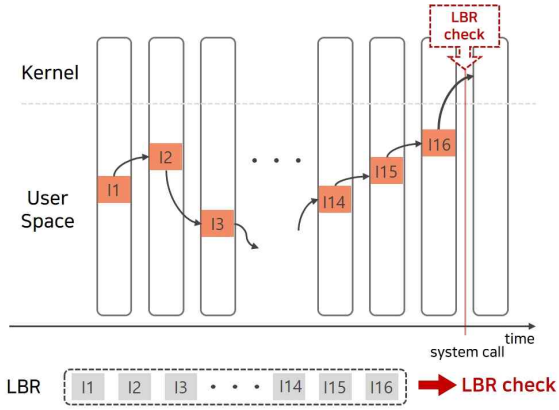


Fig. 4. Kbouncer 방어 기법

KBouncer는 프로그램에서 발생하는 간접 분기문을 모니터링하여 ROP 공격을 탐지하는 기법이다[4]. KBouncer는 간접 분기문 조사를 위해 Last Branch Record(LBR) 이라는 인텔 하드웨어를 사용한다. 또한, 이 기법은 커널 모드에서 검사가 진행되기 때문에 검사를 위해서는 유저 모드와 커널 모드 간에 전환(context switching)이 필요하다. 따라서 모드 전환의 오버헤드를 줄이기 위해 시스템 함수가 호출되었을 때만 검사를 진행한다.

Fig. 4.와 같이 LBR에는 호출된 간접 분기문 $Ii\,(1 \le i \le 16)$가 저장된다. 시스템 함수가 호출되면 KBouncer에서는 해당 호출이 ROP 공격으로 발생한 호출인지 검사하기 위하여 LBR에 저장된 가장 최근 실행한 16개의 간접 분기문 Ii를 대상으로 유효성을 검사한다. 이때, 유효성이란 간접 분기문이 gadget으로 사용되었는가에 대한 여부로, 두 가지 요소를 검사한다. 첫 번째는 반환 주소가 호출이 선행된 주소인지 검사한다. 일반적인 실행에서 반환 주소는 call 명령어의 바로 다음 주소이다. 하지만 ROP 공격의 경우 반환 주소를 조작하여 공격자가 원하는 곳을 실행시키기 때문에 위와 같은 성질을 항상 만족하지 않는다. 따라서 16개의 간접 분기문에 대해 각각의 반환 주소가 위의 성질을 만족하지 않다면 ROP 공격으로 탐지한다. 두 번째는 간접 분기문을 통해 실행된 명령어의 개수로 판단한다. ROP 공격에서 gadget 내부의 명령어 개수는 일반적인 실행에서 간접 분기문을 통해 실행되는 명령어의 개

수보다 적다. 이 때문에 Kbouncer는 간접 분기문을 통해 실행되는 명령어의 개수가 20개 이하라면 해당 간접 분기문은 gadget일 가능성이 있다고 판단한다. 따라서 총 16개의 간접 분기분 중 8개 이상이 gadget일 가능성이 있다고 판단되었을 때 ROP 공격으로 탐지된다.

IV.분석

앞서 소개한 연구들은 ret 명령어를 통하여 gadget들을 조합해 공격자가 원하는 악의적인 실행을 하는 ROP 공격을 효과적으로 막아낸다.

그러나 CFI와 SPI 기법의 경우 코드 재작성을 통해 각 기법을 프로그램에 적용하기 때문에, 프로그램에 동적으로 로드되는 API나 라이브러리 등에는 기법을 적용할 수 없다. 즉, 동적으로 로드되는 API나 라이브러리에 존재하는 코드를 이용하여 페이로드를 생성하였을 때는 ROP 공격을 성공적으로 수행할 수 있다. 이는 코드 재작성을 통한 기법 적용의 한계이며, CFI와 SPI의 치명적인 약점이라고 볼 수 있다[8].

KBouncer의 경우 코드 재작성을 하지 않고, 시스템 함수가 발생하였을 때 kernel 모드를 통하여 LBR에 저장된 간접 분기문을 검사한다. 따라서 위와 같은 한계가 존재하지 않으며, 동적으로 로드되는 API나 라이브러리 내부에 있는 코드로 페이로드를 작성하더라도 공격을 탐지할 수 있다. 하지만 KBouncer 또한 한계가 존재한다. KBouncer는 간접 분기문에 대해 두 가지 검사를 수행하여 gadget인지 판단한다. 따라서 두 가지 성질을 모두 만족하는 gadget을 사용한다면, 공격자는 성공적으로 공격을 수행할 수 있다[4]. 즉, 호출이 선행되는 주소를 갖고 있으며, gadget의 명령어 개수가 20개를 넘는다면 이 gadget은 정상적인 간접 분기분으로 판별된다. 따라서 이런 구조로 되어 있는 gadget들을 찾아 사용한다면 KBouncer또한 우회가 가능해진다.

V.결론

본 논문에서는 ROP 공격의 방어 기법에 관한 연구들을 비교 분석하였다. 각 연구에서는 프로그램의 흐름을 검사하거나, 스택 프레임 구조를 정형화하여 ROP 공격에 대응한다. 또는 시스템 함수로 이어지는 간접 분기문이 가젯인지 검사하는 메커니즘을 이용하여 ROP 공격에 대응하기도 한다. 그러나 각 연구에는 한계가 존재하고 이 한계들로 인해 완벽한 방어는 불가능하다.

추후 연구에서는 KBouncer 기법과 제어 흐름 무결성 검증기법을 결합하여 각 연구의 한계를 줄이는 방법에 관하여 연구할 예정이다.

[참고문헌]

[1] H. Shacham. "The Geomerty of Innocent Flesh on the Bone : Return-into-libc without Function Calls (on the x86)" In Proc. of ACM CCS, pp 340-353, 2007

[2] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti, "Control-Flow Integrity," In Proc. of ACM CCS, pp 340-353, 2005.

[3] A. Quach, M. Cole, and A. PraKash "Supplementing Modern Software Defenses with Stack-Pointer Sanity" In Annual Computer Security Applications Conference, pp116-127 2017.

[4] V. Pappas, M. Polychronakis, and A. D Keromytis, "Transparent ROP Exploit Mitigation using Indirect Branch Tracing" In Proc. of USENIX Security Symposium, pp 447-462, 2013

[5] Y. Cheng, Z. Zhou, M. Yu, X. Ding, and R. H. Deng "ROPecker: A Generic and Practical Approach for Defending Against ROP Attacks" In Symposium on Network and Distributed System Security (NDSS), 2014.

[6] E. Göktas, E. Athanasopoulos, H. bos, and G. Portokalidis "Out Of Control: Overcoming Control-Flow Integrity" In IEEE Symposium on Security and Privacy(S&P), 2014

macOS 커널에서의 보호기법과 우회방법 분석†

이유찬* 권태경*

*연세대학교 정보대학원 정보보호 연구실

Mitigations and Bypass methodology Analysis on macOS Kernel

Yoo Chan Lee* Tae Kyoung Kwon*

*Information security Lab, Graduation School of Information, Yonsei University.

요 약

macOS는 애플 사의 데스크탑과 랩탑에서 사용하는 운영체제로 XNU 커널을 사용하고 있다. XNU 커널은 Mach, BSD, IOKIt으로 구성되어 있는데 최근 XNU 커널에 대한 취약점 연구가 활발히 진행되고 있다. macOS에서는 커널 취약점에 대한 패치와 취약점이 존재하더라도 익스플로잇을 어렵게 하기위한 보호기법들이 적용되어있는데 본 논문에서는 이 보호기법들의 동작 원리와 이러한 보호기법들의 우회방법에 대하여 연구하여 학술적으로 정리하였다. 보호기법들은 많은 취약점들을 무력화시키지만 아직까지 많은 취약점이 보호기법을 우회하여 익스플로잇이 가능하다. 따라서, 보호기법들의 강화 혹은 새로운 보호기법을 통해 취약점이 발생하더라도 공격자가 원하는 행위를 어렵게 만들어야한다.

I. 서론

과거의 소프트웨어는 보안에 신경 쓰지 않고 기능만 고려하여 개발하였기에 많은 보안 위협에 노출되어 있었다. 특히 Buffer Overflow, Format String Bug, Command Injection, Use After Free 등의 메모리 오염 취약점은 공격자로 하여금 원격 코드 실행 (Remote Code Execution), 정보 노출 (Information Leakage), 권한 상승 (Privilege Escalation) 등을 가능하게 하여 심각한 결과를 초래하였고 악성코드로부터 시스템 파일이 수정되어 컴퓨터에 문제를 발생시켰다.

현대의 소프트웨어에서는 공격자의 악성 행위로부터 사용자를 보호하고, 보안 위협에서 벗어나기 위해 여러 가지 보호기법을 개발하여 사용하고 있다. 보호기법이란, 프로그램에 취약점이 존재하더라도, 익스플로잇을 어렵게 하여 공격자가 원하는 결과를 얻지 못하게 막는 장치를 말한다. 커널에서 사용하는 보호기법은 코드에서 발생하는 메모리 오염 공격으로부터 방어하기 위한 메모리 보호기법과 공격자가 루트킷 같은 악성코드를 심기 위해 파일 시스템을 수정하는 것을 막기 위한 파일 시스템 보호기법이 있다. 또한, 보호기법은 운영체제별로 다르게 적용되어 사용되고 있기 때문에 같은 기능을 하더라도 명칭이 다른 경우가 있다.

본 논문에서는 Apple 사의 데스크탑과 랩탑에서 사용되고 있는 macOS 운영체제에서 사용되고 있는 커널 보호기법에 대해서 알아보고 각 보호기법의 우회법에 대하여 분석할 것이다.

II. 배경 지식 및 관련 연구

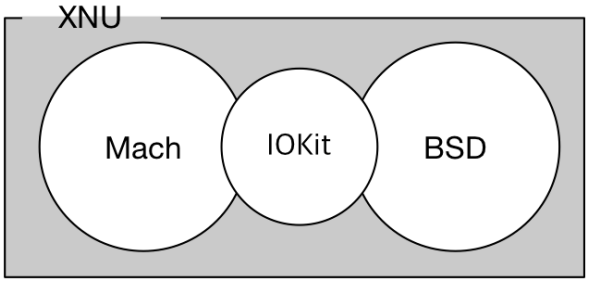


그림 1 - XNU 구조

†이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00513,기계학습을 활용한 UNIX 기반 커널 취약점 탐지 자동화 연구)

2.1 XNU

XNU는 애플 사의 장비에서 사용하는 운영체제 macOS, iOS, watchOS, tvOS에서 사용하고 있는 커널이다. XNU는 그림 1과 같이 Mach core와 Berkeley Software Distribution (BSD), IOKit으로 구성되어있다. XNU에서 Mach는 멀티 태스킹, 메모리 관리, IPC 등을 관리하고, BSD는 기본 보안 정책, 네트워크, 파일 시스템을 담당하고 있다. 마지막으로 IOKit은 장치 드라이버를 만들기 위한 시스템 프레임워크, 라이브러리, 기타 필요 자원들의 집합이다.

2.2 커널 취약점 탐지 연구

최근 해외 학계에서 커널 취약점을 탐지하는 방법에 대해 관심을 보이고 있다. 퍼징을 통해 동적인 방식으로 취약점 탐지를 자동화 하는 방법 [1]과 커널에서 발생하는 취약점 유형을 정의하여 정적인 방법으로 취약점을 찾는 방법 [2]이 계속해서 연구되고 있다. 또한, 커널 취약점 탐지를 할 때 가장 큰 난제인 복잡한 구조와 방대한 크기를 해결하기 위해 후킹 [3], Symbolic Execution [4] 등을 이용해 커널 코드를 호출하는 방법을 추출하여 퍼징을 하는 방식이 연구되고 있다.

III. 보호기법과 우회방법

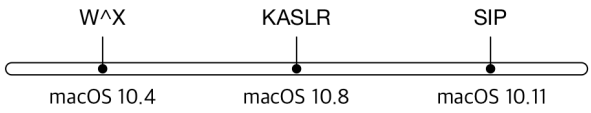


그림 3 - macOS 보호기법 적용시기

macOS의 보호기법에는 커널 메모리를 보호하기 위한 KASLR, W^X과 파일 시스템을 보호하기 위한 SIP가 있다. 또한, CPU에 따라 SMAP 과 SMEP이 적용되어있다. 그림 2를 통해 최근 5년간 macOS 취약점이 많이 연구되고 패치된 것을 확인할 수 있다. 이를 통해 macOS에 대한 연구가 많이 증가하고 보호기법이 적용된 후에도 많은 취약점이 존재하는 것을 알 수 있다.

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	1	1													
2001	5		1								1	1			
2002	20	7	8	3								2			
2003	25	3	5	6								3			
2004	54	4	11	8				1		2	2	5			
2005	96	13	28	20	1		1	1		5	3	9			
2006	106	34	48	34	5		1	1		6	4	5			
2007	110	33	48	30	11		3	2	1	11	9	18			
2008	95	31	43	21	8		3	2		9	13	6			
2009	81	44	43	30	13		3		1	4	5	7			1
2010	97	56	52	42	15		3	3		9	5	3	1		
2011	74	35	36	29	9		1			8	8	2			
2012	37	21	20	13	4					1	6	2			
2013	72	30	15	13	4					15	15				
2014	150	54	73	42	20					25	24	6			1
2015	444	225	180	184	146		2	1		53	61	63	1		3
2016	215	149	109	114	96		1			6	40	21			
2017	299	189	158	135	117		2			40	54	9			
2018	107	41	49	34	30					24	18	2			
Total	2088	970	927	758	479		20	11	2	218	268	164	2		5
% Of All		46.5	44.4	36.3	22.9	0.0	1.0	0.5	0.1	10.4	12.8	7.9	0.1	0.0	

그림 2 - cvedetails.com에 등록된 macOS 연도별 취약점 갯수

3.1 KASLR

KASLR은 Kernel Address Space Layout Randomization의 약자로 macOS 10.8에서 처음 적용되었다. KALSLR은 Use-After-Free, Buffer Overflow와 같은 메모리 오염 취약점을 통해 시도하는 코드 재사용 공격을 방지하기 위해 사용되는 메모리 보호기법이다. macOS에서 KASLR은 커널 영역의 기준 메모리 주소에서 랜덤 값을 더하여 고정된 주소가 아닌 유동 주소로 만들어 커널이 부팅될 때마다 커널 영역의 바이너리, 모듈 , 스택 등의 주소를 변경한다. 이 때, KASLR이 적용된 메모리 주소 - 기준 메모리 주소의 값을 kslide라고 부른다. 이 보호기법을 통하여 ROP나 RTL 같은 코드 재사용 공격의 안정성을 낮추어 공격을 어렵게 한다. KASLR은 macOS에서 기본 설정으로 적용되어 있다. nvram 명령어를 통해 boot-args 변수에 kslide=0을 입력해주면 KASLR을 적용하지 않을 수 있다.

우회방법. 유저 영역의 프로세스에서 매번 프로세스 실행 시마다 적용되는 ASLR과 다르게 KASLR은 부팅이 될 때마다 KASLR이 적용되어 커널 영역 메모리 주소가 바뀐다. macOS의 경우 모든 커널 바이너리와 모듈이 하나의 kslide를 이용하기 때문에 정보 노출 취약점을 통해 커널 코드 영역 메모리 주소를 얻게 되면 KASLR에 의해 바뀐 커널 바이너리 메모리 주소와

커널 모듈들의 메모리 주소를 알 수 있다. 64bit 컴퓨터가 사용된 이후로 무작위 공격을 통해 성공할 수 있는 확률이 매우 낮아지게 되었다. 그 결과, 공격자가 공격 할 때 반드시 정보 노출 취약점을 통해 KASLR을 우회한 뒤 공격을 해야 한다.

3.2 W^X

W^X는 macOS 10.4부터 적용된 메모리 보호정책으로 기존에 읽기, 쓰기, 실행 권한이 있는 메모리 페이지에 원하는 명령어를 어셈블리어로 작성 한 뒤, 해당 페이지에 입력하고 실행하는 방식으로 쉽게 커널 영역에서 권한 상승을 발생 시켰다. 이를 방지하기 위하여 커널 영역의 모든 메모리 페이지에서 쓰기 권한과 실행 권한을 동시에 줄 수 없게 막는 보안 정책이다. 이 정책은 커널 모듈뿐만 아니라 스택과 힙 영역에서도 적용된다. 이 보호기법은 일반적으로 유저 영역 메모리에서 사용되고 있는 Data Execution Protection (DEP)와 기능이 같다.

우회방법. ROP 공격과 같은 코드 재사용 공격을 통해서 쉽게 우회가 가능하다.

```
Lee:~ yoochanlee$ sysctl machdep.cpu.leaf7_features
machdep.cpu.leaf7_features: SMEP ERMS RDWRFSGS TSC_THRE
AD_OFFSET BMI1 HLE AVX2 BMI2 INVPCID RTM SMAP RDSEED AD
X IPT SGX FPU_CSDS MPX CLFSOPT
```

그림 4 - sysctl 명령어를 통한 SMAP/SMEP 확인

3.3 SMAP/SMEP

SMAP/SMEP은 Supervisor Mode Access Protection과 Supervisor Mode Execution Protection의 약자로 CPU가 커널 모드로 동작할 때, 유저 영역 메모리에 접근하거나 실행하는 것을 막는 보호기법이다. SMAP/SMEP은 CPU 레벨에서 동작하는 보호기법으로 CR4 레지스터의 21번 bit와 20번 bit에 SMAP/SMEP의 실행 상태에 대한 정보가 들어있다. 기존에 *ret2usr* 공격을 통해 코드 실행 취약점이 있을 때, 커널 모드로 유저영역 코드를 실행하는 것을 막기 위해 개발되었다. 이 보호기법을 통해 커널 취약점을 이용해 유저 영역에 접근하거나 실행하는 것이 불가능해졌다. 그림 3에 나와있는 명령어를 통해 사용하고 있는 CPU에서 SMAP/SMEP을 지원하는 지 알 수 있다.

우회방법. 커널 영역 코드 실행 취약점을 통해 CR4 레지스터를 조작하여 SMAP/SMEP에 해당하는 bit를 수정하여 이 보호기법을 우회할 수 있다.

3.4 SIP

SIP는 System Integrity Protection의 약자로 rootless라고도 불리며, macOS El Capitan에서부터 적용된 파일 시스템 보호기법이다. SIP는 악성 프로그램으로부터 악성 커널 모듈을 실행시키거나, 중요한 시스템 파일이나 폴더의 변경을 보호하기 위해 설계되었다. 기존에는 root 권한이 있으면 커널 컴파일, 커널 모듈 로드, 시스템 파일 수정 등 모든 기능을 수행할 수 있었으나, SIP가 macOS에 적용된 이후로, root 권한을 획득하더라도 SIP로 인해 별다른 행위를 하지 못하게 되었다. 정상적인 방법으로 SIP를 끄기 위해서는 복구 모드에 들어가서 그림 3의 명령어를 입력해야한다.

```
Lee:~ yoochanlee$ csrutil
usage: csrutil <command>
Modify the System Integrity Protection configuration. All configuration changes
apply to the entire machine.
Available commands:

    clear
        Clear the existing configuration.
    disable
        Disable the protection on the machine. Only available in Recovery OS.
    enable
        Enable the protection on the machine. Only available in Recovery OS.
    status
        Display the current configuration.

    netboot
        add <address>
            Insert a new IPv4 address in the list of allowed NetBoot sources.
        list
            Print the list of allowed NetBoot sources.
        remove <address>
            Remove an IPv4 address from the list of allowed NetBoot sources.
```

그림 5 - csrutil 명령어

우회방법. SIP의 on/off 상태에 대한 정보가 커널 영역 메모리에 저장되어 있기 때문에, 커널 영역 코드 실행 취약점을 통해 해당 커널 메모리를 수정하여 이 보호기법을 우회할 수 있다.

IV. 결론 및 향후 연구 과제

본 논문에서는 현재 macOS에 적용된 보호기법들에 대한 원리와 각 보호기법의 우회 방법에 대해서 분석하였다. 보호기법은 많은 취약점으로부터 익스플로잇을 방어해 주지만, 현재도 보호기법들을 우회하기위한 정보 노출 취약점과 커널 영역 코드 실행 취약점이 계속해서 등장하고 있으며, 제조사는 취약한 코드 패치를 통해 취약점을 막는 임기응변식 대처를 하고 있다. 그러나, 제보되지 않은 취약점은 패치할 수 없고 새로운 기능이 개발되거나 새로운 공격 벡터가 등장하여 새로운 취약점이 계속해서 발생할 것이기 때문에 기존 보호기법의 강화 혹은 새로운 보호기법을 통하여 취약점이 발생하더라도 공격자가 원하는 행위를 할 수 없게 해야 한다.

[참고문헌]

[1] JEONG, Dae R., et al. Razzer: Finding Kernel Race Bugs through Fuzzing. In: Razzer: Finding Kernel Race Bugs through Fuzzing. IEEE. p. 0.

[2] WANG, Wenwen; LU, Kangjie; YEW, Pen-Chung. Check It Again: Detecting Lacking-Recheck Bugs in OS Kernels. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018. p. 1899-1913.

[3] HAN, HyungSeok; CHA, Sang Kil. IMF: Inferred Model-based Fuzzer. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017. p. 2345-2358.

[4] YUN, Insu, et al. QSYM: a practical concolic execution engine tailored for hybrid fuzzing. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). USENIX} Association}, 2018. p. 745-761.

소프트웨어 취약점을 이용한 자동 exploit 생성 모듈 분석

이상무, 김주환, 윤주범
세종대학교 정보보호학과

Automatic software exploit generation module analysis

Sangmoo Lee, Juhwan Kim, Joobeom Yun
Department of Computer and Information Security, Sejong University.

요 약

현대의 소프트웨어 취약점 분석은 대부분 전문가의 개인 역량에 의해 수동으로 수행되고 있다. 하지만 수동적인 취약점 분석은 많은 비용과 시간이 소모된다는 한계점을 가지고 있다. 이에 따라 소프트웨어 취약점 분석을 자동화하는 필요성이 대두되고 있다. 본 논문에서는 미국의 DARPA에서 진행한 소프트웨어 취약점을 이용한 자동화 공격 방어 시스템 대회인 Cyber Grand Challenge에서 3위를 차지한 shellphish팀의 시스템을 분석한다. 이 시스템의 모듈 중 취약점을 이용하여 exploit을 자동 생성하는 rex를 집중 분석하며, 각 exploit 생성 기법의 특징과 구현 방법에 대해 분석하고자 한다.

I. 개요

현대의 취약점 분석은 대부분 전문가의 개인 역량에 의해 수동으로 수행되고 있다. 하지만 수동적인 취약점 분석은 많은 비용과 시간이 소모된다는 한계점을 가지고 있다. 이에 따라 소프트웨어 취약점 분석을 자동화하는 필요성이 대두되고 있다. rex[1]는 여러 기법을 사용하여 바이너리를 분석하는 angr[2] 라이브러리와 exploit 개발을 위한 라이브러리인 pwntools[3]를 결합한 자동화 exploit 생성 모듈이다. 이는 [그림 1]과 같이 대상 바이너리와 대상 바이너리의 크래시(Crash)를 유발하는 입력 값을 분석하여 크래시의 종류를 구분하며, 이에 따른 여러 가지 exploit 기법을 대입시켜 exploit code를 자동으로 작성한다.

rex에서 사용하는 exploit 생성 기법에는 총 8가지 기법이 존재한다. 이는 대상 바이너리의 구조와 크래시 정보에 기반하여 여러 기법의 적용 여부를 순차적으로 검사하며, 적용이 가능한 기법에 대해서 개별적인 exploit code를 생성한다. <표 1>과 같이 5가지 기법들은 CGC(Cyber Grand Challenge) 대회용 바이너리를 위한 exploit 기법이며, 3가지 기법들은 유닉스 기반의 ELF 바이너리를 위한 exploit 기법이다.

II. 관련 연구

2.1 Driller

Driller[4]는 AFL[5] 퍼저(Fuzzer)와 angr 라이브러리를 결합한 시스템으로, 소프트웨어의 크래시를 유발시키는 시스템이다. 이는 퍼저를 이용하여 바이너리에 무작위 값을 계속해서 입력함으로써 바이너리를 탐색하며, 퍼저가 특정 구획으로 탐색하기 어려운 조건문이 있을 때만 angr를 호출한다. 이 때, 특정 구획을 탐색할 수 있는 입력 값을 생성하기 위해 angr의 symbolic execution 기법을 사용하며, 생성된 입력 값을 퍼저에게 전달한다.

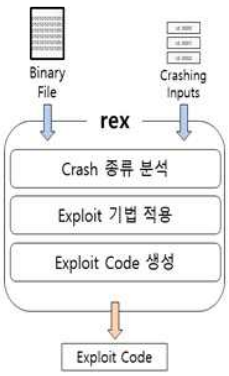


그림 1 rex overview

표 1 Exploit techniques

CGC	UNIX
circumstantial set register	call to shellcode
rop leak memory	call to JMP SP shellcode
rop set register	ROP to system
shellcode leak address	
shellcode set register	

2.2 T-Fuzz

T-Fuzz[6]는 대상 바이너리의 체크섬 검사, 해시 검사 등과 같은 복잡한 검사로 인해 퍼저가 특정 구획을 탐색할 수 없는 단점을 효과적으로 개선한 시스템이다. 이는 먼저 탐지율을 높이기 위해 Non-Critical 검사 영역과 TCP Packet Parser 등과 같은 Critical 검사 영역을 분류하여 Non-Critical 검사 영역을 제거한다. 이후 퍼저의 코드 커버리지를 높이기 위해 바이너리의 조건문을 찾아 반대되는 조건식으로 수정한다.

이러한 접근법으로 Driller와 T-Fuzz는 퍼저가 바이너리를 더 깊게 탐색할 수 있도록 하였으며, 크래시를 발생시킬 가능성을 높였다. 하지만 이들은 소프트웨어에 크래시를 유발할 수 있게만 목적을 둔 시스템이며, 더 나아가 크래시를 이용하여 시스템을 공격할 수 있는 exploit 생성 기법에 대해서는 언급되지 않는다. 이에 본 논문에서는 앞서 말했듯이, 소프트웨어 크래시를 이용하여 자동으로 exploit을 생성하는 rex 모듈을 분석한다.

III. Exploit 생성 기법

3.1 Return to Shellcode

Return to Shellcode는 소프트웨어 취약점을 이용한 가장 기본적인 exploit 생성 기법이다. 스택(Stack) 내에 저장된 버퍼의 크기보다 큰 값의 입력값을 복사하여 반환 주소(return address)가 변조되는 취약점인 스택 버퍼 오버플로우(Buffer Overflow)가 선행되어야 한다. 스택 상 반환 주소를 쉘 코드(Shell Code)의 시작 주소로 변조하면, 의도된 명령어의 집합인 쉘 코드가 실행되고 쉘 코드는 root 권한의 쉘을 실행시킨다. RTS Exploit 기법은 DEP(Data Execution Prvention)에 의해 방지된다. DEP는 현대 운영체제가 지원하는 메모리 보호 기법 중 하나이며, 데이터를 저장하는 스택 영역의 실행권한을 막는 것으로 스택 상에 저장된 쉘 코드가 실행되는 것 자체를 방지한다.

3.2 Return to Libc

RTS Exploit 기법이 DEP에 의해 방지되면서 이를 우회하기 위해 새로 고안된 exploit 생성 기법이 Return to Libc이다. Return to Libc는 DEP가 설정된 운영체제에서도 실행 가능한 메모리 영역인 공유

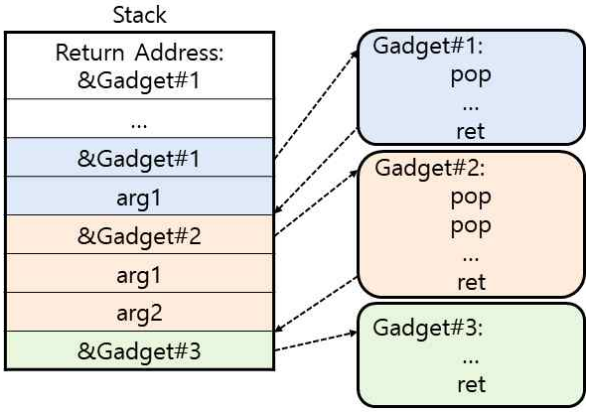


그림 2 ROP attack process

라이브러리 영역에서 원하는 명령어를 실행할 수 있도록 반환 주소를 변조하는 기법이다. RTS 공격 기법과 마찬가지로 스택 버퍼 오버플로우가 선행되어야 하며, 반환 주소를 공유 라이브러리 영역의 system() 함수 주소를 가리키게 함으로써 root 권한의 쉘을 얻을 수 있게 한다.

3.3 Return Oriented Programming

Return Oriented Programming은 [그림 2]와 같이 RTL 공격 기법에 Gadget이라는 개념을 결합한 공격 기법이다. Gadget은 공격 대상 프로그램이 사용하는 메모리에 이미 존재하는 코드 조각을 의미한다. ROP는 RTL과 Gadget을 이용하여 공격에 필요한 코드를 새로 프로그래밍 할 수 있는 공격 기법이다. 공격 대상 바이너리에서 유용할 가능성이 있는 명령어들(Gadgets)을 찾고 버퍼 오버플로우를 통해 첫 번째 Gadget의 명령어의 주소로 반환 주소를 겹쳐씀으로써 묶일 수 있다. 이후 Gadgets의 첫 번째 주소를 이후 스택에 저장한다. 첫 번째 Gadget이 끝날 때, ret 명령어가 실행되며, 이는 다음 Gadget의 주소를 pop하고 그 지점으로 jmp하게 된다. 두 번째 Gadget이 끝날 때 ROP Chain은 세 번째 Gadget과 함께 계속된다. 작은 명령어 순서 열을 묶음으로써 공격자는 임의적인 프로그램 행동을 이미 존재하는 라이브러리 코드에서 새로 만들어 연속적으로 수행할 수 있게 된다. RTL 공격 기법은 공유 라이브러리 상 위험한 함수를 제거함으로써 공격이 제한될 수 있지만, ROP 공격 기법은 Exploit 방어 기법 중 어느 것을 적용해도 효과적이지 않다.

IV. rex 모듈 접근법 분석

4.1. 크래시 분류 과정

rex 모듈의 시작점이 되는 Crash 클래스는 <표 2>와 같은 인자를 받는다. 인자 값의 대부분은 크래시 분류 과정에서 스스로 초기화가 되는 정보이며, 넘겨줘야 하는 주요한 인자는 'binary'와 크래시를 발생시키는 입력 값인 'crash'이다. 예를 들어, 크래시를 발생시키는 입력 값은 버퍼 오버플로우가 발생하여

표 2 Arguments information for class Crash

argument	description
binary	exploit을 수행할 대상 바이너리의 경로
crash	대상 바이너리의 크래시를 발생시키는 입력 값
pov_file	대상 바이너리의 포맷이 CGC일 경우, Proof of Vulnerability(PoV) 정보
aslr	대상 바이너리의 ASLR(Address Space Layout Randomization) 설정 상태
constrained_addrs	바이너리 탐색 과정에서 제한될 주소 값 목록
crash_state	이미 추적된 크래시 상태의 주소 지정
prev_path	크래시가 발생하는 경로에 대한 정보
format_infos	크래시를 분석할 때, 쓰인 포맷에 대한 정보
rop_cache_tuple	angr 모듈의 ROP 기능에 쓰이는 튜플
use_rop	ROP를 사용할지의 여부
explore_steps	이미 바이너리를 탐색한 단계의 수
angrop_object	angrop 모듈에 관한 객체이며, 이 객체로 탐색 기능을 수행할 수 있음
argv	부가적인 특정 입력 값 지정(i.e, ['./calc', 'param1'])
concrete_fs	바이너리 분석을 위해 호스트의 파일 시스템을 사용
trace_timeout	Trace 모듈의 수행 시간을 설정

반환 주소를 변조시키는 입력 값을 의미한다.

CGC 포맷의 바이너리는 취약점 자체에 초점을 맞출 수 있도록 제작되었기 때문에, 본 논문에서는 CGC 포맷의 바이너리를 대상으로 적용한 exploit 생성 기법보다는 [그림 3]과 같이 사용자가 일반적으로 사용하는 UNIX 시스템의 구조를 가진 바이너리를 대상으로 적용되는 exploit 기법을 분석하였다. rex 모듈의 자동화 exploit 생성에서 사용되는 기법들은 공통적으로 스택 버퍼 오버플로우를 통해 명령 포인터(instruction pointer)를 조작할 수 있어야 한다. 기존의 exploit 생성 방법에서 수동으로 작업했던 특정 함수의 주소 값이나 메모리 상태, 레지스터 상태 등을 angr 모듈을 통해 분석하고 공격을 수행한다.

4.2 Call to shellcode

Call to shellcode는 기존 Exploit 기법 중 RTS 기법에 기반한 자동화 Exploit 기법이다.

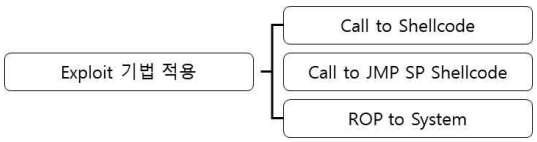


그림 3 Exploit mechanism of rex

명령 포인터를 조작하여 쉘코드를 실행시키는 공격 기법이기 때문에 IP Overwrite가 가능한지 판별하고 쉘 코드를 쓰기 가능한 메모리 영역에 저장한 뒤 쉘 코드에 대한 값과 해당 주소에 대한 명령 포인터를 제한할 수 있는지 확인한다. 이후 스택으로 바로 이동할 수 있는지 확인하고 메모리 보호 기법인 ASLR이 비활성화되었는지 확인한다. 표준 입력(stdin)이 접근할 수 있는 스택의 모든 영역과 구체적인 제약이 없는 가장 큰 메모리 영역을 찾는다. NOP sled로 점프할 수 있는지 확인하고 위 과정에서 조사한 쓰기 가능한 주소 중 하나에 쉘 코드를 저장한다. 반환 주소를 쉘 코드의 주소로 변조하여 메모리에 저장된 쉘 코드를 실행시킨다.

4.3 Call to JMP SP shellcode

Call to JMP SP shellcode는 RTS Exploit 기법을 기반으로 쉘 코드의 종류 중 JMP SP shellcode를 사용하는 exploit 기법이다. 명령 포인터를 조작하는 IP Overwrite가 가능한 지 판별하고 jmp sp′를 글로벌 메모리에 저장한다. jmp sp′가 메모리에 있어야 하는 제약 조건을 적용하고 jmp sp′ stub에서 IP가 가리켜야 하는 구속 조건과 쉘 코드가 sp에 존재해야 하는 제약 조건을 추가한다. Call to shellcode 기법과 동일하게 반환 주소를 쉘 코드의 주소로 변조하여 메모리에 저장된 쉘 코드를 실행시킨다.

4.4 ROP to System

ROP to system은 ROP 공격 기법에 기반하여 system() 함수의 주소 값을 구한다. 이후 "/bin/sh"를 쓰기 가능한 공간에 저장하여 system("/bin/sh\x00")을 실행하도록 프로그램 흐름을 조작한다. 명령 포인터를 조작하는 IP Overwrite가 가능한 지 판별하고 system() 함수의 주소 값을 탐색한다. 문자열 "/bin/sh\x00"를 메모리에 저장하고 ROP 연속 호출 기법을 구성한다. 바이너리에 /bin/sh가 존재해야 한다는 제약 조건을 적용하고 ROP 연속 호출 기법을 바이너리에 삽입한다. 해당 주소에 ROP 연속 호출 기법이 존재해야 한다는 제약 조건을 추가한다. 반환 주소를 첫 번째 가젯의 주소로 변조하여 공격자가 의도적으로 프로그래밍한 ROP 연속 호출을 실행한다.

V. 실험 결과

실험 환경 사양은 다음과 같다.

○ OS: Ubuntu 14.04 LTS

○ CPU: Intel Core i5-7200U 2.50GHz

○ RAM: 8GB

KISA 정보보호 데이터 챌린지에서 제공된 취약점이 존재하는 ELF 바이너리 15개를 가지고 실험하였고 그 중 2개의 바이너리가 Exploit에 성공하였다. [그림 4]는 rex의 Call to shellocde 기법을 적용하여 생성한 Exploit code이다. Exploit 도구인 pwntools기반으로 작성되었다.

```
import sys
import time
from pwn import *


r = process("./q11")
r.send('\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\xd6\x82
\x04\x08\x00\x83\x04\x08\xea\x82\x04\x08
\x00\x00\x00\x00)\xa0\x04\x08\x16\x00\x0
0\x00)\xa0\x04\x08\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00')
r.interactive()
```

[그림 4]rex의 Exploit Code

해당 프로그램 실행 시 [그림 5]와 같이 루트 권한의 쉘코드를 실행시킨다.

```
(rex) root@ubuntu:~/kisa# python rop_to_
system.py
[+] Starting local process './q02': pid
26714
[*] Switching to interactive mode
$ whoami
root
```

[그림 5] root 쉘을 얻은 모습

VI. 결론

본 논문에서는 자동화 exploit 생성 모듈인 rex를를 분석하여 각 exploit 생성 기법의 특징과 구현 방법에 대해 분석하였다. rex는 symbolic execution 기법을 사용하여 바이너리를 분석하는 angr 모듈을 통해 기존 수동 exploit 과정에서 필요한 공격용 정보를 수집하였으며, 여러 exploit 기법을 각각 대입하여 exploit code를 생성했다. 하지만 UNIX 시스템의 구조를 가진 바이너리를 대상으로 한 실험에서는 낮은 성공률을 보였으며, 자동화 exploit 생성 기능의 개선이 필요한 것으로 보였다. 오픈 소스로 공개된 rex는 현재까지도 꾸준히 개선되고 있다.

[참고문헌]

[1] rex, https://github.com/shellphish/rex/blob/master/README.md

[2] Y.Shoshitaishvili, R.Wang, C.Salls, N.Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel and G. Vigna, (State of) The Art of War: Offensive Techniques in Binary Analysis, IEEE Symposium on Security and Privacy. 2016.

[3] pwntools, https://paper.seebug.org/papers/Archive/refs/pwntools.pdf

[4] N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, Driller: Augmenting fuzzing through selective symbolic execution, Network and Distributed System Security Symposium, 2016.

[5] American Fuzzy Lop, http://lcamtuf.coredump.cx/afl/

[6] P. Hui, Y.Shoshitaishvili and M. Payer, T-Fuzz: fuzzing by program transformation, IEEE Symposium on Security and Privacy. 2018.

프로세스 가상 메모리 데이터 유사성을 이용한 메모리 주입 판단 방법

임수민* 임을규

*한양대학교 컴퓨터소프트웨어학과

한양대학교 컴퓨터소프트웨어학부

A Method for Determining Memory Injection Using Process Virtual Memory Data Similarity

Su-Min Lim*

Eul-Gyu Im

*Division of Computer and Software, Hanyang University

Department of Computer and Software, Hanyang University

요 약

파일리스 악성코드는 악의적인 실행 파일을 파일 시스템에 배치하지 않고 작동시키는 악성코드이다. 파일리스 악성코드는 악성 행위를 수행할 페이로드의 흔적을 은닉하기 위해 메모리 주입 방법을 이용한다. 고도화된 실행 매커니즘을 통해 자식 프로세스 가상 메모리 공간에 악성 실행 코드를 주입하기도 하고, 기존에 실행되고 있던 합법적인 프로세스 가상 메모리 공간에 주입하기도 한다.

본 논문은 메모리 주입 공격이 발생했을 경우, 악성 행위 실제 수행 여부와 상관없이 메모리 주입 여부를 검출할 수 있는 방법을 제안한다. 메모리 주입이 의심되는 프로세스와 동일한 실행 조건을 갖는 복제 프로세스를 실행시키고, 각 프로세스 가상 메모리 영역에 속해있는 데이터 집합을 퍼지 해시를 이용해 비교한 다음 유사도를 산출한다.

I. 서론

파일리스 악성코드는 악의적인 실행 파일을 파일 시스템에 배치하지 않고 작동시키는 악성코드이다. 악성코드 실행 프로세스에서 특정 단계가 기존의 실행 방법과 다르게 디스크 내에 쓰여지지 않은 상태로 실행되는 경우를 의미한다. 원격 실행을 통해 감염되기도 하고 문서 파일에서 명령 프롬프트 프로세스를 실행시켜 악성 쉘코드를 종속적으로 실행시킬 수 있다. 기존 악성코드의 실행 방법 대비 파일리스의 실행 매커니즘은 고도화되었다고 볼 수 있으며, 이로 인해 파일디스크 및 시그니처 기반의 악성코드 탐지 기술을 통해 파일리스 악성코드를 탐지하는 것은 굉장히 어렵다 [1]. 실제로 Ponemon에 따르면, 2017 년에 파일리스 공격이 전년도 대비 45 % 증가했으며 성공적인 공격의 경우 77 %나 증가한 것으로 나타났다 [2]. 파일리스 형태의 공격이 수와 복잡성이 시간이 지날수록 점차 증가할 것으로 예상되므로 파일리스 악성코드 차단 방법 모색이 시급하다 [3].

파일리스 악성코드는 악성 행위를 수행할 페이로드의 흔적을 은닉하기 위해 메모리 주입 방법을 이용한다 [4]. 고도화된 실행 매커니즘을 통해 자식 프로세스 가상 메모리 공간에 악성 실행 코드를 주입하기도 하고, 기존에 실행되고

있던 합법적인 프로세스 가상 메모리 공간에 악성 실행 코드를 주입하기도 한다. 악성 행위 패턴 등을 이용한 동적 분석을 통해 주입된 악성 페이로드의 존재를 감지할 수도 있지만, 악성코드는 점점 다양하고 복잡한 분석 우회 및 은닉 기술을 사용하기 때문에 악성 행위 기반의 메모리 주입 공격 검출은 불확실하다 [5 , 6].

따라서 본 논문은 메모리 주입 공격이 발생했을 경우, 악성 행위 실제 수행 여부와 상관없이 메모리 주입 여부를 검출할 수 있는 방법을 제안한다. 메모리 주입이 의심되는 프로세스와 동일한 실행 조건을 갖는 복제 프로세스를 실행시키고, 각 프로세스 가상 메모리 영역에 속해있는 데이터 집합을 퍼지 해시를 이용해 비교한 다음 유사도를 산출한다.

II. 관련 연구

박희환 외 1인은 Windows 시스템 파일에 기생하는 악성코드의 치료 방법을 연구하였다 [7]. 사용자의 로그인/로그오프를 담당하는 프로세스인 winlogon.exe 등과 같은 윈도우 운영체제의 관리자 권한을 가진 서비스 프로세스에 메모리 주입 활동이 일어났을 때, 감염된 프로세스를 강제 종료하면 컴퓨터 시스템에 치명적인 블루스크린 등의 결과가 발생할 수 있다. 따라서 윈도우 운영체제의 시스템 파일을 종료하지 않고 주입된 dll 파일만 제어하기 위해, 종료 함수를 삽입한 스레드를 제어하여 악성 dll의 동작을 종료할 수 있다. 그런데 위 치료 방법은 dll 주입 공격을 통한 메모리 주입의 경우에만 치료가 가능하다. 실제 악성코드가 이용하는 메모리 주입은 프로세스 할로윙, 쉘코드 주입 등 다양한 방법을 통해 이루어진다 [4].

박창욱 외 3인은 퍼지해시를 통해 유사 악성코드 분류 모델에 대해 연구하였다 [8]. PE 섹션 중 .data 영역을 비교인자로 채택해, 악성코드 간의 유사도를 측정하는 방법이다. 위 방법은 PE 구조가 패킹되지 않아야하며, .data 섹션이 필수적으로 존재하고 무결성을 유지해야한다는 조건이 필요하다. 그런데 최근 악성코드의 동향을 살펴보면, 악성코드 탐지 및 분석을 회피하기 위해 패킹된 바이너리 형태로 유포되는 경우가 많다 [6]. 또한 PE(Portable Executable) 포맷 형태의 실행파일이 아닌 문서형 악성코드나 원격 실행을 통한 파일리스 형태의 악성코드의 수가 급증하고 있으므로, 해당 방법을 통한 분류는 실제 악성코드 분류에 적합하지 않다 [3].

메모리 주입 공격이 일어났을 경우, 피해 프로세스의 가상 메모리 공간에 주입된 실행 코드와 같은 흔적이 존재한다는 사실에 기반해, 가상 메모리 데이터 비교를 통한 메모리 주입 공격 여부를 판단할 수 있다.

III. Windows API

윈도우 운영체제 보안정책상 특정 프로세스가 타 프로세스 메모리 영역의 데이터에 접근하는 것은 불가능하다. 하지만 윈도우는 디버깅의 이유로 ReadProcessMemory 라는 Windows API(Application Program Interface) 를 제공한다. ReadProcessMemory 함수를 이용하기 위해 ReadProcessMemory 함수가 필요로하는 인자를 구해야한다. 아래 그림 1 은ReadProcess Memory 함수의 구조를 나타낸 것이다.

```
BOOL WINAPI ReadProcessMemory(
  _In_   HANDLE hProcess,
  _In_   LPCVOID lpBaseAddress,
  _Out_  LPVOID lpBuffer,
  _In_   SIZE_T nSize,
  _Out_  SIZE_T *lpNumberOfBytesRead
);
```

그림 1 ReadProcessMemory 함수

ReadProcessMemory 함수는 다섯 개의 인자를 필요로 하는데, 다섯 번째 인자는 일반적으로 사용하지 않으며 NULL 값을 넣어 무시할 수 있다. ReadProcessMemory 가 필요로 하는 나머지 네 개의 인자는 순서대로 아래와 같다.

(가) 대상 프로세스의 Open Handle

(나) 대상 프로세스에서 읽을 값의 시작 메모리 주소

(다) 메모리에서 읽은 값을 저장할 포인터

(라) 메모리에서 읽을 데이터의 바이트 수

```
HANDLE WINAPI OpenProcess(
  _In_   DWORD dwDesiredAccess,
  _In_   BOOL bInheritHandle,
  _In_   DWORD dwProcessId
);
```

그림 2 OpenProcess 함수

프로세스 핸들을 구하기 위한 OpenProcess 함수의 구조는 위 그림 2 와 같다. 또한 프로세스 메모리에서 유효한 값을 스캔하기 위해, VirtualQueryEX 함수를 이용해 읽을 주소의 페이지 정보를 얻어온다. VirtualQueryEX 함수는 프로셋의 핸들, 값을 가져올 메모리의 주소, MEMORY_BASIC_INFORMATION 구조체, MEMORY_BASIC_INFORMATION의 길이를 인자로 갖는다. 아래 그림 3은 VirtualQueryEX 함수의 구조를 나타낸 것이다.

```
SIZE_T WINAPI VirtualQueryEx(
  _In_      HANDLE hProcess,
  _In_opt_  LPCVOID lpAddress,
  _Out_     PMEMORY_BASIC_INFORMATION lpBuffer,
  _In_      SIZE_T dwLength
);
```

그림 3 VirtualQueryEX 함수

MEMORY_BASIC_INFORMATION 는 프로세스 가상 주소 공간에 있는 페이지 범위에 대한 정보를 가지는 구조체이다.MEMORY_BASIC_INFORMATION 구조체가 필요로 하는 멤버는 순서대로 아래와 같다.

(가) 페이지 영역의 기본 주소에 대한 포인터

(나) VirtualAlloc 함수에 의해 할당 된 페이지 범위의 기본 주소에 대한 포인터

(다) 영역이 처음 할당되었을 때 메모리 보호 옵션

(라) 기본 주소에서 시작하는 영역의 크기(바이트)

(마) 영역의 페이지 상태

(바) 해당 지역의 페이지에 대한 액세스 보호값

(사) 영역의 페이지 유형

IV. 프로세스 가상 메모리 비교

본 논문에서 제안하는 실시간 메모리 주입 공격 탐지 방법은 아래와 같다.

(가) 현재 실행 중인 프로세스 중 악성 의심이 드는 프로세스의 이름을 찾는다.

(나) (가)의 프로세스 이름과 동일한 이름의 프로세스들의 실행 경로와 실행 매개변수, 해당 프로세스의 pid 를 출력한다.

(다) (가)에서 입력한 이름을 갖는 프로세스 목록 중 비교에 사용할 pid 를 선정한다. 그 다음 개발한 메모리 비교 프로그램에 해당 pid 를 매개변수로 삽입한다.

(라) (다)의 pid 와 동일한 경로, 동일한 조건을 갖는 프로세스를 본 개발물 프로세스의 서브 프로세스로 생성한다.

(마) (라)에서 새로 생성한 프로세스와 (다)에서 선정한 프로세스의 가상 메모리 공간에서 읽을 수있는 영역의 데이터를 스캔해 퍼지해시 알고리즘을 이용해 비교한다. 프로세스 가상 메모리 데이터 영역의 접근은 3 장의 Windows API를 이용한다.

4.1 퍼지해시(Fuzzy Hash)

해싱(Hashing)은 모든 입력을 고정 크기 출력으로 축소시키는 것이다. 두 데이터의 해시결과가 같으면 같은 데이터라고 판단할 수 있다. 그런데 전체 데이터를 하나의 입력 값으로 해시 알고리즘을 거칠 경우, 바이트 정도의 굉장히 작은 데이터가 일부만 바뀌어도 전체 값이 완전히 바뀔 수 있다. 따라서 퍼지해시를 이용하면 기존 해시의 무결성을 확보하면서, 비교 대상과의 유사도를 구할 수 있다.

퍼지해시 알고리즘은 입력 데이터 전체에 대한 해시값을 계산하지 않고, 일정 크기 단위로 구분하여 각 단위 블록에 대한 해시값을 생성하는 방식이다. 메모리 주입 공격이 발생할 경우 동일한 조건의 다른 프로세스와 완전히 다른 가상메모리 영역을 포함하고 있을 것이다.

4.2 비교 결과

아래 그림 4 와 그림 5는 동일한 프로세스 이름을 가진 두 프로세스를 비교한 결과 값이다. 그림 4 의 경우 pid 3420, pid 3616 두 프로세스를 비교한 것이며 퍼지해시 유사도는 0으로 측정되었다. 그림 5 의 경우 pid 3420, pid 2848 두 프로세스를 비교한 것이며 퍼지해시 유사도는 83의 값이 측정되었다.

```
C:₩Users₩Administrator>cmd2.py -p compare 3420,3616
(3420, 873, 66)
12:neRrhM5HehBGFZgcM2ICGE7UvWNkM2NSJaE2nDB4gDzyFWayKb151XG:eZhMehBWg4dUvOkM2NSI1
9TDzZAhXG
(3616, 800, 68)
24:t3LmMeehBLzK2+vW9U12NSZ9TDzZuXY4Ub3:t3LmMRBLz5+vWe4SZtsnT
0
```

그림 4 두 프로세스 비교 결과 1

```
C:₩Users₩Administrator>cmd2.py -p compare 3420,2848
(3420, 873, 66)
12:neRrhM5HehBGFZgcM2ICGE7UvWNkM2NSJaE2nDB4gDzyFWayKb151XG:eZhMehBWg4dUvOkM2NSI1
9TDzZAhXG
(2848, 907, 72)
12:783KSLHehBGFZgcM2ICGE7UvWNkM2NSJaE2nDB4gDzyFWayKt91XTmx:g3ZehBWg4dUvOkM2NSI19
TDzZuXyx
83
```

그림 5 두 프로세스 비교 결과 2

V.결론

본 논문은 임의의 프로세스를 대상으로 동일한 실행 조건을 가진 복제 프로세스를 생성하고, 두 프로세스 가상 메모리 데이터 비교를 통한 메모리 주입 공격 여부를 판단하는 방법을 제안하였다. 실행파일이 메모리에 로드되어 실행되면 흔적 정보가 있기 때문에, 메모리 주입 공격 탐지의 경우 메모리 포렌식 도구를 이용한 메모리 분석 방법이 많이 활용된다. 그러나 실제 사용자 환경에서의 메모리 덤프 시간이 굉장히 오래 걸리기 때문에 즉각적인 감염 대처가 불가능하다는 치명적인 단점이 존재한다. 따라서 메모리 주입 공격이 발생되어져 있는 시스템 환경에서 프로세스 가상 메모리 데이터 비교로 실시간으로 감염여부를 판단한다면, 메모리 포렌식 방식의 단점을 극복할 수 있다.

VI. Acknowledgements

본 연구는 2017년도 산업통상자원부의 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다(No. 20174010201170).

[참고문헌]

[1] A. Magnusardottir, "Fileless ransomware. How it works and how to stop it [Online]",infosecurity, Available: https://www.infosecurityeurope.com/__novadocuments/483997?v=636650015234830000(downloaded 2018, OCT.28)

[2] M. Gorelik, R. Moshailov, et al, "Fileless Malware: Attack Trend Exposed", Morphisec Ltd, 2017

[3] J. Smelcer, "The rise of Fileless malware", Utica College, ProQuest Dissertations Publisjing, 10642524, 2017

[4] P. Black, I. Gondal, R.Layton, "A survey of similarities in banking malware behaviours", Computers & Security, vol 77, pp. 756~772, 2018

[5] T. Tomer, "Detecting the one percent: Advanced targeted malware detection", Proceedings of the RSA Conference, San Francisco, USA, Vol. 8, 2013

[6] J. A. Marpaung, M. Sain, Lee. H. J, "Survey on malware evasion techniques: State of the art and challenges", In Advanced Communication Technology (ICACT), 2012 14th International Conference, pp. 744-749, 2012

[7] Park. H. H, Park. D. W, "A Study on Treatment Way of a Malicious Code to injected in Windows System File," KSCI Review, Vol. 14, No. 2, pp. 255~262, 2006

[8] Park. C. W, Chung. H. J, Seo. K. S, Lee. S. J, "Research on the Classification Model of Similarity Malware using Fuzzy Hash," Journal of the Korea Institute of Information Security & Cryptology, Vol. 22, No. 6, pp. 1325~1336, 2012

Data Space Randomization을 활용한 MVEE의 Variant 생성 기법 연구

황동일*, 신장섭*, 전성일*, 백윤흥*
*서울대학교 전기·정보공학부, 반도체공동연구소

Variant Generation for MVEE Using Data Space Randomization

Dongil Hwang*, Jangseop Shin*, Seongil Jeon* and Yunheung Paek*
*ECE and ISRC, Seoul National University

요 약

C와 C++는 컴파일된 결과물의 실행 속도가 타 프로그래밍 언어로 작성된 것보다 빠르고 기존의 시스템과 호환성이 좋다는 점 등의 장점을 갖고 있기 때문에 많은 프로그래머들로부터 여전히 사랑받는 프로그래밍 언어이다. 하지만 C와 C++로 작성된 프로그램은 버퍼 오버플로 취약점과 같은 다양한 메모리 커럽션 취약점을 갖고 있기 때문에 쉽게 외부의 공격자들에게 노출된다. 본 연구에서는 이러한 메모리 커럽션을 기반으로 한 공격들을 막기 위해서 고안된 Multi-variant Execution Environment (MVEE)와 Data Space Randomization (DSR) 을 설명하고 이들을 활용하여 다양한 공격을 막는 방법을 제안하고자 한다.

I. 서론

프로그래밍 언어 관련 시장 조사 업체인 TIOBE에서 조사한 바에 따르면 2018년 11월에 사용된 프로그래밍 언어 중 C와 C++는 14.396%, 8.282%의 점유율을 기록하여 Java에 이어 각각 2위와 3위를 기록한 것으로 나타났다 [1].

보다 직관적으로 쉽고 편하게 사용할 수 있는 다양한 프로그래밍 언어가 개발된 오늘날에도 이처럼 C와 C++가 널리 사용되고 있는 것은 기존 시스템과의 호환성이 좋고 컴파일된 결과물의 실행 속도가 빠르다는 장점이 있기 때문이다. 하지만 C나 C++을 사용하여 프로그래밍 된 프로그램에는 악용될 여지가 있는 메모리 커럽션 취약점이 존재한다.

예를 들면 한 프로그램이 기존에 선언된 어레이에 접근하고자 할 때, 미리 할당된 메모리 범위 내의 인덱스에 올바르게 접근하는지를 매번 확인하는 Java와는 달리 C와 C++는 이러한 확인 과정을 거치지 않는다. 이로 인해 C나 C++로 구현된 프로그램은 공격자가 시스템의 메모리에 원하는 값을 쓰는 것을 가능토록 하는 버퍼 오버플로 취약점을 갖게 된다.

<표 1> 2018년 11월의 TIOBE Index[1]

Rank	Programming Launguage	Ratings
1	Java	16.746%
2	C	14.396%
3	C++	8.282%
4	Python	7.683%
5	Visual Basic .NET	6.490%
6	C#	3.952%
7	JavaScript	2.655%
8	PHP	2.376%
9	SQL	1.844%
10	Go	1.495%

위와 같은 메모리 커럽션 취약점의 존재는 이를 이용한 공격 기법들의 등장으로 이어졌다. 예를 들면, 취약점을 사용하여 실행하고자 하는 악성 코드를 공격 대상이 될 시스템의 스택에 삽입한 뒤, 리턴 주소나 점프 주소 등을 조작하여 프로그램의 제어 흐름 (control flow)을 삽입한 악성 코드의 위치로 이동하여 이를 실행시키는 공격이 가능하다. 이와 같은 공격 방식을 "코드 삽입 공격"이라고 한다.

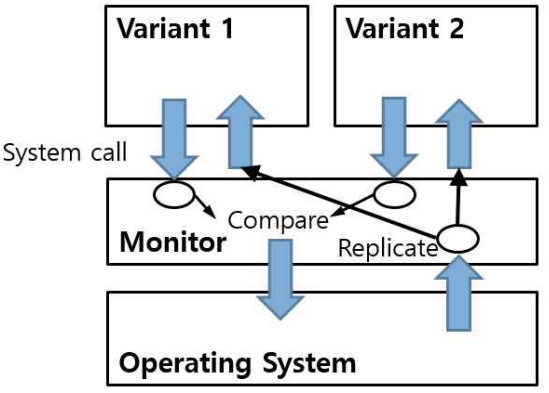
하지만 코드 삽입 공격은 W^X protection이라고 하는 간단한 방어 기술로 무력화 시킬 수 있다. 메모리 영역을 "쓰기 가능 (writable)" 혹은 "실행 가능 (executable)" 중 한 가지 권한만 가질 수 있도록 설정함으로써 스택과 같은 데이터 영역에 공격자가 새로 삽입한 악성 코드의 실행을 차단하는 것이다.

W⊕X protection과 같은 방어 기술을 우회하기 위해 새롭게 개발된 공격 방식이 코드 재사용 공격 (Code Reuse Attack, CRA)으로 오늘날에도 수많은 시스템에게 위협이 되고 있다. CRA는 공격자가 실행하고자 하는 악성 코드를 스택에 삽입하는 것이 아니라 실행 가능 영역의 코드의 조각 (gadget)들을 모아서 공격자가 원하는 행동을 하도록 만드는 공격 방식이다. CRA는 코드 삽입 공격과는 방식이 다르지만 메모리 커럽션 취약점을 이용한 스택 내의 리턴주소나 점프주소의 변경이 선행된다는 점에서 궤를 같이 한다.

이처럼 나날이 발전하고 있는 메모리 커럽션 취약점 기반의 공격들을 막기 위해서 많은 연구자들이 새로운 방어 기술의 개발에 매진하고 있다. 본 논문에서는 이러한 방어 기법 중에서도 Multi-Variant Execution Environment (MVEE)와 Data Space Randomization (DSR)을 활용하여, 메모리 취약점 기반의 공격에 보다 확실하게 대비하는 방법을 제안하고자 한다.

II. Multi-variant Execution Environment

Multi-variant execution environment (MVEE)의 개념은 2006년 Cox에 의하여 최초로 고안되었으며 [2], 지금도 관련 연구가 활발히 이루어지고 있다 [6][7]. 이들은 공통적으로 그림 1과 같이 하나의 프로그램을 조금씩 다른 형태를 갖는 여러 개의 variant로 복제하여 동시에 실행시키고 이들의 행동을 비교함으로써 공격 시도가 있었는지의 여부를 판단한다.



(그림 1) Multi-variant Execution Environment

Variant간의 행동 비교는 대게 시스템 콜이 발생할 때 마다 이루어지는데, 그 이유는 공격자들이 의도대로 시스템에 영향을 주기 위해서는 시스템 콜을 거쳐야만 하기 때문이다. 따라서 모니터는 시스템 콜이 발생할 때마다 각각의 variant가 사용한 시스템 콜 넘버와 아규먼트 값을 비교하고 불일치함이 발견되면 공격이 발생한 것으로 간주한다.

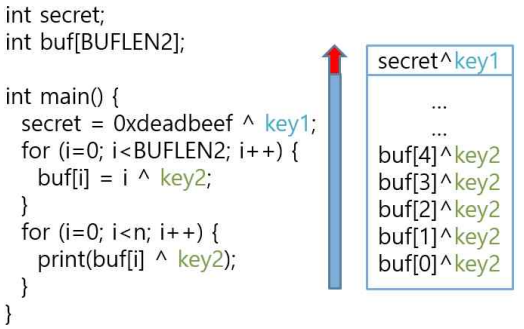
Variant들은 정상적인 프로그램 수행 상태일 때는 서로 같은 행동을 보이지만 공격 시도가 있었을 시에는 서로 불일치한 행동을 보여야 하는데, 이를 위해 기존의 MVEE 연구는 주로 address space layout randomization (ASLR)을 기반으로 variant들을 생성하였다. 각 variant에 ASLR을 적용하여 유효한 주소 공간을 서로 겹치지 않도록 하는 것이다.

결과적으로 공격자가 메모리 커럽션 취약점을 사용하여 하나의 variant의 제어 흐름이 자신이 의도한 절대 주소로 점프하도록 하더라도 이는 다른 variant에서는 유효하지 않은 주소가 될 것이므로 행동 불일치가 발생하게 되고 모니터가 탐지할 수 있으며, 두 variant의 주소 공간 불일치로 인하여 코드 재사용 공격 역시 매우 어려워지게 된다.

하지만 ASLR을 적용하여 variant를 생성했을 때에는, 특정 data만을 조작하여 시스템 콜의 아규먼트는 바뀌지 않는 유형의 공격에 대한 대비가 불가능하다. 예를 들어 payload의 크기를 실제보다 크게 설정하여 서버에 요청하는 Heartbleed 공격 [3]의 경우 ALSR 기반의 MVEE를 적용했을 때 시스템 콜의 넘버나 아규먼트가 바뀌지 않기 때문에 탐지가 불가능하다.

III. Data Space Randomization

Data Space Randomization (DSR)은 버퍼 오버플로와 같은 취약점을 사용해, 공격자가 메모리에 원하는 데이터를 씀으로써 이루어지는 공격을 방어하기 위한 기법이다 [4]. DSR을 적용하면 적법한 데이터 흐름을 발생시키는 명령어들은 그대로 동작하지만, 그렇지 않은 데이터 흐름은 공격자가 의도한 대로 이루어지지 않게 된다. 이를 위해서 그림 2에서처럼 프로그램의 메모리에 저장되는 모든 변수를 특정 키 값을 사용하여 XOR 연산을 통해 암호화시킨다.



(그림 2) Data space randomization

그림 2에서 secret 값은 메모리에 저장될 때 key1을 사용하여 암호화된 상태로 저장되었기 때문에, 다시 불러올 때에도 같은 key1를 통해 복호화를 해야만 원래의 값을 얻을 수 있다. 버퍼 오버리드와 같은 취약점을 사용하여 적법하게 secret을 불러오는 코드가 아니라 다른 코드를 사용해 secret 값을 읽어올 경우, 공격자가 볼 수 있게 되는 값은 secret^key1^key2가 되어 본래의 secret 값을 알 수 없게 된다. 반대로 공격자가 자신의 payload를 버퍼 오버플로 취약점을 사용하여 삽입하는 경우에도 DSR에 의하여 유효하지 않은 키와 XOR 된 값이 메모리에 삽입되고, 이 paylaod를 불러올 때는 적법한 원래의 키를 사용하기 때문에 결과적으로 공격자가 의도한 payload가 로드되지 않아 공격을 실패하게 된다.

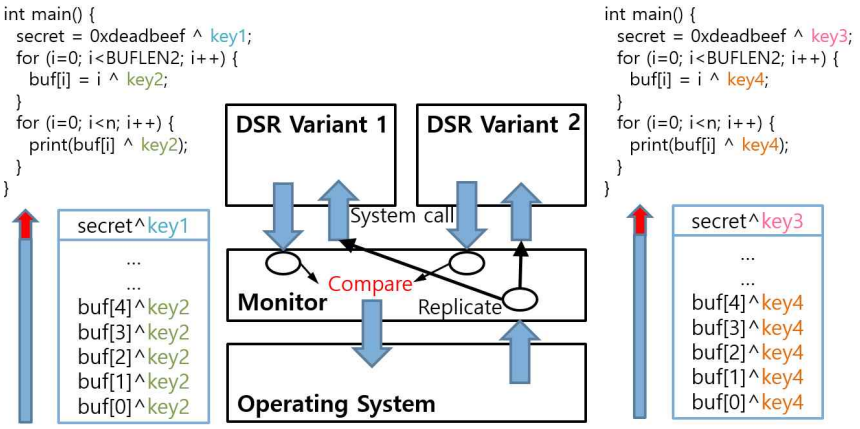
하지만 DSR을 프로그램에 적용하기 위해선 모든 strore와 load가 발생할 때 마다 key를 사용하여 XOR 연산을 하는 추가 명령어가 수행되어야만 하기 때문에 성능이 느려진다는 단점이 존재한다. 이러한 성능 문제를 해결하기 위해서 하드웨어의 수정을 통해 프로세서의 파이프라인에 키 연산을 도와주는 모듈을 추가하고, 이를 처리하는 명령어를 추가하여 DSR의 성능을 개선시키려는 연구가 진행된 바 있다 [5].

성능 부하보다 더 큰 문제는 공격자가 평문을 알고 있다면, Heartbleed 같은 정보 유출 공격을 통해 암호문을 얻은 뒤 키 값을 너무나도 쉽게 유추할 수 있다는 것이다. 이러한 공격을 known plaintext attack이라고 한다. 공격자가 이를 통해 키를 알아낸다면, DSR이 적용한 시스템에서도 유효한 payload를 생성하여 심화된 공격을 할 수 있게 된다.

IV. DSR을 활용한 MVEE의 Variant 생성

본 장에서는 MVEE와 DSR의 각각의 단점을 상쇄하고 장점을 살리기 위하여 MVEE의 variant를 생성할 때 각각 서로 다른 키를 사용하는 DSR을 적용하는 방안을 제안한다.

위와 같은 방식으로 MVEE의 variant를 생성할 경우 그림 3과 같이 각 variant들은 서로 다른 키들을 사용하게 된다. 그리고 buf를 불러오는 코드를 사용하여 secret 값에 접근할 경우 로드되는 값이 각각 secret^key1^key2와 secret^key3^key4로 일치하지 않으므로 시스템 콜이 수행될 때 사용하는 아규먼트에 불일치가 생기고, MVEE 모니터에서는 이를 탐지하여 공격이 일어난 것으로 간주하게 된다.



(그림 3) DSR을 활용한 MVEE의 variant 생성

이 방어 기법을 사용한다면 ASLR 기반의 MVEE로는 방어하는 것이 힘들었던, 특정 데이터의 조작만으로 이루어지는 Heartbleed와 같은 공격에 대응할 수 있다. 그리고 정보 유출이 MVEE 모니터를 사용하여 탐지되기 때문에 known plaintext attack 또한 성립이 불가능해진다.

물론 이를 적용하는 데에 가장 큰 걸림돌은 성능 문제일 것이다. 하드웨어의 도움을 받아 DSR의 성능이 개선되었다고는 하나 여전히 무시할 수 없는 성능 오버헤드가 존재하고, MVEE 역시 시스템 콜 간의 비교가 이루어지는 과정에서 큰 성능 오버헤드가 발생하기 때문이다. 따라서 본 방어 기법의 적용을 위해서는 전용 하드웨어를 설계하는 등 최적화된 방어 동작을 할 수 있도록 해야 할 것이다.

V. 결론

메모리 커럽션 취약점을 사용한 공격을 막기 위한 기술인 multi-variant execution environment (MVEE)와 data space randomization (DSR)은 각각 장단점을 가지고 있다. 본 논문에서는 두 기법을 종합하여 보다 넓은 범위의 공격을 막을 수 있는 새로운 방어 기법을 제안하였다.

[Acknowlegment]

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00213, 능동적 사전보안을 위한 사이버 자가변이 기술 개발)

[참고문헌]

[1] http://www.tiobe.com/tiobe-index/

[2] B. Cox et al., "N-variant systems: A secretless framework for security through diversity," in USENIX Security, 2006.

[3] N. Mehta, The Heartbleed Bug. http://heartbleed.com/, 2014

[4] S. Bhatkar et al., "Data space randomization," in DIMVA, 2008.

[5] Brian B. Belleville et al., "Hardware Assisted Randomization of Data," in RAID, 2018

[6] K. Koning et al., "Secure and efficient multi-variant execution using hardware-assisted process virtualization," in Conference on Dependable Systems and Networks (DSN), 2016.

[7] S. Volckaert et al., "Secure and efficient application monitoring and replication," in USENIX Annual Technical Conference (ATC), 2016.

SSL 암호화 전 패킷 추출 연구 방안

김영수†*, 원혜린**, 정재우*, 류민수*, 김현지***, 김의연**, 김경곤‡****
*세종대학교 정보보호학과, **서울여자대학교 정보보호학과, ***인하대학교 정보통신공학과,
****고려대학교 정보보호융합학과 산업협력중점 교수

Study on packet extraction before encryption with SSL

Young-Su Kim†*, Hye-Rin Won**, Jae-Woo Jeong*, Min-Su Ryu*,
Hyeon-Ji Kim***, Eui-Yeon Kim**, Kyoung-Gon Kim‡****
*Department of computer and information security, Sejong University.
**Department of Information Security, Seoul Women's University.
***Department of Information and Communication Engineering, Inha University.
****Division of information security, Korea University.

요 약

현재 많은 애플리케이션에서는 네트워크의 전송 계층 상에서 클라이언트와 서버에 대해 인증을 진행하고, 데이터를 암호화하는 SSL을 사용하고 있다. 본 고에서는 SSL을 활용하는 애플리케이션 중 하나의 예시로 인스턴스 메신저를 기준으로 SSL 암호화 전에 패킷 추출에 대한 연구를 진행하였다. 이를 통해 취약점 분석을 진행할 때 하나의 방법론으로 적용할 수 있다.

I. 서론

현재 SSL(Secure Socket Layer)는 인증, 신뢰성, 보안성 세 가지의 측면에서 장점을 가지고 있기 때문에 많은 애플리케이션 단위에서 활용하고 있다. 인증을 활용하여 모든 메시지의 오리진을 보장하고 있고, 메시지 무결성 점검인 MAC을 사용하여 신뢰성을 준다. 또한 비밀 키를 정의한 데이터가 교환된 후에 구성요소 간의 메시지가 암호화가 되기 때문에 보안성에서 또한 강점을 가지고 있다. [1]

이러한 이점으로 많은 애플리케이션이 SSL을 사용하고 있으나, 더 나은 보안성 향상을 위해서는 이러한 애플리케이션들의 취약점 분석을 진행해야 한다. 이를 위해서는 SSL을 분석하는 방법을 이해하고 있어야 한다. 따라서 본 고는 취약점 분석을 진행할 때 하나의 방법론으로 적용할 수 있는 SSL 암호화 전 패킷 추출 방법을 제시하고자 한다. 이를 위해 본 논문에서는 하나의 예로서 메신저 애플리케이션을 활용하였다.

II. SSL 동작 방식

본 연구에 앞서 SSL이란 보안 소켓 계층을 의미한다. 특히 SSL은 OSI 7계층인 응용 계층, 표현 계층, 세션 계층, 전송 계층, 네트워크 계층, 데이터 링크 계층, 물리 계층 중에 응용 계층과 전송 계층 사이에 독립적인 프로토콜을 만들어서 동작이 된다. 여기에서 응용 계층의 프로토콜에서 외부로 데이터를 보낼 경우 SSL으로 전달이 된다. 이후에 SSL에서 받은 데이터를 암호화하여 TCP에 전송을 한다. 이후에 데이터는 외부로 빠져나간다. 또한 반대로 전달받을 경우에는 TCP로부터 받은 데이터를 SSL이 받고, 이 과정에서 복호화하여 응용 계층에 전달을 한다. 이러한 과정들은 애플리케이션이 SSL을 TCP로 인지하고 있고, TCP는 SSL을

* 본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best)의 연구 지원으로 수행되었습니다. †주저자, best7ward@gmail.com ‡교신저자, anesra@gmail.com(Corresponding author)

애플리케이션으로 인지하기 때문에 가능하다.

SSL 프로토콜의 계층 구조를 살펴 보면 바로 다음과 같다.

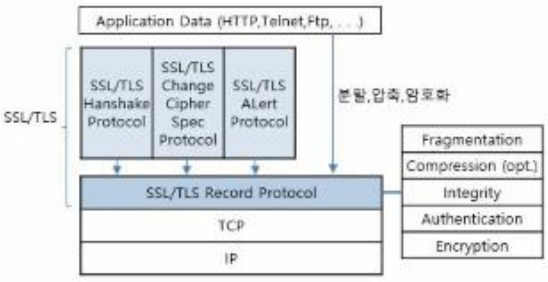


그림 1. SSL/TLS 프로토콜 스택[1]

SSL은 현재 TCP 위에서 Record Protocol을 통해 보안을 제공하고 있고, SSL handshake[2] Protocol이나 SSL Change Cipher Spec Protocol, SSL Alert Protocol을 통해 SSL의 작동을 관리하고 있다.

2.1. Record Protocol

그림 2. Record Protocol구조

보안을 제공하고 있는 Record Protocol은 데이터의 분할(2^{14} 또는 그 이하의 블록으로 데이터를 나누는 것)과 압축을 수행한 후, 안전한 TCP 패킷으로 변환한다. [3] 이 과정을 통해 데이터 암호화와 무결성을 위한 메시지 인증을 수행하는 프로토콜이다. 또한 SSL의 작동을 관리하는 세 프로토콜을 관리하는 역할을 한다. Change Cipher Spec Protocol의 예를 들자면, 데이터를 보내기 좋게 지르거나 붙인다. 그리고 선택적으로 압축을 한 후에 MAC[4]을 적용한다. 여기서의 MAC은 Message Authentication Code라 해서 메시지를 인증하는 데에 사용되는 짧은 정보를 의미한다. 그리고 이를 암호화 한 후에 TCP로 전달을 진행한다.

2.2. Change Cipher Spec Protocol

상호 암호 통신을 수행하기 위해서는 보안 알고리즘 정보를 사전에 협의를 해야 하는데, 이를 위해서 일종의 보안 매개 변수를 상대에게 전달하는 데 사용되는 프로토콜이다. 이 프로토콜의 내용에는 하나의 바이트, 1이라는 값이 항상 들어간다. 또한 프로토콜 필드에는 Change Cipher Spec을 나타내는 20이 들어간다.

표 1. Change Cipher Spec Protocol구조

2.3. Alert Protocol

20	3	0	0	1	1	
Protocol	Version		Length		Message	MAC

Alert Protocol은 Handshake 과정에서 오류가 발생하게 된다면 상대방에 오류를 통보하는 기능을 의미한다. 주로 2바이트로 구성이 되며, 첫 번째 바이트는 warning이, 두 번째 바이트에서는 오류 메시지가 들어가게 된다.

21	3	0	0	2	level	Description
Protocol	Version		Length		Message	

표 2. Alert Protocol구조

III 연구 사항

SSL은 전송 계층의 보안을 위해 만들어졌고, TCP 내 메시지를 암호화를 하여 보낸다. 본 과정에 대해 생각을 해보았을 때 SSL은 이미 만들어진 메시지를 암호화하는 과정이다. SSL 암호화 루틴[5] 내에서는 보내려는 메시지 버퍼가 있고, 그 루틴에 대하여 후킹을 하게 된다면 원래 보내려는 메시지에 대한 원문을 볼 수 있다. 이에 SSL은 다음과 같은 방식으로 암호화를 진행하는 것을 알 수 있었다.

1. 오픈 소스 라이브러리 내 암호화 함수 사용

2. 직접 구현한 암호화 루틴을 사용

1번의 경우 라이브러리에서 제공하는 암호화 함수에 인자로 메시지 버퍼 및 인자가 들어오게 된다. 이 경우에는 함수에 후킹을 건 뒤에 인자 내용을 확인하면 원문에 대하여 얻을 수 있다.

* 본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best)의 연구 지원으로 수행되었습니다. †주저자, best7ward@gmail.com ‡교신저자, anesra@gmail.com(Corresponding author)

그림 3. Libssh.c 내 ssl3_write 후킹 시도

2번의 경우 암호화 루틴[9]을 찾는 다양한 방법이 있다. 모든 루틴에 대한 정적 분석, break point를 통한 동적 분석 등이 존재한다. 하지만 효율적으로 찾을 수 있는 방법은 다음과 같다.

(1) 문자열 기반 탐색

(2) 핸들러 기반 탐색

(3) 부채널 기반 탐색

(1) 문자열 기반 탐색은 효율이 낮지만 범용적으로 사용될 수 있는 방법이다. Encrypt, SSL, openssl 같은 특정 문자열에 대하여 탐색을 시도를 하는 방법이다. 과정은 다음과 같다.

1. 문자열 검색을 시도한다.

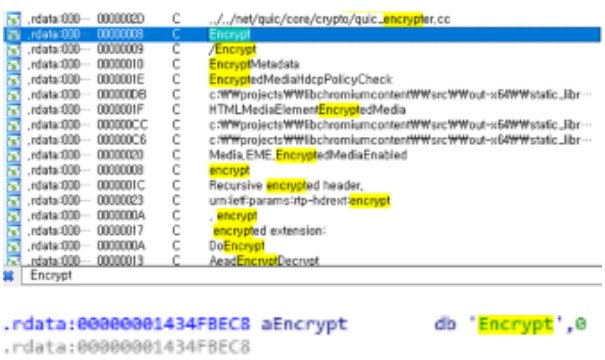


그림 4. Encrypt 문자열 검색

2. 역참조를 할 수 있는 Cross Reference 기능을 사용하여 문자열이 참조되는 곳에 대하여 탐색을 시도한다.

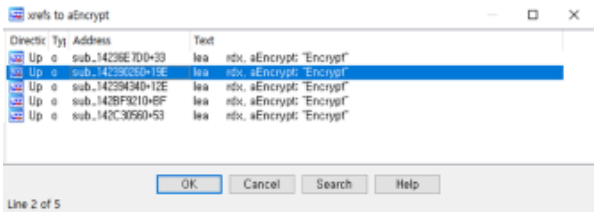


그림 5. 문자열 참조 부분 탐색 시도

3. 후킹을 시도하여 루틴을 찾는다.

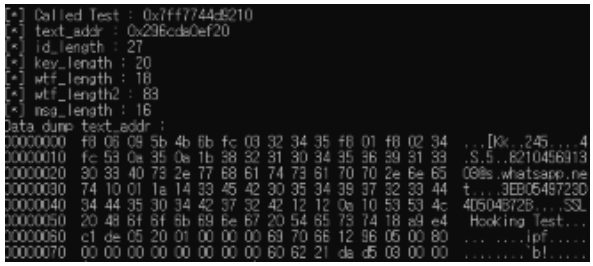


그림 6. 후킹 시도

(2) 핸들러 기반 탐색은 핸들러가 사용된다는 특정 상황에서 사용될 수 있는 방법이다. 먼저 핸들러를 찾는다. 핸들러는 대부분 주요 함수마다 사용되고, 기능에 대한 문자열을 받는다. 핸들러를 찾는 방법에는 정도가 없지만 찾는다면 효율적으로 암호화 루틴을 찾을 수 있다.

1. 동적 분석 및 정적 분석을 통해 핸들러 함수를 찾는다.

```
v9 = sub_1402E7760(
       (__int64)&v14,
       (__int64)"DoEncrypt",
       (__int64)"../../components/webcrypto/webcrypto_impl.cc",
       388,
       v8);
```

그림 7. 핸들러 함수 발견

2. 핸들러에 대하여 후킹을 진행하여 실행되는 제공되는 API 목록을 확인한다.

* 본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best)의 연구 지원으로 수행되었습니다. †주저자, best7ward@gmail.com ‡교신저자, anesra@gmail.com(Corresponding author)

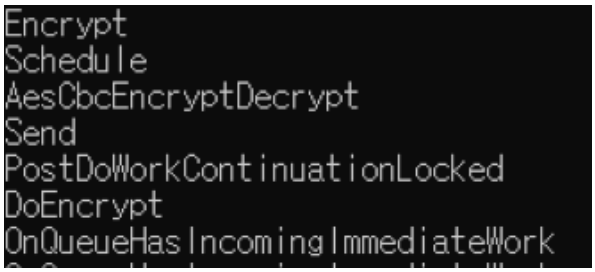


그림 8. 확인된 API 목록의 일부

3. API 전후를 살펴 후킹을 진행하며 해당하는 함수를 찾는다.

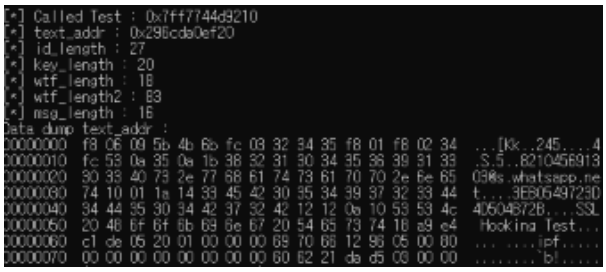


그림 9. 함수 발견

(3) 부채널 기반 탐색은 함수의 호출 횟수를 카운팅하여 찾는 방법으로 제시된 세 방법 중 가장 효율적이다. 본 방식은 SSL[6]이 send된 횟수와 함수가 호출된 횟수가 똑같거나 비슷하다면 SSL 암호화 루틴이라는 것에서 착안을 하였다.

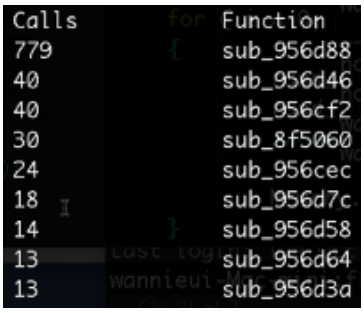


그림 10. frida-discover 함수 호출 카운팅

1. frida-discover[7]를 이용하여 함수 호출을 카운팅한 후 wireshark를 이용하여 SSL packet의 수를 카운팅한다.

2. 카운팅하여 같거나 비슷한 함수에 대하여 후킹을 진행한다.

Ⅳ. 결론

본고가 진행한 연구는 모든 SSL 암호화에 있어서 적용할 수 있는 탐색 방법이며, 현재 SSL을 사용하는 애플리케이션에서 적용할 수 있다. 이와 같이 SSL/TLS 암호화 전 패킷 추출을 하는 방법론을 통하여 취약점 분석을 진행할 경우 다음과 같은 행위를 할 수 있다.

1. 값 변조를 통한 공격

2. 라이브러리를 변조한 것처럼 퍼징 진행 가능

3. 프로토콜 분석

또한 암호화가 진행되기 전의 패킷 추출을 진행하였기 때문에 취약점 분석에 활용할 수 있는 다양한 정보들을 쉽게 확인할 수 있다.

[참고문헌]

[1] SSL 사용 이유, http://publib.boulder.ibm.com/tividd/td/ITLM/SC32-1431-01/ko_KO/HTML/tlminmst44.htm

[2] An overview of the SSL or TLS handshake, https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

[3] SSL/TLS 구성 프로토콜, http://www.ktword.co.kr/abbr_view.php?m_temp1=2552

[4] 김민우, 권태경, 메시지 인증 코드에 대한 연구 동향 분석 및 성능 비교, 정보과학회 논문지 제43권 제11호, 2016.11, 1245-1258

[5] 서화정, 김호원, NDK 기반 대칭키 암호화 구현에 관한 연구, 한국통신학회 종합학술대회 논문집(동계) 2012, 362-363

[6] SSL 암호화 구성, https://technet.microsoft.com/ko-kr/library/ms172455(v=sql.105).aspx

[7] frida, https://www.frida.re/

* 본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best)의 연구 지원으로 수행되었습니다. †주저자, best7ward@gmail.com ‡교신저자, anesra@gmail.com(Corresponding author)

Non-ActiveX 기반 금융보안모듈 대상의 보안 취약점 분석

이정민* 이주선 이진우 이준오 이시훈 박주현

*세종대학교 정보보호학과, 단국대학교 소프트웨어학과, 서울대학교 컴퓨터공학부, 고려대학교 정보보호학부, 아주대학교 사이버보안학과, 한세사이버보안고등학교 해킹보안과

Offensive Research of Non-ActiveX based Security Module

Jeong-Min Lee*, Ju-Sun Lee, Jin-Woo Lee, Jun-Oh Lee, Si-hoon Lee, Ju-hyun Park

*Department of Computer and Information security, Sejong University

Department of Software Engineering, DanKook University

Department of Computer Science and Engineering, Seoul University

Divsion of Information Security, Korea University

Department of Cyber Security, Ajou University

Division of Hacking Security, Hansei High School

요 약

URI-Scheme 방식으로 동작하는 Non-ActiveX 기반 금융보안모듈 프로그램은 기존 ActiveX 플러그인(Plug-in)과 같이 데몬의 형태로 사용자 PC의 백그라운드에서 항상 실행되는 형태이며, 브라우저와의 통신을 위해 열려진 포트가 네트워크 너머의 공격자에게 공격 벡터로 활용될 수 있어 여전히 보안 이슈를 해결하지 못한 상태이다. 브라우저를 통한 시스템 자원 제어가 가능한 취약점의 공통된 특징은 공격자 입장에서 높은 비용을 들이지 않고 파급력있는 공격을 수행할 수 있다는 점에서 중요하게 다루어져야 한다. 본 연구에서는 1금융권 16개 은행사에서 사용중인 Non-ActiveX 기반 금융보안모듈 클라이언트 프로그램의 취약점 분석을 통해, 발견한 취약점의 유형과 해당 취약점을 이용한 공격 모델을 소개하여 상용 금융보안모듈의 취약성과 위험성을 입증하였다.

I. 서론

1.1 Non-ActiveX기반 보안프로그램의 배경

은행권은 2015년 초, 정부의 ActiveX 폐지 지시에 따라 IE 뿐만 아니라 Safari, Chrome, Firefox 등 다양한 웹브라우저에서도 이용이 가능한 오픈뱅킹 서비스를 제공하게 되면서 ActiveX 기반 보안 프로그램들의 대부분을 URI-Scheme 방식으로 고안된 Non-ActiveX 기반 보안 프로그램으로 대체하였다. 이는 ActiveX 플러그인(Plug-in)과 같은 비표준 기술에서 발생하는 브라우저별 비호환성, 프로그램간 충돌, 보안성 이슈를 해결하고, 여러 플러그인 기술을 웹으로 수용하기 위한 웹 표준(HTML5)기술을 도입하려는 움직임과 함께한다.

그러나, 보안 분야 중 키보드 보안과 같이 특정 기술이 필요한 솔루션의 경우 웹 표준 기술로 대체할 수 있는 방법이 존재하지 않을 경우, 차선책으로 별도의 응용 프로그램을 제작하여 배포하거나, 브라우저와 응용 프로그램 간 서버 통신을 통해 서비스를 제공하는 방안을 대안으로 제시하고 있다. 실제로 현재 대부분의 솔루션은 별도 응용 프로그램과 데이터를 주고 받으며 기능하는 방식으로 구현되어 있다.

* 주저자, 교신저자 : 세종대학교 정보보호학과 학부 (ljm960905@gmail.com)

URI-Scheme 방식으로 동작하는 Non-ActiveX 기반 프로그램들 역시 ActiveX와 같이 데몬의 형태로 사용자 PC의 백그라운드에서 항상 실행되는 형태이며, 브라우저와의 통신을 위해 열려진 포트가 네트워크 너머의 공격자에게 공격 벡터로 활용될 수 있어 여전히 보안 이슈를 해결하지 못한 상태라고 볼 수 있다. 이렇게 브라우저를 통한 시스템 자원 제어가 가능한 취약점의 공통된 특징은 공격자 입장에서 높은 비용을 들이지 않고 파급력있는 공격을 수행할 수 있다는 점에서 중요하게 다루어져야 한다.

이미 라자루스, 안다리엘 그룹은 ActiveX, Non-ActiveX 취약점을 이용해 지속적으로 정부와 주요 기관, 기업에 대한 사이버 공격을 지속하고 있으며, 이는 프로그램과 통신 방식의 취약점으로 인해 국가기밀 유출, 기업정보 유출, 기반시설 마비 및 파괴와 같은 심각한 위협에 노출되어 있다는 것을 의미한다.

한국은행이 발표한 1분기 인터넷뱅킹 실 사용자는 6780만명이며, 온라인에서 금융서비스를 이용하기 위해서는 Non-ActiveX 기반 보안프로그램을 필수로 설치해야 하기 때문에 약 7천만 사용자의 PC가 위협에 노출되어 있는 상황이다.

따라서 URI-Scheme 방식으로 동작하는 Non-ActiveX 기반 보안모듈 클라이언트의 기능 및 동작원리 분석, Code Auditing을 통한 취약점 분석 연구를 통해 온라인 금융서비스 이용자에 대한 잠재적 위협을 제거하고자 하였다.

II. 취약점 분석

2.1 언패킹(Unpacking)

패킹(Packing)된 코드를 분석(Auditing)하기 위해서는 별도의 언패킹(UnPacking) 작업이 필요하다. 패킹된 대부분의 프로그램에서는 국내에서 가장 많이 쓰이는 패커인 Themida가 사용되었고, 대개 .exe 파일 혹은 .dll 파일에 패킹되어 있었다.

한 가지 특징으로는, 서비스(Service) 프로세스에는 거의 패킹이 되어 있지 않고 그의 자식 프로세스인 클라이언트(Client) 프로그램이 주로 패킹된다는 사실이다.

패킹된 exe파일은 대부분 VS/C, MS 링커(Linker)로 컴파일(Compile) 되어있어 OEP(Original EntryPoint)를 쉽게 찾을 수 있었다. C#으로 빌드된 1개를 제외한 모든 타겟들은 OEP Pattern Scan으로 후보군을 나열하여 반자동화 언패킹 하였다. API에 적용된 난독화(Obfuscation)는 Cheat Engine의 Lua Script를 이용하여 Follow를 구현하여 Jump Trick을 해제하였다. API 리다이렉션(Redirection)은 기존에 존재하는 툴인 UIF(Universial Import Fixer)를 사용하여 Fix하였다.

2.2 역공학(Reverse Engineering)

앞서 기술하였듯이, 패킹되어 있는 프로그램의 경우 언패킹 작업을 거친 뒤 역공학을 수행하였다. 역공학 수행과정에서 분석의 용이도를 높이기 위해, 심볼(Symbol)이 존재하지 않는 Windows 환경의 바이너리에 대한 심볼 복구를 우선적으로 수행하였다. 심볼 복구는 상용 디스어셈블러(Disassembler)인 IDA의 FLIRT 플러그인으로 시그니쳐(Signatures)들을 적용하여 해결하였다. 이 방법으로도 해결되지 않았을 경우, Mac OS X 환경에서 심볼이 존재하는 동일한 바이너리를 확인하여 분석을 시도하였다.

역공학을 통해 복구된 소스코드를 통해 단순 동적분석으로는 발견하지 못한 여러 핸들러와 함수를 확인할 수 있었다. 특히, 최초 개발 시점이 오래 되어 여러 차례 패치된 프로그램들의 경우 최신버전에서 사용하지 않는 기능을 위한 함수들이 여럿 발견되어, 기능상 오류를 발생시키지 않으면서 공격벡터(Attack Vector)로 사용될 수 있는 가능성이 있었다. 핸들러의 경우, 프로그램을 단순 동작해 보는 것으로는 확인할 수 없었던 설정 값 세팅, 배포(Distribute)정보 확인 등의 추가

* 주저자, 교신저자 : 세종대학교 정보보호학과 학부 (ljm960905@gmail.com)

기능들을 확인할 수 있었고, Custom URI-Scheme 프로토콜로 전송되는 핸들러 호출 명령어와 JSON 형식의 데이터가 처리되는 루틴(Routine) 또한 세부적으로 확인할 수 있었다.

2.3 의도하지 않는 입력(Unexpected Input)

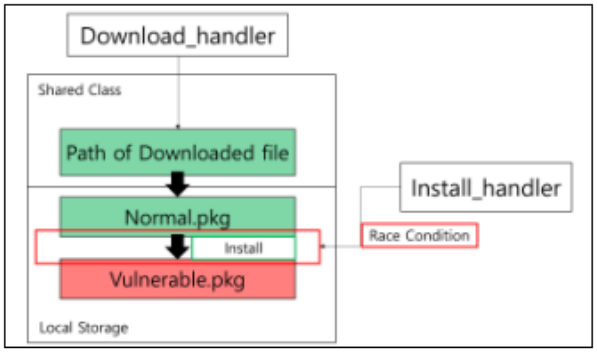
다수 보안모듈 클라이언트에 대한 기능분석 결과, URI-Scheme을 이용한 여러 JSON-RPC(Remote Procedure Call) 핸들러로 설정 값 세팅, 배포정보 확인, UI 구성, 파일 다운로드 등의 여러 행위가 가능함을 확인하였다.

실제로, 각 보안모듈이 동작하기 위해 이용하는 정상 URI-Scheme 문법을 준수하여 핸들러를 호출하고 파라미터 값을 세팅하여 전송하는 행위를 통해 보안모듈 클라이언트의 작동을 중지시키는 DoS(Denial of Service) 공격, 악성 파일 다운로드 공격, 임의 주소 쓰기 취약점을 통한 사용자 PC 장악 등의 악성 행위 발생의 가능성을 확인하였다.

III. 취약점 유형

3.1 Race Condition

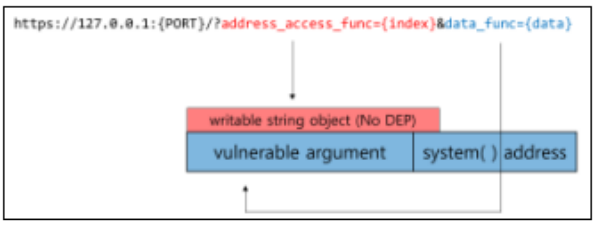
호출 시 동일 클래스(Class)를 참조하는 일부 핸들러의 경우, 해당 핸들러가 참조하는 메모리 자원이 공유되어 Race Condtion 공격이 가능한 상황이 발생하였다.



[그림1] Race Condition Vulnerability Model

3.2 Arbitrary Memory Write

URI-Scheme 프로토콜을 통한 JSON-RPC 동작 시 변수 또는 파라미터 값의 검증이 미흡하여 발생하는 임의 주소 쓰기(Arbitrary memory write) 가능성 또한 발견되었다.

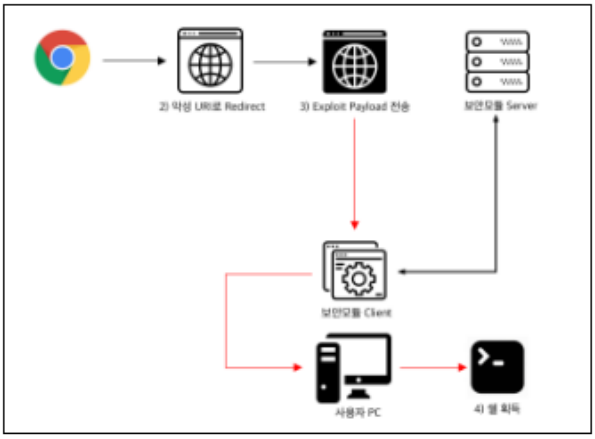


[그림2] Arbitrary Write Vulnerability Model

3.3 Attack Modeling

이외에도 명령 삽입(Command Injection) 취약점과 예외처리(Exception Handling)루틴이 부재하여 발생 가능한 원격 명령 실행(Remote Command Execution)의 가능성 또한 발견되었다.

이렇게 URI-Scheme 방식의 JSON-RPC를 이용하는 보안모듈은 브라우저를 통한 시스템 자원 제어가 가능하다는 점에서, 공격자 입장에서 높은 비용을 들이지 않고 파급력있는 공격을 수행할 수 있으므로 매우 중요하게 다루어져야 한다. 해당 공격 모델의 도식화 이미지는 다음과 같다.



[그림3] Non-ActiveX 기반 보안모듈 공격 모델

* 주저자, 교신저자 : 세종대학교 정보보호학과 학부 (ljm960905@gmail.com)

공격자가 악성 스크립트 혹은 웹페이지를 구성하여 사용자 PC에서 상시 동작하는 보안모듈 클라이언트에 위에서 언급한 취약점을 이용, Exploit Payload를 전송하게 된다면 사용자 PC를 장악하게 되어 인증서 탈취, 개인정보 유출, 크립토재킹(Cryptojacking), 좀비PC 양산 등의 막대한 피해를 입힐 가능성이 매우 높다.

또한 일부 보안모듈의 경우 사용자 권한이 아닌 관리자 권한으로 동작하는 프로그램도 존재하므로, 해당 프로그램에서 발생한 취약점을 통해 원격명령 실행이 가능하게 된다면, 원격에서 사용자 PC의 관리자 권한이 탈취되는 피해를 야기할 수 있다.

3.4 신규 취약점 목록

No	Product	Vulnerability Type	OS	RCE/LPE
1		ToC-ToU, Command Injection	OS X	RCE
2		ToC-ToU	OS X	RCE, LPE
3		Out of Bounds	OS X/Ubuntu/Fedora	RCE
4		Command Injection	Ubuntu/Fedora	LPE
5		Command Injection	Ubuntu/Fedora	RCE, LPE
6		Buffer Overflow	Windows7/10	RCE
7		Buffer Overflow	Windows7/10	RCE
8		Command Injection	OS X	LPE
9		Command Injection	OS X	LPE
10		Type Confusion	Windows7/10	RCE
11		Buffer Overflow	Windows7/10	RCE
12		Directory listing with root	Windows7/10, OS X	EOP
13		File copy with password and header	Windows7/10, OS X	EOP
14		Delete Certificate without password	Windows7/10, OS X	EOP
15		Permission Issue	OS X	EOP
16		Arbitrary File Write	Windows7/10, OS X Ubuntu/Fedora	EOP

[그림4] 9개 보안모듈에서 발견된 다수 취약점

실제로 취약점이 발견된 다수의 보안모듈은 1금융권 16개 은행사, 29개 증권사 및 공공기관 등에 배포되어 서비스되고 있었다. 현재 해당 보안모듈은 취약점 제보 및 패치가 완료되었고, 재배포 단계에 있는 상태이다.

IV. 결론

JSON-RPC를 위해 URI-Scheme Protocol 으로 동작하고, 플러그인으로 설치되는 Non-ActiveX 기반 프로그램들은 사용자 PC의 백그라운드에서 항상 실행되는 형태이며, 브라우저와의 통신을 위해 열려진 포트가 네트워크 너머의 공격자에게 공격 벡터로 활용될 수 있다. 해당 방식으로 동작하는 모듈에서 발생하는 취약점의 경우, 공격자 입장에서 높은 비용을 들이지 않고 파급력있는 공격을 수행할 수 있다는 점에서 중요하게 다루어져야 한다.

따라서, 주요 소프트웨어의 경우 개발단계부터 보안성 검토가 이루어져야 하며 배포 직전 단계에서도 취약점 점검 또는 모의해킹을 통해 보안성을 높이는 방법을 제안해볼 수 있다.

행정안전부는 2018년 11월, 공공 웹사이트 플러그인 제거 가이드라인을 배포하여 잠재적 보안위협을 제거하는 데에 노력을 기울이고 있다. 본 연구를 통해 기존 보안프로그램의 취약성이 검증되고, 새로운 정책에 부응 가능한 기술적인 후속 연구가 수행되기를 바란다.

[참고문헌]

[1] 인터넷 이용환경 개선을 위한 기술 안내서, 미래창조과학부, 한국인터넷진흥원, Dec, 2016

[2] 공공 웹사이트 플러그인 제거 가이드라인, 행정안전부, Nov. 2018

* 주저자, 교신저자 : 세종대학교 정보보호학과 학부 (ljm960905@gmail.com)

사이버전 피해평가 시뮬레이션 도구의 설계 및 구현

박진호*, 김도연*, 김두회*, 윤현수*, 김용현**, 김동화**, 신동규*, 신동일*
*세종대학교 컴퓨터공학과, **국방과학연구소

Design and Implementation of Simulation Tool for Cyber Battle Damage Assessment

JinHo Park*, DoYeon Kim*, DuHoe Kim*, HyeonSu Youn*, YongHyun Kim**, DongHwa Kim**, DongKyoo Shin*, DongIl Shin*

*Division of computer engineering, Se-jong University. **Agency for Defense Development

요 약

최근 몇 년 사이에 사이버 공격이 점차 보편화되면서 사이버 공간에서 이루어지는 공격을 사이버전이라는 일종의 전쟁으로 간주하고 있는 상황이다. 그러나 사이버전은 피해를 직접적으로 식별할 수 없기 때문에, 공격에 대한 아군의 피해를 파악하기 어렵다. 사이버전에서 발생할 수 있는 공격에 대한 피해를 효과적으로 평가하여, 이를 토대로 산출된 결과가 지휘 통제 단계에서 최종적인 판단에 도움을 주는 지표가 되기 위해, 본 논문은 MOCE(Measure of Cyber Effectiveness) 산출 식을 통해 사이버 공격에 대한 피해를 산출할 수 있는 DEVSim++ 기반의 사이버전의 피해 평가 시뮬레이션을 제시한다.

I. 서론

최근 몇 년 사이에 사이버 공격이 점차 보편화되면서 이는 국가 안보에 심각한 위협이 되고 있다. 그렇기 때문에 사이버 공간에서 이루어지는 공격을 사이버전이라는 일종의 전쟁으로 간주하고 있는 상황이다 [1]. 그러나 사이버전은 실제 물리전과는 달리 피해를 직접적으로 식별할 수 없기 때문에, 공격에 대한 아군의 피해를 파악하기 어렵다. 방위, 은행, 통신, 운송, 전력 및 기타 여러 시스템이 네트워크 인프라에 의존함에 따라, 시뮬레이션 방법론은 위협을 분류하고 결과를 평가하는데 필수적이다 [2].

본 논문은 사이버 공간에서 공격이 발생했을 때, MOCE 산출 식으로 피해량을 산출하는 DEVSim++ 기반의 피해평가 시뮬레이션을 제시한다. 시뮬레이션을 실행하여 산출된 결과인 피해율을 토대로 지휘 통제 단계에서 최종적인 판단에 도움을 주는 지표가 되는 것을 목표로 한다.

2장에서는 피해평가 시뮬레이션에 사용하는 도구와 자산 중요도 설정 방법, MOCE 설정을 위한 사이버 공격 효과 분류 모델을 소개한다. 3장은 피해평가 시뮬레이션의 구조와 피해량의 산출 식 및 가상 공격의 시나리오, 그리고 시각화된 최종 결과에 대해 설명한다. 4장에서는 결론 및 향후 연구를 내며 마친다.

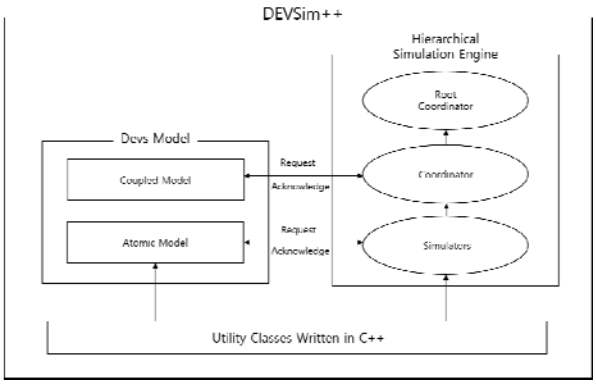
II. 관련연구

Musman은 사이버 공격으로 인해 발생한 피해에 대한 효과를 DMIFUI(Degradation, Modification, Interception, Fabrication, Unauthorized use, Interruption)의 6가지로 분류하고, 이 효과가 임무 데이터에 미치는 영향에 대해서 설명하였다 [3]. 사이버 지휘 통제 단계에서의 판단을 돕기 위해서는, 피해량을 보다 자세하게 공격자의 의도별로 나누어 산출하는 것이 필요하다. 이에 본 논문에서는 DMIFUI의 6 가지의 공격 의도 분류 기준 중 Interception,

Modification, Interruption을 선택하여 공격 효과에 대해 최종적인 피해량 산출 분류 기준으로 사용한다. Unauthorized use는 Interception, Fabrication은 Modification, 그리고 Degradation은 Interruption의 범위 내에 한정이 가능하므로 기준에서 제외한다 [4].

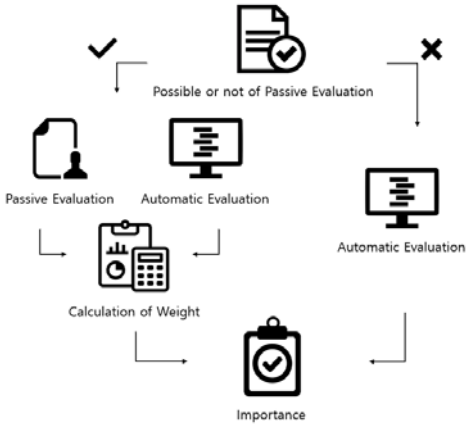
본 논문에서는 사이버 피해평가 시뮬레이션 구현 도구로 DEVSim++을 사용하였다. DEVS(Discrete Event System Specification)는 계층형태의 모듈 방식을 사용하는 추상적인 이산 이벤트 모델이다. 이를 기반으로 하여 시뮬레이션을 위한 모델링 및 추상 시뮬레이터 개념을 C++언어로 구현한 도구가 DEVSim++이다. 시뮬레이션의 실시간 상호작용 시스템을 표현하기 위해, DEVSim++은 계층적 스케줄링 알고리즘을 구현한다. 또한 모델링을 위해 Atomic 모델과 Coupled 모델을 DEVS 프레임워크 내에서 방향 그래프 형태로 제공한다 [5]. Atomic 모델은 입력 사건에 대한 상태 변화를 나타내는 개체로, 시뮬레이션에서는 하나의 PC로서 사용된다. Coupled 모델은 개체간의 관계를 나타내는 모델로, 시뮬레이션에서는 한 부대에 속한 PC Atomic 모델들을 연결하는 역할을 한다.

그림 1은 DEVSim++의 시뮬레이션 시스템 구조다. 이는 C++로 구현된 클래스가 시뮬레이션 엔진과 Atomic, Coupled 모델 간의 상호작용 스케줄링에 영향을 미치는 것을 보여준다.



(그림 1) DEVSim++ 시뮬레이션 시스템 구조

사이버 공격으로 인한 피해를 받기 전에 미리 사이버 자산의 중요도를 평가해놓는다면, 피해를 받은 사이버 자산의 중요도를 피해 산출식에 대입하여, 결과로 산출된 피해의 정확도를 높일 수 있다.



(그림 2) 사이버 자산 평가 시스템 구조

그림 2는 본 논문에서 설계한 사이버 자산의 중요도를 산출하는 자산 평가 시스템 구조이다. 자산의 중요도를 크게 기밀성, 무결성, 가용성의 세 분류로 나누어 평가한다. 기밀성은 군사 기밀 등급과, 무결성은 자산의 생성 날짜와 수정된 날짜의 기간차이, 가용성은 자산의 백업 파일 개수로 중요도를 산출한다 [6].

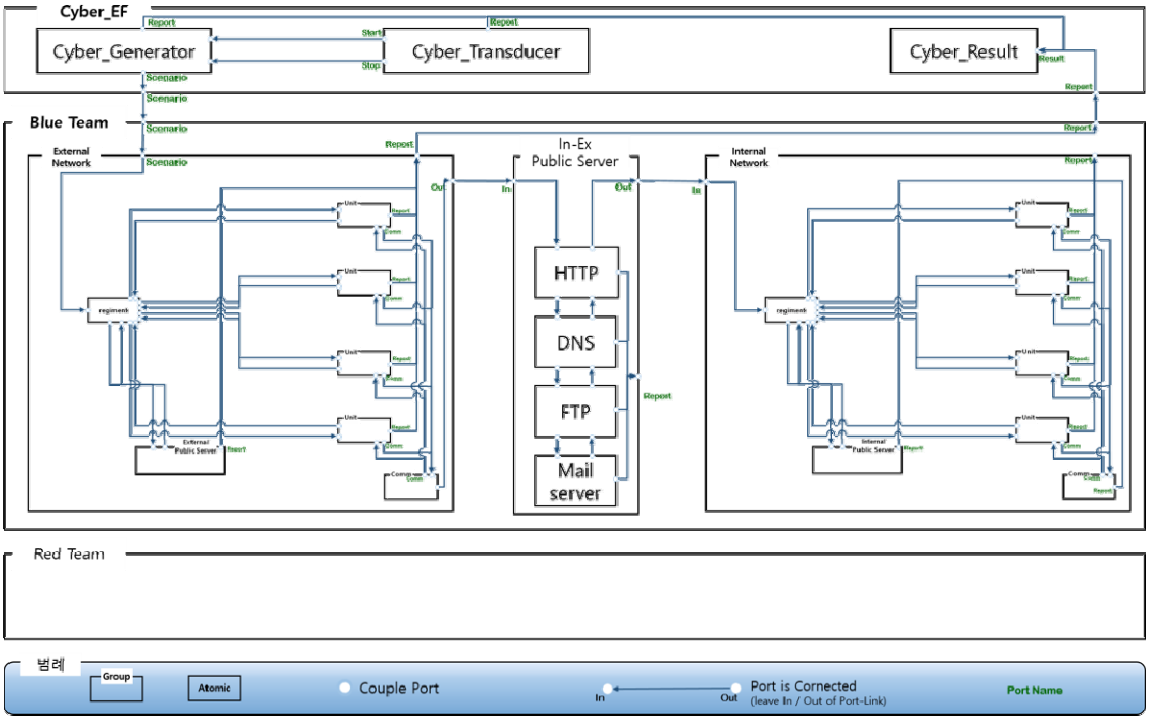
III. 피해량 산출과 시뮬레이션 도구의 개발

본 논문의 사이버 피해평가 시뮬레이션의 실행은 먼저 피해평가에 필요한 파일의 입력으로 시작한다. 입력 파일은 그림 2의 사이버 자산 평가 시스템을 이용하여 산출한 사이버 자산의 중요도, 보유한 사이버 자산의 개수 및 데이터의 크기 등을 속성으로 갖는다.

파일의 입력 이후에는 그림 3의 가상 네트워크 구조에 따라 Cyber_Generator 모듈에서 공격 시나리오가 생성되며 사이버 공격이 시작된다. 공격 시나리오는 외부 네트워크를 통해 공용서버로, 공용서버를 통해 내부 네트워크로 침투한다. 시나리오를 통해 공격을 받은 부대 단위 유닛 내의 PC들은, 공격의 종류마다 범위가 다르게 정해진 피해를 받는다.

공격이 모두 종료되면 피해를 받은 각 유닛마다 피해량을 텍스트 파일 형태로 출력한다. 파일의 출력 이후에는 DMIFUI 기반으로 제작한 공격 효과 분류 기준마다 최종 MOCE 피해

율을 산출한다.



(그림 3) 사이버전 피해평가를 위한 가상 네트워크 시스템 구조

$$\left\{\left(\sum_{i=1}^{n} W_i \times D_i \times R_i\right) \div \left(\sum_{j=1}^{n} W_j \times D_j\right)\right\} \times 100 \quad -(1)$$

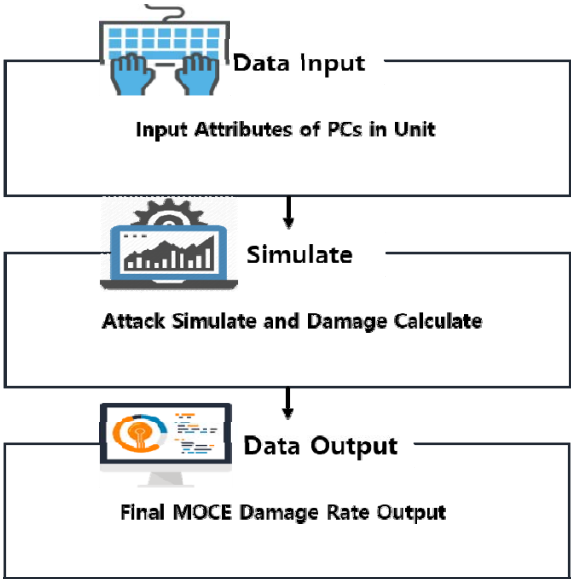
수식 1의 MOCE Interception은 탈취된 데이터의 양 및 중요도로 피해율을 산출한다. W_i는 i번째 데이터의 중요도, D_i는 i번째 데이터의 크기, R_i는 i번째 데이터의 탈취 여부를 의미한다.

$$\left\{\left(\sum_{i=1}^{n} DT_i \times DS_i \times R_i\right) \div \left(\sum_{j=1}^{n} DT_j \times DS_j\right)\right\} \times 100 \quad -(2)$$

수식 2의 MOCE Interruption은 시스템의 성능 저하와 그에 따른 작업 속도의 지연율로 피해율을 산출한다. DT_i는 i번째 PC(서버)의 피해 시간, DS_i는 i번째 PC(서버)의 피해 입은 파일의 가용메모리 크기, R_i는 i번째 PC(서버)의 네트워크 피해 여부를 의미한다.

$$\left\{\sum_{i=1}^{n} (W_i \times D_i + S_i) \times R_i \div \sum_{j=1}^{n} (W_j \times D_j + S_j)\right\} \times 100 \quad -(3)$$

수식 3의 MOCE Modification은 공격자에 의해 기존과 다르게 변경된 데이터의 양 및 중요도로 피해율을 산출한다. W_i는 i번째 데이터(소프트웨어)의 중요도, D_i는 i번째 데이터의 크기, S_i는 i번째 소프트웨어의 크기, R_i는 i번째 데이터(소프트웨어)의 수정 및 삭제 여부를 의미한다.

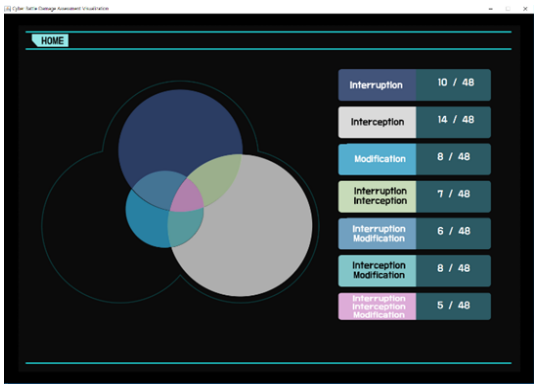


(그림 4) 피해평가 시뮬레이션 실행 순서

그림 4의 마지막 단계인 MOCE 최종 피해율 산출 이후에는 이를 시각화 프로그램을 통해 가시화한다. 사이버 지휘 통제 단계에서의 판단에 도움이 될 수 있도록, 그림 5와 같이 전체 피해율뿐만 아니라 각 유닛 별로 수식 1, 2, 3의 피해율을 가시화한다. 또한 PC보다 서버의 피해가 더 치명적이므로 PC와 서버의 피해율도 각 공격 분류 별로 나누어 표현한다.

(그림 5) 가상 공격에 대한 피해율 가시화

또한 각 Unit내에서 피해를 받은 공격분류마다의 PC와 서버 개수를 알게 되면, 피해 분류 별로 대처가 가능하기 때문에 그림 6과 같이 각 공격의 효과 분류별로 PC와 서버 개수를 그래프 형태로 표현한다.



(그림 6) 피해 PC/서버별 받은 공격의 가시화

IV. 결론

본 논문에서는 MOCE 산출 식을 통해 사이버 공격에 대한 피해를 산출할 수 있는 사이버전의 피해 평가 시뮬레이션 방안을 제시했다.

추후에 시뮬레이션을 반복 수행하여 산출한 결과물과 실제 사이버 공격으로 피해를 입은 결과물로 데이터셋을 구성하여, 시뮬레이션에 대해 신뢰성을 증명할 예정이다.

ACKNOWLEDGMENT

본 연구는 방위사업청과 국방과학연구소의 지원으로 수행 되었습니다(UD160066BD).

[참고문헌]

[1] Hathaway, O. A et al. 2012. "The Law of CyberAttack." California Law Review 100 (4), pp. 817-885. Aug. 2012.

[2] Chi, Sung-Do et al. "Network Security Modeling and Cyber Attack Simulation Methodology." Springer, 6th Australasian Conference, ACISP 2001, pp. 321-333 Jul. 2001.

[3] Musman., Scott., et al.. "Computing the impact of cyber attacks on complex missions." Systems Conference (SysCon), 2011 IEEE International. IEEE, pp. 46-51, Apr. 2011.

[4] 박진호 et al. "사이버전 피해 평가를 위한 사이버 공격의 분류 체계 제시." 2017 한국정보처리학회 추계학술발표대회, 학술대회 논문집, pp. 235-238, Nov. 2017.

[5] T. G. Kim, C. H. Sung, S.-Y. Hong, J. H. Hong, C. B. Choi, J. H. Kim, K. M. Seo, and J. W. Bae, "DEVSim++ Toolset for Defense Modeling and Simulation and Interoperation." The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, pp. 129-142, Nov. 2010.

[6] 박진호 et al. "사이버 자산의 중요도 산출 방안 제시." 2018 한국통신학회, 하계종합학술발표회 논문집, pp. 365-366, Jun. 2018.

임베디드 리눅스 시스템의 보안 설정 검사 방법 연구†

김서영*, 권태경*

*연세대학교 정보대학원 정보보호 연구실

A Study of Security Configuration Checking for Embedded Linux Systems

Seoyoung Kim*, Taekyoung Kwon*

*Information Security Lab, GSI, Yonsei University

요 약

임베디드 리눅스는 IoT 장비에서 많이 가장 많이 쓰이는 운영체제로 IoT 환경이 구축됨에 따라 임베디드 리눅스 시스템의 보안이 더욱 중요해지고 있다. 현재 발표된 IoT 보안 가이드는 포괄적인 보안 방법을 제시하며 기술적으로 구체적인 보안 설정 방법을 제안하는 가이드는 부족한 실정이다. 본 논문에서는 OpenSCAP의 체크리스트를 기반으로 88개 규칙으로 이루어진 임베디드 시스템 보안 설정 체크리스트를 제시하고 임베디드 리눅스 시스템 환경에서 제시한 체크리스트를 기반으로 보안 설정 검사가 가능하도록 최적화 하는 방법을 제안한다. 제안한 방법을 바탕으로 도구를 설계하여 임베디드 리눅스 배포판인 OpenWrt에 적용하였다.

I. 서론

임베디드 리눅스는 안정성, 확장성, 휴대성, 오픈소스 사양 및 표준, 지원 및 대형 프로그래머 기반, 무료 가용성 등의 기능을 제공[5]하여 응용 프로그램 개발 시간을 단축 할 수 있다는 장점 때문에 IoT 장비에서 많이 사용되고 있다. IEEE의 The Eclipse IoT Working Group이 발표한 IoT Developer Survey 2018에 따르면, IoT 장비의 71.8%가 Linux를 사용하여 높은 점유율을 보유하고 있다는 것을 알 수 있다. 이에 IoT 환경이 활발히 연구개발 되고 환경이 구축되기 시작하면서 임베디드 리눅스 시스템의 보안은 더욱 중요해질 전망이다. 하지만 현재 발표된 IoT 보안 가이드라인은 포괄적인 가이드라인을 제공하며, 구체적으로 리눅스 시스템 설정 값을 어떻게 구성하라는 지침이 명시되어 있지 않다[7][8].

UNIX 서버, 윈도우즈서버, 제어시스템, PC, 네트워크 장비를 대상으로 점검 및 조치 방법을 나타내는 가이드는 지속적으로 업데이트 되어 한국 인터넷 진흥원에서 문서로 제공한다[4]. 그러나 이런 문서를 그대로 임베디드 리눅스 시스템에 반영하기에는 임베디드 리눅스에 해당 사항이 없는 경우가 빈번하다. 따라서 구체적인 리눅스 보안 설정 방법이 임베디드 리눅스 시스템에도 필요하다.

OpenSCAP은 NIST(National Institute of Standards and Technology)가 제시한 정책에 정의된 모든 보안 설정 규칙을 하나씩 검토하고 규칙을 만족하는지 보고서로 결과를 출력하는 응용 프로그램을 제공한다. 보안 관리 업무를 지닌 모든 기관은 OpenSCAP이 제공하는 가이드와 스캐닝 도구를 이용하여 보안 설정 검사를 자동화하여 시간과 비용을 절약할 수 있다. 그러나 OpenSCAP은 임베디드 시스템을 검사하기에는 너무 큰 용량과 자원이 요구되며, 임베디드 시스템을 위한 체크리스트를 지원하지 않는다는 한계가 있다.

본 논문에서는 임베디드 리눅스 보안 설정 검사 체크리스트를 제시하고, 임베디드 시스템 환경에서 체크리스트 기반으로 보안 설정 검사를 진행 할 수 있도록 최적화 하는 방법을 제안한다. 체크리스트 작성을 위해 OpenSCAP 가이드에서 임베디드 시스템에 적용할 수 있는 규칙만 추출하여 적용시키는 방법론을 사용하였다. 임베디드 시스템 보안 설정 검사를 위해 busybox를 사용하고 기본 sh 쉘을 사용하여 임베디드 시스템에 최적화 하는 전략을 가지고 접근

† 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00304),

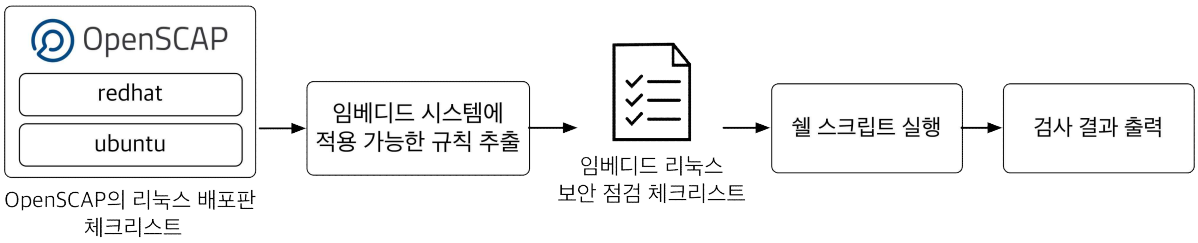


Fig. 1. Embedded linux security settings scan tool structure

하는 방법을 제안하였다. 제안한 방법을 쉘 스크립트로 구현하여 임베디드 리눅스 배포판인 OpenWrt 환경에 적용시켜 검사 항목이 실제 임베디드 환경에서 어떻게 적용되는지 확인하였다.

II. 연구 배경

2.1 OpenSCAP

SCAP (Security Content Automation Protocol)은 자동화 구성, 취약성 및 패치 확인, 기술 제어 준수 활동 및 보안 측정을 지원하는 다목적 프레임 워크이다[1]. OpenSCAP은 SCAP을 준수하고 시스템 관리자 또는 보안 감독관이 보안기준을 평가, 측정 및 집행하는데 도움이 되는 여러 도구를 오픈소스 형태로 제공한다[2].

OpenSCAP은 NIST를 비롯한 여러 국가 기관에서 제시하는 보안 정책에 따라 시스템이 정책을 준수하는지 정책에 정의된 모든 보안설정 규칙을 하나씩 검토하고 규칙을 만족하는지 보고서로 결과를 출력하는 응용 프로그램을 제공한다. 보안 관리 업무를 지닌 모든 기관은 OpenSCAP이 제공하는 스캐닝 도구를 이용하여 보안 설정 검사를 자동화하여 시간과 비용을 절약할 수 있다.

오픈 소스 커뮤니티에서 개발한 다양한 강화 가이드 및 구성 기준선을 제공한다. 다양한 리눅스 배포판(e.g. Redhat, Ubuntu, Centos)을 대상으로 NIST나 ANSSI에서 배포한 신뢰성 있는 보안 가이드 문서를 준수하는 보안 설정 체크리스트를 상세히 지원한다. 그러나, 임베디드 리눅스를 위한 보안 설정 검사 리스트는 지원하지 않고, 프로그램이 임베디드 환경에서 구동되기는 너무 무겁다는 한계가 있다.

OpenSCAP의 리눅스 배포판들의 체크리스트를 기반으로 임베디드 리눅스의 체크리스트를 생성한다. OpenSCAP이 제공하는 체크리스트는 보안 정책을 준수하므로, 임베디드 리눅스 시스템에도 정책을 준수하는 체크리스트를 제시할 수 있다.

2.1 임베디드 리눅스 시스템 보안

임베디드 시스템을 위한 것이 아닌 일반적으로 서버에 적용되는 상세한 보안 가이드라인은 많은 연구 결과가 나오고 있다. 2014년에 한국인터넷진흥원에서는 주요 정보통신기반 시설을 대상으로 기술적 취약점 분석하는 상세 가이드를 지속적으로 업데이트하고 있다[4]. 임베디드 시스템에 적용할 수 있는 체크리스트와 상세한 가이드 라인이 필요하다.

2004년 Koopman는 임베디드 시스템은 적은 CPU 처리 능력으로 인해 보안을 위해 소모하는 자원에 민감하다고 주장하였다[3]. 이러한 임베디드 시스템의 특징은 검사과정에서 많은 파워가 소모되지 않도록 해야한다는 것을 알 수 있다. 비용에 민감한 임베디드 시스템 특징 때문에 출시되는 임베디드 장비들은 busybox가 설치되어 배포된다. busybox는 유닉스 명령어들을 경량화 버전으로 지원하는 소프트웨어로, 임베디드 시스템에 최적화 되어있다.

따라서 본 연구에서는 임베디드 시스템의 보안 설정 검사 진행을 위해 busy box 명령어를 사용한 쉘 스크립트로 진행하는 방법을 제안한다.

III. 보안 설정 검사 도구 설계

Fig. 1.은 임베디드 리눅스 시스템의 보안 설정 검사를 위해 제안한 도구의 전체 구조이다. OpenSCAP에서 제공하는 redhat과 ubuntu의 체크리스트에서 임베디드 시스템에 적용 가능한 검사 규칙을 추출하여 임베디드 리눅스 시스템 보안 설정을 위한 체크리스트를 구성하였다. 이는 busybox 명령어를 사용한 sh 쉘 스크립트로 보안 설정을 자동으로 검사하며 모든 규칙을 검사한 후에는 수정해야 할 보안 설정 값을 리포트 형식으로 제공한다.

3.1 보안 점검 체크리스트

3.1.1 체크리스트 선별 기준

OpenSCAP에서 제공한 체크리스트를 분석한 결과 다음과 같은 경우를 제외한 모든 규칙을 임베디드 시스템에 적용 가능하다고 판단하여 선별하였다.

- 보안을 위해 새로운 프로그램을 설치하는 경우
- 검사해야하는 설정 파일이나 폴더가 임베디드 리눅스 시스템에서 지원되지 않는 경우
- GUI 인터페이스 관련 규칙

이렇게 추출한 검사 규칙은 Hardening the filesystem, File Permissions and Mask. Account and Access Control, Network Configuration and Firewalls, Remote Control과 같이 5개의 그룹으로 분류하였으며 총 검사하는 체크리스트 규칙은 88개이다.

Table 1. Number of system scan rules by group

Group	# of Rule
Hardening the filesystem	4
File Permissions and Mask	22
Account and Access Control	27
Network Configuration and Firewalls	24
Remote Control	11
total	88

3.1.2 그룹별 시스템 검사 규칙

Table 1은 그룹별로 시스템 검사 규칙 수를 나타냈다. 각 그룹은 다음과 같은 목적을 지니고 검사를 수행한다.

Hardening the filesystem 파일 시스템을 강화함으로써 서비스, 데이터 및 구성을 효율적으로 분리하고, 파일 시스템의 수준에서 접근 권한 레벨을 정확하게 관리할 수 있다. 또한 리눅스 파일 시스템은 다양한 설정을 할 수 있는 sysctl 명령어를 지원한다.

File Permissions and Mask 전통적인 Unix 보안은 사용자에게 권한이 없는 파일을 읽거나 수정할 수 없도록 하는 파일 및 디렉터리 사용 권한에 크게 의존한다. 특정한 권한을 가진 파일 또는 디렉토리를 검색하여 권한 설정이 잘 되어있는지 확인하고, 권한이 없는 사용자가 접근 할 수 없도록 커널 변수 값을 설정하도록 한다.

Account and Access Control 공격자가 특정 계정에 대해 접근 권한을 얻는다면 취득한 계정의 권한을 가지고 허용되는 범위에서 작업을 수행할 수 있다. 따라서 승인되지 않은 사용자가 쉘 접근 권한을 얻을 수 없도록 계정을 보호 하는 것은 시스템을 보호하는 중요한 요소이다.

Network Configuration and Firewalls 임베디드 시스템의 대부분은 네트워크에 연결되며 이로 인해 위협받을 가능성이 커진다. 공격자가 시스템에 연결 할 수 없도록 특정 IP 주소, 호스트 및 네트워크에서 트래픽을 허용 또는 거부하도록 지정하는지 검사한다.

Remote Control 임베디드 시스템의 대부분은 배포 후 관리를 위해 SSH나 Telnet과 같은 원격 연결을 지원한다. 그러나 개방된 포트로 인해 공격자에게 쉽게 노출이 되므로 가능한 SSH나 Telnet을 사용하지 않도록 권장하며 불가피하게 필요할 경우 보안 설정을 통해 엄격히 관리되어야한다.

```
[+] Hardening the Filesystem
----------------------------------------
  * Ensure /home, /tmp, /var Located On
Separate Partition
    - Checking /home mount point
      [ NOT FOUND ] directory /home does not
exist
    - Checking /tmp mount point
      [ OK ] found /tmp as a separated mount
point
    - Checking /var mount point
      [ WARNING ] Directory most likely stored
on / file system
  * Disallow creating symlinks to a file you
not own
    - Checking /proc/sys/fs/protected_symlinks
      [ WARNING ] It Should be set to 1

...

[+] RESULT
----------------------------------------
Total Checking Count = 107
OK COUNT = 17
WARNING COUNT = 55
NOT FOUND COUNT = 35
```

Fig. 2. Result of system setting check

3.2 시스템 검사 방법

3.2.1 많은 임베디드 리눅스 지원

체크리스트의 모든 규칙은 쉘 스크립트로 검사하여 시스템 관리자 또는 개발자가 직접 일일이 확인할 필요가 없도록 자동 검사를 진행한다. busybox 명령어를 사용하여 특정 명령어를 지원하지 않는 임베디드 시스템에서도 검사 가능하도록 쉘 스크립트를 작성하였다. 쉘 스크립트는 bash 쉘이 아닌 기본 sh를 사용하여 bash 쉘을 지원하지 않는 환경에서도 검사 가능하도록 한다. sh는 POSIX 표준에 지정된 기본 쉘 구문만을 사용하여 이식성이 좋다. /bin/sh는 모든 unix, linux 시스템에서 작동한다. 많은 임베디드 리눅스 시스템에서 검사가 가능하도록 busybox 명령어와 sh 쉘을 사용하여 검사 과정을 임베디드 시스템에 최적화 하였다.

3.2.2 검사 결과 출력

각 규칙을 검사한 후에는 OK, Suggestion, Warning, Not Found와 같이 세 종류의 결과가 출력된다. **OK** 안전한 값으로 설정되어 있어 설정 변경이 필요 없는 경우이다. **Warning** 보안을 위해 권고하는 값으로 설정이 되어 있지 않아 안전한 설정으로 변경해야하는 경우로, 어떤 설정을 변경해야하는지 상세한 지표를 제시한다. **Not Found** 임베디드 리눅스 시스템 마다 구조가 다를 수 있다. 이 경우에는 규칙에서 검사해야하는 요소(파일이나 폴더)가 존재하지 않는 경우이다.

IV. 보안 설정 검사 도구 테스트

4.1 테스트 적용 환경

제안한 보안 설정 검사 도구를 임베디드 리눅스 시스템 환경에서 테스트하였다. 대상은 임베디드 리눅스 배포판인 OpenWrt로, 무선랜 라우터를 비롯한 임베디드 장치를 대상으로 하는 오픈소스 Linux 운영체제이다. OpenWrt는 무선랜 라우터에 필요한 리눅스 기능들을 패키지 형태로 제공하여 사용들에게 확장성과 편의성을 제공한다. 가상환경으로 OpenWrt를 설치하여 제안한 도구를 테스트하였다.

4.2 스크립트 실행 결과

Fig. 2.는 보안 설정 검사 스크립트 실행 결과의 일부를 나타낸다. 각 규칙에 따라 결과가 출력되어 Warning의 경우 보안 설정 권고와 함께 출력되는 것을 확인 할 수 있다. 마지막에는 전체 검사한 설정값의 전체 카운트와 OK 수, Warning 수, Not Found 수를 확인 할 수 있다. 시스템 관리자 및 개발자는 보안 설정을 수정한 후 다시 검사를 진행하여 모두 OK를 받았는지 확인할 수 있다.

OpenWrt를 대상으로 테스트 결과 총 107개의 검사 항목중에서 OK는 17개, Warning은 55개, NotFound는 35개가 검사된 것을 확인할 수 있었다.

V.결론

본 논문에서는 임베디드 리눅스의 보안 설정 체크리스트를 제안하고 시스템 검사 방법을 제안하였다. 임베디드 시스템을 위한 보안 설정 검사 리스트를 제안하기 위해 OpenSCAP의 가이드를 통해 표준을 만족하는 체크리스트를 제시하는 방법론을 사용하였다. 임베디드 환경에서 검사가 가능하도록 busybox를 사용하고 기본 sh를 사용하여 임베디드 시스템에 최적화 하는 방법을 제안하였다. 제안한 방법을 제안한 방법을 쉘 스크립트로 구현하여 임베디드 리눅스 배포판인 OpenWrt 환경에 적용시켜 검사 항목이 실제 임베디드 환경에서 어떻게 적용되는지 확인하였다.

향후 시스템 보안 관리자 또는 개발자의 시스템에서 원격으로 임베디드 장비의 보안설정을 검사하여 관리자의 환경에서 결과를 받을 수 있도록 보완한다면 현업에서 사용할 수 있는 도구가 될 것이다.

[참고문헌]

[1] S. Radack and R. Kuhn, Managing Security: The Security Content Automation Protocol, in IT Professional, vol. 13, Jan.-Feb. 2011.

[2] https://www.open-scap.org/ , OpenSCAP 공식 홈페이지

[3] P. Koopman, Embedded system security, in Computer, vol. 37, no. 7, pp. 95-97, July 2004.

[4] 한국인터넷진흥원, 주요정보통신기반시설 기술적 취약점 분석 평가 방법 상세가이드, Dec. 2017

[5] A. Lennon, Embedding Linux, in IEE Review, vol. 47, no. 3, pp. 33-37, May 2001.

[6] The Eclipse IoT Working Group, AGILE IoT, IEEE, IoT Developer Survey 2018, https://www.slideshare.net/kartben/iot-developer-survey-2018, Apr 13, 2018

[7] 한국인터넷진흥원, 홈·가전 IoT 보안가이드, 7, 2017

[8] GSMA, GSMA IoT Security Assessment Checklist, version 3.0, Sep, 2018

국내 핀테크 정책 현향*

김미연* 염흥열**
*순천향대학교 융합서비스 보안학과 **순천향대학교 정보보호학과

Fintech Policie in Korea

Mee Yeon Kim* Heung Youl Youm**
*Convergent services of security department, Soonchunhyang University.
*Division of information security, Soonchunhyang University.

요 약

핀테크는 금융서비스에서 사용할 수 있는 기술을 광범위하게 포함한다. 국내 핀테크 산업에서는 간편 결제·송금 서비스가 생겨나는 등 다양한 핀테크 서비스가 늘어나고 있다. 결제·송금뿐만 아니라 인터넷 전문은행, 자산관리, 인슈테크 등 다양한 분야에 핀테크 기반 서비스가 제공되고 있다. 국내외적으로 블록체인, 간편결제/송금, 바이오인증 등이 주요 핀테크 기술로 대두되고 있으며, 본 논문에서는 제시된 3개 영역의 정책동향을 비롯하여 전반적인 핀테크 정책 동향을 파악하였다.

I. 서론

국내 핀테크 산업에서는 간편 결제·송금 서비스가 생겨나는 등 다양한 핀테크 서비스가 늘어나고 있다. 결제·송금뿐만 아니라 인터넷 전문은행, 자산관리, 인슈테크 등 다양한 분야에 핀테크 기반 서비스가 제공되고 있다.

핀테크는 금융서비스에서 사용할 수 있는 기술을 광범위하게 포함한다. 제3자의 개입을 제거하여 수수료를 절약할 수 있는 블록체인, 차세대 본인인증 방식으로 각광받는 바이오 인증, 간편 지불결제에 사용되는 NFC, MTS, QR 코드, 자산관리 포트폴리오를 자동으로 산출해내는 로보어드바이저에 사용되는 AI 등 금융서비스에 접목되어 서비스를 구성하는 기술이라면 모두 핀테크에 포함될 수 있다.

*본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00304)

본 논문은 금융 산업에서 주요 핀테크 기술로 대두되고 있는 블록체인, 모바일 결제, 간편결제/송금 등을 포함한 지불결제, 바이오 인증을 주요 핀테크 기술로 선정하여 이에 대한 정책 동향을 파악하고자 한다.

본 논문의 2장에서는 제시된 3개의 핀테크 기술에 대한 간략한 정의 및 설명을 기술하고, 3장에서는 전반적인 핀테크 정책 동향을 포함하여 제시된 3개의 핀테크 정책 동향을 정리하였다.

II. 핀테크 요소 기술

2.1 블록체인

블록체인이란 "블록(block)을 잇따라 연결(chain)한 모음"이다. 사용자는 개별적으로 블록체인 사본을 보관하며, 거래내역을 블록에 담는다. 블록에는 확정된 거래내역이 기록되며, 모든 사용자에게 검증을 거쳐 과반수이상의 동의

를 받은 거래내역만을 유효한 거래로 인정하고 블록을 생성한다. 새로 생성된 블록은 해시를 통해 이전 블록과 연결되며 일련의 과정이 블록 생성 시간에 따라 반복된다.[1]

블록체인 기반 핀테크 기술 및 서비스는 대부분의 금융 거래에서 불가결한 상대방의 개인정보와 신용 정보를 보증받을 필요가 없다. 모든 거래 정보가 블록체인에 기록되고 거래를 위변조하는 것이 거의 불가능하기 때문이다. 블록체인은 국가 간 경계가 적용되지 않기 때문에 글로벌 상호 운용 시, 보다 저렴한 수수료로 가치 교환이 가능하다. 또한 블록체인을 이용하여 개인끼리의 안전하고 저렴한 대출 서비스, 크라우드 펀딩 서비스 등이 제공될 수 있다.[1]

2.2 간편결제

간편결제는 "온/오프라인 상거래에서 빠르고 간편하게 결제하는 전자 결제, 송금 서비스"를 일컫는다. 기존 결제 프로세스를 간소화하여 공인인증서나 보안카드 같은 불편함을 해결해준 대표적인 핀테크 서비스로, 스마트 기기에 생체정보나 결제정보 등을 저장하여 바로 결제 및 송금에 이용하는 것이 대표적이다. 대표적인 예로는 온라인 기반의 카카오페이나 페이코, 신용카드 기반의 삼성페이, 앱투앱 서비스, 모바일 결제 서비스 등이 있다.[2]

2.3 바이오 인증

바이오인증 기술은 지문, 홍채, 얼굴, 정맥등 개인 고유의 신체적 또는 서명, 음성 등 행동적 특징을 통해 본인을 인증하는 방식으로 기존 인증방식 보다 우수한 편의성과 보안성으로 주목 받으며 금융 등 여러 분야에서 활용되고 있다. 특히, 스마트뱅킹 등 스마트폰 기반 서비스의 확대와 함께 바이오인증 기술 도입 또한 급증하고 있으며, 다양한 활용사례들이 발표되고 있다.

III. 핀테크 정책 동향

3.1 핀테크 정책 동향

과학기술정보통신부와, 금융위원회에서는 전

표 1 핀테크 전반에 대한 정책 동향[3][4][5]

정책기관	주제	세부내용		발표 날짜
과학기술정보통신부	핀테크 산업 활성화 기반 조성	16년 30억 원, 17년 29억 원 예산		2017-05-18
		핀테크 서비스 개발 및 테스트 환경 제공	FIDO, 블록체인 등 기술 테스트환경 제공 및 기반기술 활성화 추진	
		핀테크 서비스 보안컨설팅 및 인력양성 지원	핀테크 서비스 개발 및 운영 전반에 대한 보안 컨설팅 지원	
			핀테크 특화 인력 양성	
		핀테크 서비스 테스트패널 운영	이용자 반응 및 니즈 분석	
		이노베이션 허브 구축 관리	기반기술 시험 환경 및 인프라 관리, 지원 프로그램 운영	
과학기술정보통신부	공인인증서 제도/규제 폐지	전자서명법 전부개정안 입법 계획 발표	기존 공인인증서 제도 폐지	2018-03-29
			공인/사설인증서 구분 폐지	
			전자서명인증업무 평가제 도입	
금융위원회	핀테크 혁신 발전전략	혁신적 금융서비스 실험, 지원	금융혁신지원 특별법 제정 추진	2018-03-20
			금융 테스트베드 시행	
			민간투자, 해외진출, R&D지원	
			지원센터 확대/강화 및 전문인력 교육/양성	
		금융권 서비스 고도화	로보어드바이저, 비대면 거래 확대, 크라우드 펀딩 개선 등 자본시장 핀테크 활성화	
			인슈테크 도입 촉진, 보험분야 혁신 촉진 등 인슈테크 활성화	
		핀테크 시장 확대	모바일 간편결제, 빅데이터, 오픈 API, 블록체인, 클라우드 활성화	
		핀테크 혁신 리스크 대응	혁신기술 보안진단 및 컨설팅 지원 등 보안대응 체계 강화	
			시범사업으로 레그테크 활용분야 확대	

반적인 핀테크 산업의 촉진 및 활성화, 적기에 핀테크 도입활용을 저해하는 법령 재논의 등을 위해 정책과 전략을 발표하였다. 17년 5월 과학기술정보통신부는 핀테크 산업 활성화 기반 조성을 위해 연간 약 30억 원의 예산을 투자하여 핀테크 산업의 혁신/성장 전략을 수립하고 혁신 주도의 기술 생태계를 조성하고자 하였다. 주요 내용으로는 '핀테크 서비스 개발 및 테스트 환경 제공, 핀테크 서비스 보안컨설팅 및 인력양성 지원, 핀테크 서비스 테스트패널 운영, 이노베이션 허브 구축 관리' 등을 제시하였다.[3]

18년 3월에는 공인전자서명과 사설 전자서명의 법적 차별을 제거하며 20년 만에 공인인증서 제도의 전면 폐지 추진을 발표하였다. 이를 통해 사설인증기관의 경쟁시장을 조성하는 한편, 전자서명인증업무 평가제를 도입하여 민영화 시 발생할 수 있는 보안 문제 등을 해결하고자 하였다.[4]

금융위원회에서도 같은 시기에 금융권 전 분에 적용할 "핀테크 혁신 발전전략"을 발표하였다. 내용으로는 금융혁신지원특별법 제정 추진과 테스트베드를 운영하는 지원 전략, 비대면 거래 확대, 인슈테크 도입 등이 포함되는 서비스 고도화 전략, 간편결제 활성화, 블록체인 테스트베드 구축 등 핀테크 시장 확대 전략, 보안대응, 레그테크 활용 등을 포함한 혁신 대응 전략 등 4가지 전략으로 구분된다.[5]

전반적인 핀테크 정책 동향이 [표 1]에 정리되어있다.

3.2 블록체인 분야 정책 동향

과학기술정보통신부와 금융위원회는 블록체인 기술을 활용하기 위한 산업적 기반을 다지기 위하여 여러 정책 지원을 마련하였다.

금융위원회는 16년 말 11월에 블록체인 협의회를 출범하여 금융권 공동 블록체인 컨소시엄의 운영계획을 발표하였으며 컨소시엄에서 개발한 금융권 블록체인 표준이 19년 내로 배포될 예정이다.[6]

과학기술정보통신부는 블록체인 산업 활성화 및 기술경쟁력 확보를 촉진하고자 [표 2]와 같은 블록체인 기술 발전전략을 선포하였다. 추진 분야는 초기시장형성, 기술경쟁력 확보, 산업 활성화 기반 조성 세 개의 분야로 나뉜다. 첫째로 블록체인 초기시장 형성을 위해 6대의 시범사업을 추진하고자 하며, 이에는 축산물 이력관리, 통관서비스, 부동산 거래, 온라인 투표, 국가전자문서 관리, 해운물류 관리 등이 포함된다. 또한 민간에서 주도적으로 블록체인 혁신을 이끌게 하기 위한 지원으로써 8대 선도사업(초연결 지능화, 스마트공장, 스마트팜, 핀테크, 에너지신사업, 스마트시티, 드론, 미래자동차)에 블록체인을 우선적으로 도입하고자 하였다.[7]

표 2 과학기술정보통신부 블록체인 기술 발전전략[7]

추진 과제	설명	추진 내용
초기 시장 형성	선제적 공공선도 사업추진	공공선도 사업 추진으로 업무 효율화 (18년 6개, 19년부터 확대)
	민간주도 개방형 혁신 지원	민간주도 블록체인 국민 프로젝트 진행 (매년 3개 민간매칭 19년~)
기술 경쟁 력 확보	블록체인 핵심기술 확보	핵심기술 확보 (선진국 대비 90% 기술 달성 목표)
	블록체인 기술 지원 센터	신뢰성, 성능 평가 및 개발 테스트베드 제공 (대규모 시험환경 제공, 19년~)
	표준화 활동 강화	표준화 로드맵 고도화 및 표준화 활동 전문가 지원 확대
산업 활성 화 기반 조성	핵심인력 양성	실무인력 양성을 위한 놀이터 구축 및 블록체인 연구센터 지정확대 (전문 인력: 17년 약 6백 명 22년 1만 명)
	블록체인 전문기업 육성	블록체인 창업 여건 조성 및 글로벌 진출 확대 (전문기업수: 17년 30여개, 22년 100개
	법제도 개선	규제개선 연구반 운영
	인식제고	블록체인 진흥주간 매년 정례화 (글로벌 컨퍼런스, 블록체인 챌린지 등)

두 번째 분야로 기술경쟁력을 확보하기 위해 22년까지의 중장기 로드맵을 배포하였다. 또한 기술지원센터, 시험환경 구축, 평가체계 수립,

등을 통해 블록체인의 신뢰성 및 성능 평가를 전폭적으로 지원하고자 하였다. ITU, ISO, W3C 등 기구의 표준화 활동 지원을 통해 블록체인의 표준화를 추진하는 것 또한 포함되었다.[7]

세 번째 분야는 블록체인 산업을 활성화시킬 기반을 조성하기 위한 정책 지원으로 인력양성을 위한 연구센터 확대, 창업자대상 공모전 개최, 적기에 기술도입을 위한 규제 개선 연구반 운영 등을 실천할 예정이다.[7]

3.2 간편결제 분야 정책 동향

간편결제 분야에 있어서는 서울시청의 적극적인 개혁의지가 반영된 행보가 돋보인다. 서울시청은 18년 7월 "소상공인 수수료 부담 제로 결제 서비스 '서울페이'" 추진반을 신설하여 소상공인이 부담하였던 카드수수료를 줄이기 위한 방안으로 제시하였다. 결제판에 모바일 기기를 접촉하면 결제가 완료되는 QR 코드를 이용한 간편결제 방식이며, 한국은행과 표준 QR 코드를 개발하고 있다. 추진반은 서비스 기획 총괄과 가맹점 관리, 제도/법령 개선을 담당하는 총괄팀과, 서울페이 허브 구축, 표준 QR 개발, 상호운용가능한 시스템 구축 등 서울페이를 위한 기반 구축하는 구축팀, 사용자 확보 추진, 홍보를 담당하는 협력팀으로 구성되어있다.[8]

서울시청은 18년 8월에 시민청에서 인터넷전문은행의 성과를 논의하는 행사를 개최하였다. 각 부서 장관과 서울시장을 포함하여 국회의원, 금융소비자 대표, 금융 및 ICT 업계 등 관련 전문가 및 소비자가 참가하여 그간의 사업성과와 금융혁신을 위한 방향성에 대해 논의하였다. 논의중에서는 인터넷전문은행, 핀테크, 빅데이터 등 분야의 발전정책과 신용정보법, 인터넷전문은행법, 금융혁신지원 특별법 등 혁신 법안에 대한 입법 논의가 제시되었다.[9]

3.3 바이오인증 분야 정책 동향

바이오인증 분야에 있어서는 민간에서 제공되는 바이오인증 서비스의 안전성과 보안을 향상하기 위한 가이드라인, 안내서 등이 주로 배포되었다.

방송통신위원회는 '바이오정보보호 가이드라인 개정안'을 공표하여(2018.01) 바이오 정보에 대한 정보보호를 강화하였다.[10] 과학기술정보통신부는 17년 8월 정보보호시스템 평가인증 지침 고시를 제정하였으며[11] 이에 따라 IT보안인증사무국은 정보보호제품 평가인증 안내서를 작성하여 각 기업이 제정된 평가인증 지침 고시에 대한 이해를 높이고 안내서에 따라 고시를 지킬 수 있게끔 격려하였다. 이에 더해 IT보안인증사무국은 [표 3]과 같은 국가용 보호프로파일과 그에 대한 보조문서를 개발하여 사무국 웹사이트에 공개하고 있다.[12]

표 3 IT보안인증사무국 배포 국가용 보호프로파일 리스트[12]

국가용 보호프로파일 리스트
· 국가용 통합인증 보호프로파일 V1.0
· 국가용 문서 암호화 보호프로파일 V1.0
· 국가용 데이터베이스 암호화 보호프로파일 V1.0
· 국가용 무선침입방지시스템 보호프로파일 V1.0

IV.결론

핀테크 산업의 활성화와 보안 및 안전성을 촉구하기 위해 정부기관 및 지자체에서는 다양한 정책적 지원과 입법 논의를 진행하고 있다. 블록체인 분야는 R&D, 테스트 사업 지원을 추진하고 있으며 주로 제시되고 있는 활용사례 구현을 관심 여겨 보고 있으며, 과학기술정보통신부에서는 블록체인 개발도입 5개년 로드맵을 발표하여 구체적인 블록체인 기술개발의 청사진을 제시하였다. 간편결제 분야에서는 간편결제 활성화를 저해하는 법령의 개정을 검토하는 활동이 이루어지고 있고, 서울시에서는 이른바 '서울페이'라고 불리는 간편결제/송금 서비스를 지원하고자 전담팀을 구성하여 추진하고 있다. 바이오인증은 인체를 기반으로 만들어진 정보로 고유성, 불변성이라는 특성을 가진 민감정보인만큼 정책 추세는 바이오인증 보안에 주력하고 있다. 이러한 정책적 지원과 협력에 기반하여 핀테크 산업 활성화를 위한 연구와 기술개발을 촉구하는 바이다.

[참고문헌]

[1] 안상욱, "블록체인 속 핀테크를 보다 ① 블록체인이란?", 블로터, 2015.06.12., accessed 2018.09.17. http://www.bloter.net/archives/230157

[2] 사경진, "간편 결제 서비스의 보안 위협 분석 및 보안성 강화 연구"(순천향대학교 석사학위논문, 2018.02)

[3] 과학기술정보통신부, "핀테크 산업 활성화 기반조성", 2017.05.18.

[4] 정부24, "'공인인증서 제도' 20년 만에 전면 폐지 추진," 2018.03.29.

[5] 금융위원회, "핀테크로 금융혁신 이끈다 -「핀테크 혁신 활성화 방안」추진", 2018.03.19.

[6] 금융위원회, "「블록체인 협의회」출범 및 「금융권 공동 블록체인 컨소시엄」운영계획 발표"2016.11.24.

[7] 과학기술정보통신부, "블록체인 글로벌 기술경쟁력 확보, 선도사업 본격 추진", 2018.06.21.

[8] 서울특별시, "소상공인 수수료 부담제로 결제 서비스" 2018.09

[9] 금융위원회, "인터넷전문은행, 경쟁과 혁신을 통해 대한민국 금융을 바꿔 나가겠습니다."2018.08.07.

[10] 방송통신위원회, "방통위, 「바이오정보 보호 가이드라인」 발표"2017.12.12.

[11] 과학기술정보통신부, "정보보호시스템 평가·인증 지침" 2017.08.24. 시행

[12] IT보안인증사무국, 보호프로파일, accessed 2018.09.17. http://www.itscc.kr/pprof/kr/list.do?profile_class=1

미국 원자력시설의 사이버보안 규제 지침과 최신 개정안 비교

이채창*

*한국원자력통제기술원

Comparison of Regulatory Guide and Recent Revision Draft on Cyber Security for U.S. Nuclear Facilities

Chaechang Lee*

*Cyber Security Division, Korea Institute of Nuclear Non-proliferation and Control

요 약

미국은 2003년 원자력시설에 대한 설계기준위협(Design Basis Threat)에 사이버공격 특성을 추가함으로써, 원자력시설에 대한 사이버보안 규제를 시작하였다. 미국 NRC(Nuclear Regulatory Commission)는 10 CFR 73.54에 관한 규제 지침 RG 5.71을 2010년 최초 발표 후, 2018년 8월에 첫 번째 개정안을 발표하였다. 본 논문에서는 미국 NRC의 원자력시설 사이버보안 규제 지침과 개정안을 비교하고, 차이점에서 나타나는 사이버보안 규제 방향의 의미를 다루고자 한다.

I. 서론

미국은 2009년부터 연방법 10 CFR 73.54 Protection of Digital Computer and Communication Systems and Networks를 통해 원자력사업자가 사이버보안 의무 사항들을 이행하도록 규제하고 있다[1]. 미국의 규제기관인 NRC는 RG(Regulatory Guide) 5.71 Cyber Security Programs for Nuclear Facilities를 2010년 최초 발행하여, 원자력사업자가 연방법에 기술된 사이버보안 요건을 만족하기 위한 지침을 제시하였다[2]. 미국은 RG 5.71에 대한 개정안 DG-5061를 2018년 8월에 발행하였으며, 이에 대한 의견 수렴을 2018년 10월까지 접수하였다[3].

우리나라는 RG 5.71을 참고하여 KINAC/RS-015 기술기준이 개발되었다[4]. 국내 원자력사업자들은 이 규제 기준에 따라 정보시스템 보안규정을 마련하고 이행하고 있다.

본 논문에서는 미국의 원자력시설 사이버보안 규제 지침과 그 개정안을 비교하고, 특징적인 차이점을 도출한다. 그리고 미국의 원자력시설 사이버보안 규제 지침의 변화에 어떤 의미가 있는지 살펴본다.

II. 본론

2.1 NRC RG 5.71 [2]

미국의 원자력 사업자들은 NRC에 의해 규제를 받으며, 사이버보안에 관한 규제 지침은 RG 5.71을 따른다. NRC는 RG 5.71을 통해 연방법 10 CFR 73.54에 따른 사이버보안 규제 입장을 제시하고, 사업자가 사이버보안계획(Cyber Security Plan, CSP)을 마련하기 위한 가이드라인을 제공한다. RG 5.71은 사업자의 CSP가 갖추어야 할 필수적인 요소들을 템플릿을 통해 제시하고 있으며, 이행해야 하는 사이버보안 조치들을 나열하고 있다.

2.1.1 규제 지침

NRC의 가이드라인을 따라 작성되는 원자력 사업자의 CSP가 갖추어야 하는 첫 번째 사항은 사이버보안팀의 조직이다. 원자력사업자는 운영과 분리된 별도의 사이버보안팀을 조직하여야 하며, 시설의 사이버보안 활동을 관리하고 감독한다.

두 번째는 원자력시설의 사이버보안 규제 대상이 되는 필수디지털자산(Critical Digital Asset, CDA)을 식별하는 것이다. 필수디지털자산은 원자력시설에서 안전, 보안, 비상대응(Safety, Security, and Emergency Preparedness, SSEP) 기능을 수행하거나, 사이버 공격을 받을 시 SSEP 기능에 악영향을 줄 수 있는 디지털 시스템을 의미한다. RG 5.71에서는 CDA를 식별하는 방안을 제시하고 있다.

RG 5.71은 세 번째 CSP로 식별된 필수디지털자산의 보호를 위한 심층방호전략을 제시한다. 안전과 연관된 CDA, 안전에 중요한(important-to-safety) CDA, 보안 기능을 수행하는 CDA, 그리고 침해를 받을 경우 이와 같은 기능에 악영향을 미칠 수 있는 지원 기기 및 장치들을 가장 높은 등급에 배치하여, 높은 등급에서 낮은 등급으로 단방향 통신이 되도록 요구한다.

다음으로 RG 5.71은 원자력 사업자가 마련한 CSP에 대해 주기적이고 지속적인 감시와 평가를 할 것을 요구한다. 새로운 보안 위협을 식별하고 시스템의 취약점을 검사하여, 적용된 사이버보안 조치들의 효과성을 분석하고 자발적인 CSP 검토 및 개선을 수행할 것을 요구한다.

마지막으로 RG 5.71은 CSP를 통해 연방법에서 기록 유지하도록 요구하고 있는 기록물들을 정의하고 연방법에 따른 보존 기한을 다룬다.

2.1.2 사이버 보안조치

RG 5.71은 원자력 사업자들이 원자력 시설에서 식별된 CDA에 적용해야 할 보안 조치들을 미국 표준 NIST SP 800-53 등을 참조하여, 기술적, 운영적, 관리적 보안조치로 구분하여 제시하고 있다[5]. RG 5.71에서 제시하고 있는 사이버보안 조치들을 정리하면 표 1과 같다.

[표 1] RG 5.71 사이버 보안조치

분류	항목
기술적 보안조치	접근 제어
	감사 및 책임
	CDA 및 통신의 보호
	식별 및 인증
	시스템 보안 강화
운영적 보안조치	매체 통제
	인적 보안
	시스템 및 정보의 무결성
	유지 보수
	물리적 및 환경적 보호
	방어 전략
	심층방호
	사건 대응
	비상대응계획
	사이버보안 인식 및 훈련
	설정 관리
관리적 보안조치	시스템 및 서비스 도입
	보안 평가 및 위험 관리

2.2 NRC DG-5061 주요 변경 사항

미국은 2018년 8월 RG 5.71에 대한 개정 초안 DG-5061 Cyber Security Programs for Nuclear Power Reactors를 발행하였다. 이 개정의 목적은 NIST 800-53의 최신 개정사항을 반영하고, 기존의 RG 5.71의 적용 과정에서 도출된 이슈들을 명확히 하기 위한 것임을 밝히고 있다[3].

2.2.1 규제 지침 변경 사항

DG-5061의 특징적인 차이점은 먼저 문서의 제목에서 확인할 수 있다. RG 5.71은 원자력시설을 운영하는 모든 사업자에 대한 지침인 반면, DG-5061는 발전용 원자로 시설 운영 사업자에 대해 초점이 맞춰져 있다.

또한, 개정안에서는 핵 증기 관련 설비 계통인 1차 계통과 터빈 발전기 관련 설비 계통인 2차 계통을 보조하는 보조설비 계통(Balance of Plant, BOP) 시스템 중에서 원자로 반응도에 직·간접적으로 영향을 미치거나 원자로 정지 혹은 과도상태를 초래할 수 있는 디지털 시스템 및 장치들을 추가로 필수디지털자산으로 식별할 것을 요구하고 있다.

아울러, 사이버 사건 대응 과정에서 발생 및 발견된 사이버보안 사건이 연방법 10 CFR

73.77 Cyber Security Event Notification에서 제시하고 있는 시간 기준 내에 보고되어야 함을 상기시키고 있다.

또한 원자력사업자가 주기적으로 수행해야 하는 CSP 효과성에 대한 분석 시 활용할 수 있는 측정 지표를 옵션으로 제시하고 상세히 기술하고 있다.

사이버 보안조치의 이행에 관하여서는, 미국 발전용 원자로 사업자들의 협력 기관인 NEI(Nuclear Energy Institute)의 사이버보안 조치 이행 방안 NEI 13-10 Cyber Security Control Assessments를 참조하여, 사업자들이 공통된 특징을 가진 CDA를 서로 그룹으로 묶어 통합적인 공격 벡터 및 공격 트리 분석을 수행하고, 어떤 사이버보안 조치 또는 대안적인 보안조치를 그룹별로 적용할 것인지 판단할 수 있도록 하였다.

2.2.2 사이버 보안조치 변경 사항

RG 5.71의 사이버 보안조치 중 기술적 보안조치의 일부 항목들이 NIST 800-53의 개정사항을 반영하여 DG-5061 개정안에서 삭제되었다. 접근 제어 항목에서는 마지막 로그온 날짜 및 시간과 마지막으로 성공한 로그온 이후 실패한 로그온 시도 횟수를 표시하도록 하는 조치가 삭제되었다. 또한, 접근 제어의 시행 및 사용에 관한 사용자의 활동을 문서화하고 감독 및 검토하도록 하는 조치가 삭제되었으며, 저장, 처리, 전송되는 하드 카피 및 소프트 카피 정보들의 보안 속성을 레이블링할 것을 요하는 조치가 삭제되었다.

기술적 보안조치의 CDA 및 통신의 보호 항목에서는 프로세스의 우선순위에 따라 리소스 사용을 제한하도록 CDA를 구성하도록 한 보안조치가 삭제되고, 대신 서비스 거부 공격(Denial of Service, DoS)을 막기 위해 과다한 리소스 사용으로 인한 서비스 거부 공격을 탐지하고 충분한 리소스가 확보되어 있는지 모니터링 하도록 요구하고 있다. 또한, CDA 및 콘솔이 최소한의 기능과 데이터 저장이 가능한 씬 노드(thin node)로 구성하도록 한 보안 조치가 삭제되었다.

2.2.3 개정안 비교

앞서 서술된 규제 지침과 사이버 보안조치의 주요 변경 사항들을 정리하면 표 2와 같다.

[표 2] RG 5.71와 DG-5061의 비교

분류	RG 5.71[2]	DG-5061[3]
지침의 대상	원자력 시설	원자력 발전소
CDA 식별	SSEP 기능을 수행하거나, 사이버 공격을 받을 시 SSEP 기능에 악영향을 줄 수 있는 디지털 시스템을 CDA로 식별	BOP 시스템 중에서 원자로 반응도에 직·간접적으로 영향을 미치거나 원자로 정지 혹은 과도상태를 초래할 수 있는 디지털 시스템 및 장치들을 추가 식별
사건 대응	사이버 사건을 안전조치(safeguard)에 관한 연방법에 따라 보고	사이버 사건 보고에 관한 연방법의 요구사항에 따라 정해진 시간 내에 보고
CSP 효과성 분석	원자력 사업자가 자체적으로 CSP의 효과성을 분석	CSP 효과성 분석을 위한 측정 지표 제시
대안적 보안 조치	사이버 보안조치 이행 불가 시, 각 CDA에 대한 공격 벡터 및 취약점 분석 후 대안 조치 수행	사이버 보안조치 이행 불가 시, CDA를 특성별로 분류하여, 특성에 따른 분석 후 공통된 대안 조치 수행 가능
기술적 보안 조치	이전 로그온 통지	삭제
	접근 제어의 감독 및 검토	삭제
	자동 라벨링	삭제
	자원 우선순위	삭제(리소스 모니터링을 통한 DoS 방지)
	씬 노드	삭제

2.3 개정안의 의의

개정안은 기존의 지침이 모든 원자력시설을 동일 선상에서 규제를 한 것과 달리, 원자력시설의 특성에 따라 사이버보안 규제의 주안점이

달라질 수 있음을 보여준다. 원자력발전소는 핵물질 불법 이전과 사보타주에 대한 설계기준위협을 설정하고 이에 대한 방호체제를 수립해야 한다. 개정안은 사이버 공격으로 인해 원자력발전소에서 발생할 수 있는 악영향을 최소화하기 위해 CDA 식별을 강화하고, 사이버 사건을 보고를 강조하는 한편, 기존 지침을 따라 사이버 보안 조치들을 이행하는 과정에서 도출된 사업자들의 의견을 반영하여 대안적인 사이버보안 조치 적용 시 선행되어야 하는 CDA의 취약점 및 공격 벡터 분석 과정을 간소화하였다.

또한, 다른 사이버 보안조치로 대신할 수 있는 기술적 보안조치 항목들을 삭제하고, 발전소의 제어시스템 환경에 적용이 제한되는 항목들을 삭제함으로써 기술적 보안조치들을 갱신하였다. 그리고 보안 조치 항목들 마다 각 보안조치의 목적과 의도를 상술함으로써, 규제의 방향을 명확히 하고 있다.

III. 결론

본 논문에서는 미국의 원자력시설 사이버보안 규제 지침 RG 5.71과 개정안 DG-5061을 비교하고, 특징적인 차이점을 도출하였다. 우리나라는 미국의 사이버보안 규제 지침을 참조하여 국내 원자력시설에 대한 사이버 보안을 규제하고 있다. 향후 RG 5.71에 대한 공식 개정본 발행 시, 상세 검토 및 분석을 통하여 우리나라 사이버보안 규제 기준에 대한 개정 방향을 수립하는 것이 필요하다.

[참고문헌]

[1] U.S. NRC. Title 10 Code of Federal Regulations (CFR) Part 73. "Physical Protection of Plants and Materials". 2016.

[2] U.S. NRC. Regulatory Gudie 5.71. "Cyber Security Programs for Nuclear Facilities". 2010.

[3] U.S. NRC. Draft Regulatory Guide (DG)-5061, "Cyber Security Programs for Nuclear Power Reactors". 2018.

[4] KINAC. RS-015, "Regulatory Standard - Security for Computer and Information System of Nuclear Facilities", rev01. 2016.

[5] NIST SP 800-53, Rev. 3, "Recommended Security Controls for Federal Information Systems," National Institute of Standards and Technology, Gaithersburg, MD, August 2009.

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

구두발표 트랙 C

블록체인 기반 스마트계약을 활용한 개선된 인앱 결제시스템 모델 제안*

신정민*, 염흥열**

*순천향대학교 융합서비스보안학과,**순천향대학교 정보보호학과

A proposal for improved in-App purchase system using a blockchain-based smart contract

Jung-Min Shin*, Heung-Youl Youm**

*Department of Convergence Service Security, SoonChunHyang University.

**Department of Information Security Engineering, SoonChunHyang University.

요 약

모바일 앱이 스마트폰, 태블릿PC등 다양한 모바일 기기의 확대와 함께 급속하게 성장함에 따라 인앱 결제 서비스 비중 또한 증가 하고 있다. 하지만 현재 앱마켓 시장에서 구글플레이와 애플의 앱스토어의 점유율이 80%이상이고 인앱 결제 시스템을 이용할시 30%의 수수료를 지불해야한다. 또한 앱마켓의 환불정책을 악용한 블랙컨슈머의 등장으로 개발사의 부담이 증가하고 있는 추세이다. 따라서 본 논문에서는 이런 마켓 플랫폼 중간에 끼어있는 인앱 결제 시스템을 블록체인 기반 스마트 계약을 활용하여 앱 개발사와 개인 간의 인앱 결제시스템 모델을 제안한다.

I. 서론

스마트폰, 태블릿PC등 다양한 모바일 기기의 확대에 따라 모바일 앱(App) 시장이 급속하게 성장하고 있다. 현재 전세계 앱 시장에서 게임 앱, 데이트앱 음악 및 비디오 스트리밍 앱이 매출의 상위권으로 그중에 게임 앱이 매출 규모가 대폭 성장하고 있다. 이에 따라 기본 결제 시스템인 인앱 결제 서비스의 비중 또한 대폭 성장 하고 있다. 현재의 인앱 결제 시스템은 보통 앱 마켓 플랫폼이 끼어있는 중앙 집중형 시스템 기반으로 이루어지고 있다. 대표적인 예로 구글의 구글플레이와 애플의 앱스토어가 있다. 현재 앱마켓 시장에서 구글플레이와 애플의 앱스토어의 점유율이 80%이상이고 인앱 결제 시스템을 이용할시 30%의 수수료를 지불해야한다.[1] 또한 앱마켓의 환불정책을 악용하여 부당이득을 취하는 블랙컨슈머의 등장으로 개발사의 부담이 증가하고 있는 추세이다. 이러한 개발사의 부담은 앱 개발의 발전을 저해할 수 있고 게임내의 균형을 무너트릴 수 있다. 따라서 본 논문에서는 현재 새로운 기술 분야에서 화두가 되고 있는 블록체인(BlockChain) 기반 스마트 계약(Smart Contract)을 연구하고 이를 활용한 인앱 결제 시스템 모델을 제안한다.

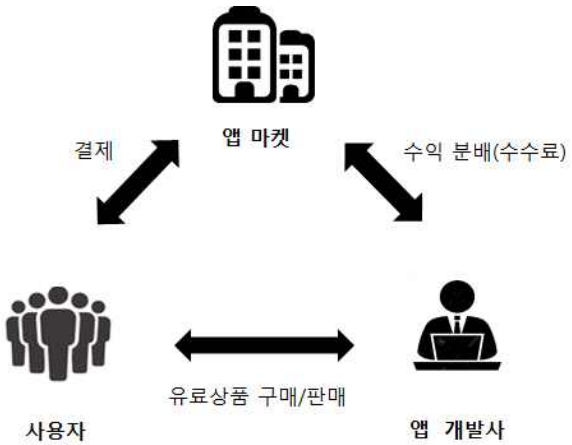
* 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "고용계약형 정보보호 석사과정 지원사업"의 연구결과로 수행되었음

II. 본론

2.1 인앱 결제시스템 현황

국내 모바일 환경에서 나타나는 결제방식으로 인앱 결제를 꼽을 수 있다. 현재 인앱 결제시스템은 앱 마켓과 앱 개발자 그리고 사용자의 관계로 이루어져있다. 사용자가 인앱 결제 시 개발사와의 직접적인 결제가 아니고 구글 플레이와 앱 스토어와 같은 마켓 플랫폼의 인앱 결제 시스템을 통하여 인앱 결제를 하게 된다. 이후

앱 마켓과 앱 개발사가 수익을 분배하는 구조로 되어있다. [그림 1]은 현재 인앱 결제 시스템의 구조는 다음과 같다.



[그림 1] 인앱 결제 시스템 구조

이러한 인앱 결제 시스템 구조에서 사용자가 인앱 유료상품을 구매시 앱마켓을 통하여 결제하게 되는데 결제 완료 후 환불을 할시 금액은 환불을 받게되지만 유료상품이 회수가 안되는 문제가 발생하게 된다. 구글 플레이 스토어의 경우 매한 상품에 대해 구매 이후 65일 이내에 환불을 진행해주는 '결제 65일 이내 미승인 구매 환불' 정책을 지키고 있고 애플 앱스토어의 경우 유료 상품 구매 이후 2주 안에 아이튠즈를 통해 적당한 카테고리 선택과 사유를 적으면 곧바로 환불을 받을 수 있다. 환불 정책을 통해 진행된 환불의 경우에는 앱 개발사 측에 환불을 요청한 유저의 환불정보를 해당 개발사와 공유하지 않는다. 개발사는 부당한 환불인지 확인할 수도 없고 유저가 사용한 유료 상품을 회수할 수도 없는 상황에서 환불금 까지 내줘야하는 삼중고에 시달리게 된다.[2]

2.2 블록체인 개요

블록체인은 탈 중앙화 형태의 분산 데이터베이스 기술이다. 각 시스템 테이터베이스의 데이터를 네트워크로 공유함으로써, 분산된 환경에서도 실시간 공유가 가능하며 합의 알고리즘의 의해 일관성있는 데이터 유지가 가능하다.[3] 분산원장 기술이라고도 불리는 블록체인은 다수의 불록으로 구성되는데 이때 블록(Block)을 잇따라 연결(Chain)한 모음으로 블록에 일정 시간 동안 확정된 거래내역을 담는 일종의 금융장부이다. 블록체인 네트워크의 참여자(노드)들은 중계자나 제 3자 없이 직접적인 거래를 할수 있으며 전달되는 블록을 검증한 후 승인된 블록만 추가하는 형태이기 때문에 신뢰성이 보장된다.[4] 비트코인에서부터 시작된 블록체인 기술은 최근 핀테크 기술과 융합되어 다양한 분야에 도입 중이다. 블록체인의 주요 특징으로는 다음 [표 1]과 같다.

[표 1] 블록체인의 특징

분산성	공인된 제3자의 공증 없이 개인간 거래 가능 ->불필요한 수수료 절감
보안성	정보를 다수가 공동으로 소유하여 해킹 불가능 ->보안관련 비용 절감
신속성	거래 승인·기록이 다수의 참여자에 의해 자동실행 ->신속성 근대화
확장성	공개된 소스에 의해 쉽게 구축·연결·확장 가능 ->IT 구축비용 절감
투명성	모든 참여자들이 거래에 공개적 접근가능 ->거래 양성화 및 규제 비용 절감

2.3 블록체인의 종류

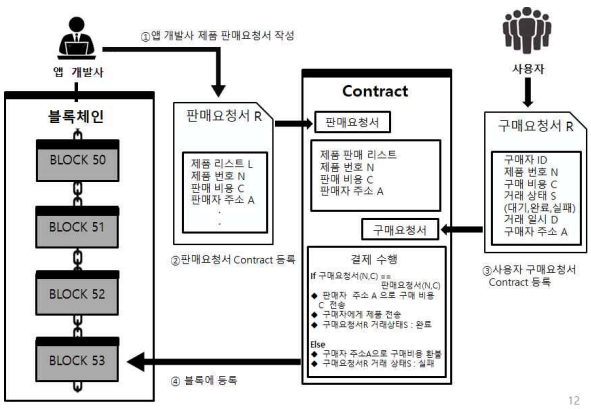
블록체인 네트워크의 특성에 따라 퍼블릭 (Public), 프라이빗 (Private), 컨소시엄 (Consortium) 으로 종류를 구분할 수 있다. 퍼블릭 블록체인 네트워크는 참여자에 대해 제한이 없어 트랜잭션을 생성하고 검증하고 승인하는 것이 모두 허용된다. 네트워크에 참여한 모든 노드는 같은 권한을 가지게 되지만, 거래 승인의 참여자가 참가하여 속도가 느려진다는 단점이 있다. 반면에 프라이빗 블록체인 네트워크는 주체가 되는 하나의 기관이 있으며 허가된 사용가만 참여할수 있는 특징이 있다. 컨소시엄 블록체인은 하나가 아닌 허가된 여러 개의 기관이 참여하는 네트워크 형태로 반 중앙형의 성격을 띠고 있고, 프라이빗 블록체인의 특성을 띠고 있다.[4]

2.4 스마트 계약 개요

스마트계약(Smart Contracts)는 중개자없이 P2P로 쉽고 편리하게 계약을 체결하고 수정할 수 있는 기술이다. 이 기술은 블록체인(Blockchain)의 분산원장기술(DLT, Distributed Ledger Technology)이라는 특성을 이용해 금융거래, 부동산 계약, 공증 등 다양한 형태의 계약을 이해 당사자 간의 공유 네트워크를 통해 계약과 계약의 결과에 대한 신뢰를 쌓아 나갈 수 있는 기반 하에 확보된 자동화된 계약처리의 형태를 갖고 있다. [5] 이 기술은 이더리움 블록체인을 기반으로 해 비트코인보다 한 단계 발전했다고 해서 블록체인 2.0이라고도 부른다. 블록체인 기술을 이용함으로서, 계약상의 조건이나 사건이 발생하였는지 여부가 제3자의 개입 없이 확인되고, 그에 따라 계약의 이행이 인간의 개입 없이 자동적으로 이루어지는 스마트 계약이 가능하게 되었다고 평가된다.[6]

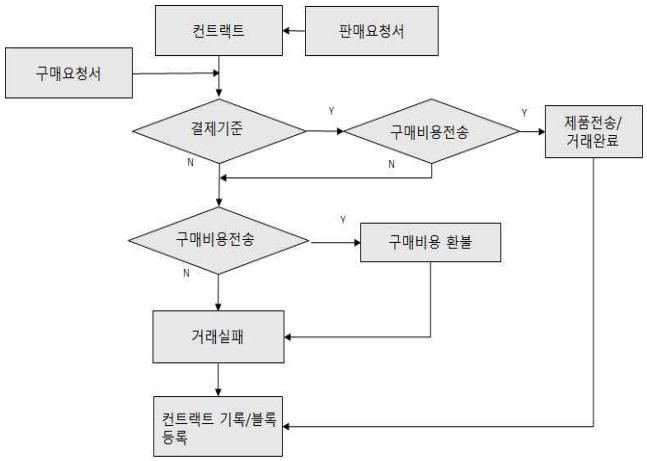
Ⅲ. 블록체인 기반 스마트계약을 활용한 인앱결제 시스템 모델 제안

본 논문에서 제안하는 블록체인 기반 스마트 계약을 활용한 인앱 결제시스템 모델을 사용하기 위해 기존의 구글 플레이나 앱 스토어와 같은 앱 마켓 플렛폼을 통하여 앱을 구매 하는 방식이 아닌 오픈 플랫폼 형태로 블록체인을 통해 안드로이드 apk 앱 파일을 배포한다.



[그림 2] 컨트랙트 구현

[그림 2]는 본 논문에서 제안하는 블록체인 기반 스마트 계약을 적용한 인앱 결제 서비스 모델이다. 앱 개발사는 판매할 유료상품의 요청서를 컨트랙트 코드로 작성하여 컨트랙트에 전송하고 앱 사용자는 컨트랙트로 부터 판매 요청서를 콜하여 구매할 제품리스트를 확인 후 구매 요청서를 작성하여 컨트랙트에 전송하고 컨트랙트 코드에 의해 구매 요청서는 컨트랙스에 기록된다. 이후 기록된 컨트랙트는 블록에 등록된다. [그림2]와 [표 2]은 결제과정 중 컨트랙트 상 결제 수행 흐름도와 수행 프로세스를 나타낸 것이다. [7]



[그림 2] 결제 수행 흐름도

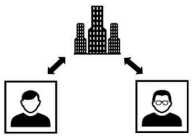
[표 2] 컨트랙트 결제 수행 프로세스[7]

결제 수행 프로세스
1. 앱 사용자가 보낸 구매요청서와 개발사에서 제품을 판매하기 위한 판매 요청서를 컨트랙트에 전송한다.
2. 구매 요청서와 결제 기준을 비교 한다.
3. 구매 요청서와 결제기준이 일치하면 다음 작업을 수행한다. - 판매자 주소A으로 구매비용 전송 - 구매자에게 제품 전송 - 구매요청서R 거래상태 : 완료
4. 구매 요청서와 결제 기준이 일치하지 않으면 다음 작업을 수행한다. - 구매자 주소A으로 구매비용 환불 - 구매 요청서R 거래상태 : 실패

3.1. 제안하는 블록체인 모델의 기대효과

기존의 인앱 결제 시스템에서 구글 플레이어와 앱스토어 같은 마켓 플랫폼이 중간에 끼어있는 결제 시스템을 이용함에 결제 기록과 환불정보를 공유하지 않아 부당한 환불인지 확인을 못하고 유료상품을 회수 하지 못한 채 환불금만 내줘야 했던 앱 개발사의 부담을 본 논문이 제안하는 블록체인 모델을 사용함으로써 거래기록이 블록에 남아 수월하게 해결할 수 있을 것이다. 또한 분산성, 보안성, 투명성 과 같은 블록체인의 특징을 통해 인앱 결제 우회와 게임 변수 조작, 불법복제 등 보안상의 문제를 해결 하고 거래과정에서의 투명성과 신뢰성이 향상된 블록체인 기반 인앱 결제시스템 모델이 될수 있을 것이다. 다음 [표 3]은 본 논문에서 제안하는 블록체인 기반 스마트계약을 활용한 인앱 결제 시스템 모델과 기존의 마켓 플랫폼이 끼어있는 중앙 집중형 결제 시스템을 비교하였다.

[표 3] 기존 결제방식과 제안 모델 비교

구분	기존 결제 시스템	제안 모델
구조		
	결제 과정에 마켓 플랫폼이 끼어있는 중앙 집중형 시스템	사용자와 개발자가 직접 결제를 하는 분산형 결제 시스템
특징	빠른 거래 속도가 장점이지만 마켓플랫폼의 수수료 비용과 환불정책으로 인한 게임 내 불균형을 야기한다.	거래정보의 투명성, 적은 시스템 구축 및 개발사 측에서 불필요한 비용 절감이 하고 안전한 결제가 가능하다

Ⅳ. 결론

본 논문에서는 현재 마켓 플랫폼이 끼어있는 인앱 결제 시스템의 문제점을 살펴보고, 최근 급속히 성장하고 있는 블록체인 기반 스마트계약을 이용한 인앱 결제 시스템 모델을 제안하였다. 이로써 사용자 측면에서는 기존의 앱 마켓 플랫폼을 통한 인앱 결제 시스템보다 더욱 안전하고 신뢰성 있는 결제가 가능하다. 또한 블록에 기록이 남아 환불을 악용하여 부당이득을 챙기는 일이 없어지면서 게임내의 균형이 유지 될 수 있다. 앱 개발사 측면에서는 앱 마켓 플랫폼을 통하여 인앱 결제를 하면서 드는 수수료와 같은 비용을 절감 효과를 누릴수 있을 것이다.

본 논문의 모델이 참고 자료가 되어 블록체인 기반 스마트계약을 활용한 인앱 결제 시스템이 보다 안전하고 편리한 결제 시스템이 될 수 있도록 도움이 되고자 한다.

[참고문헌]

[1] 한찬희, “Mediaus”, http://www.mediaus.co.kr/news/articleView.html?idxno=49802, 2015.

[2] 이장혁, “GAMEPPLE”, http://www.gameple.co.kr/news/articleView.html?idxno=138189, 2017.

[3] Crosby, M., Pattanayak, P., Verma S., and Kalyanaraman V.. Blockchain technology : Beyond bitcoin. Applied Innovation, 2, 6-10. 2016

[4] 김성은, 블록체인 기술을 이용한 혈액 콜드체인 시스템 설계 및 구현, pp. 8-9 , 2018

[5] 김제완, “블록체인 기술의 계약법 적용상의 쟁점”-‘스마트계약(Smart Contract)′을 중심으로, KCI, vol 727, No1, page 164, 2018.

[6] 위키백과 스마트계약 https://ko.wikipedia.org/wiki/스마트계약 2017

[7] 김광훈, “이더리움을 이용한 파일 거래 모델 File Exchange Model Using Ethereum”, 한국통신학회, page2, 2017.

Solidity 기반 스마트 컨트랙트 시스템 퍼징을 위한 자동 코드 생성 기법 연구†

김윤태* 이종협*

*가천대학교 금융수학과

Automated code generation for fuzzing Solidity-based smart contract systems

YunTae Kim* JongHyup Lee*

*Department of Mathematical Finance, Gachon University.

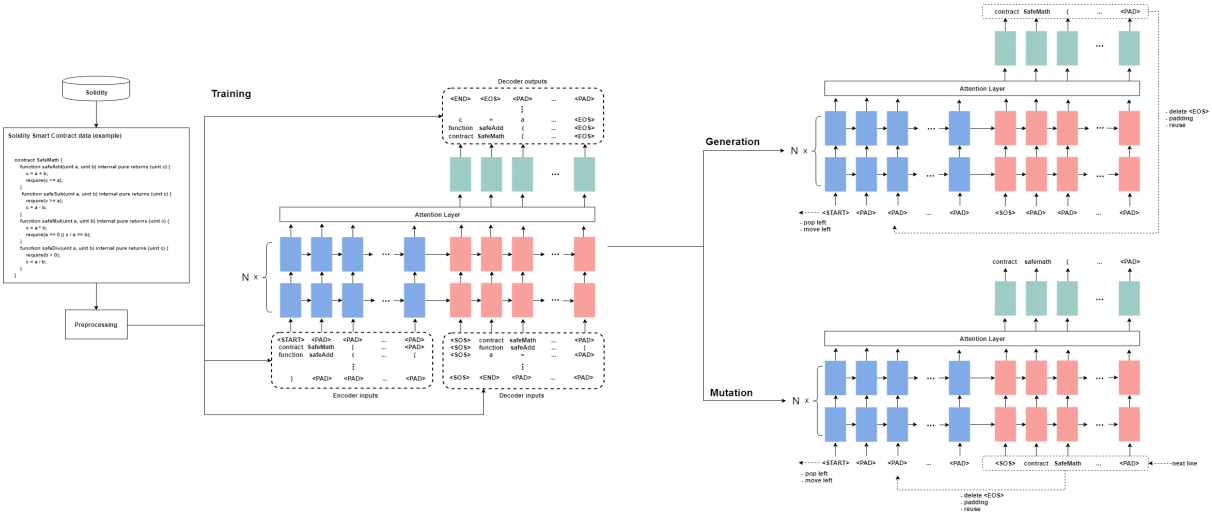
요 약

스마트 컨트랙트란, 블록체인 상에서 동작하는 프로그램 코드로써 블록체인의 서비스를 구성하는 필수 요소이다. 스마트 컨트랙트는 암호화폐를 이용한 금융을 위시한 다양한 환경에서 사용되고 있기 때문에 관련 시스템의 문제점이 블록체인의 서비스 자체의 문제로 연결될 수 있어서 조심스러운 접근이 필요하다. 하지만 스마트 컨트랙트를 위해 대표적으로 사용되고 있는 Solidity언어가 개발 초기 단계에 머물러 있어, 제한적인 테스트 코드만을 생산하고 있다. 따라서 본 논문에서는 스마트 컨트랙트 시스템에 대한 효과적인 검증을 위한 Solidity 기반 퍼징 기법을 제안한다. 제안하는 S2S Fuzzing(Sequence to Sequence Fuzzing) 기법은 딥러닝을 이용하여 Solidity 소스코드를 자동으로 생성한다. S2S Fuzzing은 LSTM 모델을 기반으로 하는 Seq2Seq(Sequence to Sequence Learning)모델로, 입력 방법에 따라 제너레이션 퍼징과 뮤테이션 퍼징을 구현할 수 있다는 특징이 있다.

I. 서론

현 시점에서 블록체인(Blockchain)은 분산성, 보안성을 앞세워 금융, 의료 등 다양한 분야에서 활용되고자 한다. 2세대 블록체인으로 대표되는 이더리움(Ethereum)부터 도입된 스마트 컨트랙트(Smart Contract)는 블록체인의 활용성을 보다 넓혀나간다. 스마트 컨트랙트란, 블록체인 상에서 작동하는 프로그램으로 특정 계약 내용을 자동화하여 이행한다. 스마트 컨트랙트가 암호화폐(Cryptocurrency)에 대한 처리도 가능한 만큼, 취약점이 발생할 경우 막대한 금전적 문제를 야기할 수 있다. 이러한 이유로 다양한 스마트 컨트랙트 분석 시스템들이 등장하고 있다. 하지만, 스마트 컨트랙트는 아직 생소한 개발환경이기 때문에, 실질적으로 다양하게 스마트 컨트랙트를 분석하기 위한 데이터 셋이 없는 실정이다. 최근 스마트 컨트랙트를 기반한 연구에서는 이더리움 블록체인에 적용되어 있는 스마트 컨트랙트 코드들을 이용하고 있지만, 많은 코드들이 중복되어 나타나고 있어 코드의 수에 비하여 다양성이 제한적이라는 문제를 가지고 있다. 본 논문에서는 이러한 문제를 해결하고자 다양한 형태의 Solidity 코드를 자동으로 생성하는 기법을 제안한다. 특히 제안하는 기법은 Solidity를 사용하는 다양한 시스템들에 대한 퍼징(Fuzzing)의 용도로 적용될 수 있다. 퍼징 데이터의 처리에 따라 제너레이션 퍼징(Generation Fuzzing)과 뮤테이션 퍼징(Mutation Fuzzing)으로 구분할 수 있다. 우리는 입력 방법에 따라 두 가지의 퍼징 전략을 딥러닝(Deep Learning) 모델을 이용해 재현하여, Solidity 코드를 생성하는 것으로, Solidity뿐만 아니라 Solidity의 정적분석 도구(*e.g.* Oyente, Mythril)를 퍼징할 있도록 한다. 본 논문의 구성은 다음과 같다. 2장에서는 제안하는 기법에 사용되는 딥러닝 알고리즘

† 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임. (2018-0-00251, 스마트 컨트랙트 프라이버시 보호 및 취약점 분석 기술개발)



[그림 1] Generation and Mutation fuzzing process with Seq2Seq model

에 대해 설명하며, 3장에서는 코드 생성 모델인 S2S Fuzzing에 대해 기술한다. 4장에서는 실험 과정과 그 결과에 대해 다루며, 5장에서는 관련 연구를, 6장에서는 결론에 대해 다룬다.

II. 배경

본 논문에서 제안하는 모델은 Seq2Seq(Sequence to Sequence Learning)를 기반으로 소스코드를 학습하여 코드를 변형하거나 새로운 코드를 생성한다. Seq2Seq모델은 RNN(Recurrent Neural Networks)과 같은 뉴럴 네트워크를 기반으로 구축된 인코더(Encoder)와 디코더(Decoder)로 구성되어 있다. 여기서 RNN은 LSTM(Long-Short Term Memory), GRU(Gated Recurrent Units)로 바꿔 적용할 수 있다[1].

Seq2Seq모델의 성능을 향상시킬 수 있는 기법 중 대표적으로 어텐션 매커니즘(Attention Mechanism)[2]이 있다. 이 기법은 Seq2Seq 모델에 처음 도입된 기법으로, 디코더에서 출력값을 산출할 때, 인코더의 전체 출력 값이 아닌, 일부 중요한 출력 값의 가중치를 높여 집중함으로써 보다 정확한 예측을 할 수 있도록 한다.

III. Sequence to Sequence Fuzzing

본 논문에서는 Seq2Seq 알고리즘을 기반으로 퍼징 기법의 뮤테이션 퍼징과 제너레이션 퍼징을 구현하여, 퍼징을 위한 코드를 생성하는 S2S Fuzzing(Sequence to Sequence Fuzzing) 기법을 제안한다. 두 가지의 코드 생성 스타일은 모델의 입력 값의 형태에 따라 구분되며, mutation의 경우, 변형될 코드 데이터를 필요로 한다.

3.1 데이터 전처리

스마트 컨트랙트의 소스코드는 Solidity의 스타일 가이드[3] 가 제공되고 있지만, 가이드를 따르지 않는 스마트 컨트랙트가 다수 존재하기 때문에, 수집된 데이터를 직접 적용하기에는 어려움이 있다. 동작하는 코드에 집중할 수 있도록 불필요한 주석을 모두 제거하고, 들여쓰기, 공백 등을 제거함으로써, 모델이 명령어와 연산자 등의 패턴을 학습할 수 있도록 한다. 들여쓰기의 경우, 차후 생성 과정에서 정규표현식을 사용하여 별도로 추가하게 된다. 또한, Solidity의 연산자(*e.g.* '+', '-', '&&', '==')를 구분할 수 있도록 처리한다. 코드 생성에 사용되는 단어는 DeepSmith[4]를 참고하여, 빈도를 기준으로 상위에 해당하는 단어를 선정하고 이외의 단어는 문자 단위로 변환하여, 모델이 학습해야하는 패턴의 복잡도를 낮출 수 있게 한다.

처리된 소스코드 데이터는 seq2seq모델의 인코더와 디코더에 일정한 길이로 입력될 수 있도록, 길이가 부족한 코드 줄 데이터에 '<PAD>'를 채워, 길이를 일정하게 변환한다. 변환된 코드 줄 데이터는 앞선 시점의 N개의 코드 줄을 병합한 인코더의 입력 값

[표 1] Example of source code generation and mutation

Generation	Mutation

contract StandardToken is ERC20 {
mapping (address => uint) balances ;
mapping (address => mapping (address => uint256)) allowed ;
function approve (address _spender , uint256 _value) public returns (bool) {
balances [msg . sender] = safeSub (balances [msg . sender] , _value) ;
balances [_to] = safeAdd (balances [_to] , _value) ;
Transfer (msg . sender , _to , _value) ;
	function burnFrom (address _from , uint256 _value) public returns (bool success) {
	require (balanceOf [_from] >= _value) ;
	require (_value <= allowance [_from] [msg . sender]) ;
	balanceOf [_from] -= _value ;
	allowance [_from] [msg . sender] -= _value ;
	totalSupply -= _value ;
	Burn Burn (_from , _value) ;
	return true ;
	}

$(S_{t-N+1}, S_{t-N+2}, \cdots, S_t)$, '<SOS>'를 맨 앞에 추가하여 디코더의 입력 값 S_{t+1}^{sos}를, '<EOS>'를 맨 뒤에 추가한 S_{t+1}^{eos} 디코더의 출력 값으로써 학습에 사용된다.

3.2 제안 기법

[그림 1]은 S2S Fuzzing 모델의 학습 방법과 학습된 모델을 이용해 제너레이션 퍼징과 뮤테이션 퍼징을 하는 전반적인 프로세스를 나타낸다. Solidity 소스코드는 전처리 과정을 통해 인코더의 입력 값, 디코더의 입력, 출력 값으로 데이터를 변환하고, 각각 모델 학습과정에서 사용된다. 모델의 활용 방법은 다음과 같다.

제너레이션 퍼징. 새로운 소스코드를 생성하고자 할 경우, 학습할 때 '<START>'와 '<END>' 신호를 학습하여, 어떠한 사전 정보 없이, 다음의 코드 줄을 생성하고, 생성된 코드를 다시 입력 값으로 사용하는 단계별 생성으로, 최종 적으로 '<END>' 신호가 생성되는 시점까지 소스코드를 생성하게 된다.

뮤테이션 퍼징. 기존의 소스코드를 변형하고자 한다면, 사전에 준비한 소스 코드를 입력 순서대로 인코더의 입력 값과 디코더의 입력 값으로 사용하는 것으로, 디코더에 실제 소스코드 정보를 제공함으로써, 입력된 소스코드를 모델에서 학습된 패턴에 적절하게 변형하는 형태로 반환하게 된다.

IV. 실험 및 결과

4.1 데이터 셋

본 연구에서 사용하는 소스코드 데이터는 etherscan.io[5]에서 수집한 검증된 스마트 컨트랙트 데이터 43,116개를 사용했다. 수집된 소스코드는 3.1절에서 언급된 과정을 통해 소스코드 내의 불필요한 정보를 제거하고, 마스킹(masking)되었다. 이후, 소스코드를 줄 단위로 구분하여 각각 인코더의 입력 값, 디코더의 입력 값, 디코더의 출력 값으로 변환하여 생성한 데이터 셋 약 398만개를 학습에 사용했다. 각각의 시퀀스(sequence) 데이터는 최대 길이를 30개로 제한하였으며, 빈도를 기반으로 전체 134,323개의 단어 중, 상위 약 0.1%에 해당하는 1,352개의 토큰을 사용하고 이외의 토큰은 문자 단위로 변형하였다.

4.2 설정

본 연구에서 실험한 Seq2Seq 모델은 인코더에서 고려할 사전 코드 줄의 수 N = 5로 설정하여 150의 길이를 가지는 시퀀스를 인코더에 입력하였다. 모델 구조는 100의 임베딩 크기와 인코더와 디코더가 각각 256개의 유닛(units)을 가지며, 2개의 은닉층 모델을 사용하였다. 모델의 학습을 위해 0.0001의 학습율(learning rate), 데이터 크기를 고려해 15의 반복 횟수(epoch), 256의 배치 크기를 사용해 학습했다. 학습 과정에서 정규화(regularization)을 위해, 0.95의 지수적 감쇠(exponential decay)를 적용하였다. 생성 결과에 대한 성능 평가는 [4]에서 평가 지표로 사용한 BLEU(Bilingual Evaluation Understudy)[6] 점수를 사용하여 1-gram을 대상으로 모델의 성능을 측정했다.

4.4 실험 결과

모델 학습 결과, 크로스 엔트로피 손실 값은 5.0431, BLEU 점수는 최대 0.2114까지 상승한 결과를 확인할 수 있었다. [표 1]은 학습된

모델을 이용해 제너레이션 퍼징, 뮤테이션 퍼징을 위한 코드 생성 결과의 일부이다. 제너레이션 퍼징은 스마트 컨트랙트의 대부분을 차지하는 토큰 발행 목적의 컨트랙트를 생성하였다. 사전정보가 주어지지 않아 동일한 목적의 코드를 생성하지만, 컨트랙트 간의 상속 및 함수의 선언 등 세부적인 내용에 대해서도 높은 완성도를 보인다. 반면에 뮤테이션 코드는 사전정보를 제공받아 다양한 목적의 컨트랙트가 생성되는 것을 확인할 수 있다. 함수 단위에서 높은 완성도를 보이지만, 컨트랙트 단위에서는 개선이 필요할 것으로 판단된다.

V. 관련 연구

최근 연구 중, 대상 컴파일러는 다르지만 본 논문과 유사한 연구가 진행된 바가 있다. 해당 연구에서는 딥러닝을 이용해 OpenCL을 대상으로 하는 퍼징 기법인 DeepSmith를 제안한다[4]. 해당 논문은 실제 소스 코드를 입력받아, 코드 정보를 재표현하는 인코더를 거쳐, LSTM 기반 생성모델을 사용해 여러 코드 표본을 생성했다. 소스 코드 데이터를 인코더에서 다루기 위해, 프로그래밍 언어의 예약어, 주로 쓰이는 이름 등은 토큰(token)단위로, 이외의 단어는 문자 단위로 처리하여 단어의 개수가 불필요하게 증가하는 것을 예방했다.

퍼징을 위해 소스코드를 생성하는 모델은, 딥러닝에서 문서 생성(text generation) 분야의 연구를 응용할 수 있다. 해당 연구에는 최근 이미지 생성 분야에서 우수한 성능을 보이는 GAN(Generative Adversarial Networks)를 텍스트 데이터에 적용한 seqGAN[7], 사전 정보를 이용해 텍스트를 생성하는 CSGAN[8], CSGAN을 개선한 stepGAN[9] 등이 있다.

본 논문에서 제안하는 S2S Fuzzing은 코드 생성을 코드 줄 단위로 생성하여 모델이 불필요하게 많은 파라미터를 가지지 않도록 하였으며, Seq2Seq 모델을 사용해, 제너레이션, 뮤테이션 퍼징의 역할을 수행할 수 있도록 하는 것에 의의가 있다.

VI. 결론

본 논문에서는 다양한 Solidity 코드를 딥러닝에 기반하여 자동으로 생성하는 모델인 S2S Fuzzing을 제안한다. 해당 모델은 입력 값의 형태에 따라 제너레이션과 뮤테이션 코드를 생성하여, Solidity를 기반으로 하는 시스템에 대한 퍼징의 용도로 사용될 수 있다. S2S Fuzzing은 Seq2Seq Learning모델을 사용하여 구현되었으며, 사전 정보를 재 표현하는 인코더에 이전 시점의 N개의 코드 데이터를 입력값으로 사용하여 코드를 생성한다. 본 모델은 토큰 발행 목적의 스마트 컨트랙트에 한해서, 완성도 높은 코드를 생성하였다. 차후 연구에서는 보다 다양한 목적의 컨트랙트를 랜덤하게 생성할 수 있도록, seqGAN, CSGAN 등의 적용을 고려하며, 나아가 특정 목적의 컨트랙트에 대해 다수의 코드를 생성할 수 있는 모델을 구축하고자 한다.

[참고문헌]

[1] L.Sutskever, O.Vinyals, and Q.V.Le, "Sequence to sequence learning with neural networks," Advances in neural information processing systems, pp.3104-3112, 2014.

[2] D Bahdanau, K Cho, and Y Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[3] https://solidity.readthedocs.io/en/v0.4.24/

[4] C.Cummins, P.Petoumenos, and A.Murray, "Compiler fuzzing through deep learning," Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, p. 95-105, 2018

[5] https://etherscan.io/

[6] K.Papineni, S.Roukos, T.Ward, and W.J. Zhu, "BLEU: a method for automatic evaluation of machine translation," Proceedings of the 40th annual meeting on association for computational linguistics, p.311-318, 2002.

[7] L Yu, W Zhang, J Wang, and Y Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, p2852-2858, 2017.

[8] Z.Yang, W.Chen, F.Wang, and B.Xu, "Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets," arXiv preprint arXiv:1703.04887, 2017.

[9] Y.L.Tuan and H.Y.Lee, "Improving Conditional Sequence Generative Adversarial Networks by Stepwise Evaluation," arXiv preprint arXiv:1808.05599, 2018.

IoT 블록체인을 이용한 토큰 기반 권한 관리

박 환*, 김미선*, 서재현*
*목포대학교 정보보호학과

Token-based Rights Management Using IoT Blockchain

Hwan Park*, Mi-sun Kim*, Jae-hyun Seo*
*Division of information security, Mokpo National University.

요 약

최근 IoT 환경에서 인증, 접근 제어 등을 위해 토큰을 이용한 시스템 연구들이 국내외에서 진행되고 있다. 하지만 기존의 토큰을 이용한 방식의 시스템은 중앙 집중적인 특성을 가지고 있어 확장성이 특징인 IoT 환경에서는 적합하지 않다. 본 논문에서는 중앙 서버를 대신하기 위해 IoT 블록체인 네트워크를 구성하고 토큰 정보를 공유하여 권한 관리를 수행하고자 한다. 블록체인을 이용하여 IoT 장치 간에 토큰 정보를 공유함으로서 토큰에 대한 무결성을 보장하고 모든 IoT 장치들이 토큰을 신뢰할 수 있도록 하고자 한다. 토큰에 대한 신뢰성이 확보되기 때문에 토큰의 소유자(Owner)는 토큰 재발급 과정을 수행하지 않고 기존 발급 받은 토큰을 이용하여 리소스에 대해 지속적으로 접근할 수 있다.

I. 서론

IoT 장치는 이기종의 네트워크들이나 표준이 적용되어지므로 다양한 보안 위협에 노출되어있다. 최근에는 IoT 환경에서 인증, 접근 제어 등을 위해 토큰을 이용한 시스템 연구들이 국내외에서 진행되고 있다. 하지만 기존의 토큰을 이용한 방식의 시스템은 중앙 서버가 토큰을 관리하는 중앙 집중적인 특성을 가지고 있다. 중앙 집중적인 시스템은 중앙 서버의 보안이 취약할 경우 네트워크 전체의 보안에 영향을 미칠 수 있으며 투명성을 보장하지 않아 중앙 서버에 대한 신뢰성이 확보되지 않을 경우 시스템 운영 상 어려움이 생기는 단일 지점 장애(Single Point of Failure) 문제점을 가지고 있어 확장성이 특징인 IoT 환경에는 적합하지 않다.

본 논문에서는 중앙 서버를 대신하기 위해 IoT 블록체인 네트워크를 구성하고 토큰 정보를 공유하여 권한 관리를 수행하고자 한다. 블록체인을 이용하여 IoT 장치 간에 토큰 정보를 공유함으로서 토큰에 대한 무결성을 보장하고 모든 IoT 장치들이 토큰을 신뢰할 수 있도록 하고자 한다. 토큰에 대한 신뢰성이 확보되기 때문에 토큰의 소유자(Owner)는 토큰 재발급 과정을 수행하지 않고 기존 발급 받은 토큰을 이용하여 객체(Object)에 대해 지속적으로 접근할 수 있다.

II. 관련 연구

2.1 블록체인을 이용한 IoT 인증

[1]에서는 블록체인을 활용하여 사물인터넷 환경에서 이기종 기기간의 상호 인증 프로토콜을 설계하였으며, [2]에서는 블록체인 기반 사물인터넷 인증 연구를 진행하였다[1, 2]. 본 논문에서는 블록체인의 특성을 이용하여 토큰을 발급한 사실과 토큰 정보를 공유함으로서 토큰에 대한 신뢰성을 확보하고 접근 권한에 대한 검

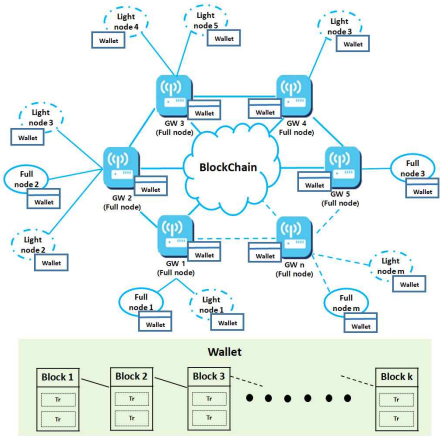
증을 필요로 한다.

2.2 IoT에서 Capability 토큰 기반 접근 제어 시스템

[3]에서는 IoT 환경에서 Capability 토큰을 생성하여 접근 제어를 수행하는 시스템을 제안하였다. CL 접근 제어 방식을 기반으로 하였으며, 주체의 권한을 Capability 토큰으로 정의하고 객체에게 토큰을 전달하여 접근 여부를 결정한다. IoT 환경과 같이 접근 요청이 빈번히 발생할 수 있는 환경에서 토큰의 무결성만을 확인하고 접근을 허용함으로서 접근 제어 처리 시간을 감소 시킬 수 있다[3].

III. IoT 블록체인을 이용한 토큰 기반 권한 관리

본 논문에서는 중앙 집중적인 시스템의 문제점을 해결하기 위해 IoT 블록체인을 이용하여 토큰을 관리함으로서 접근 제어를 수행하는 시스템을 제안하고자 한다. 본 논문에서 구성한 IoT 블록체인 네트워크는 (Fig 1)과 같다.



(Fig 1) IoT blockchain network

IoT 블록체인 네트워크는 게이트웨이, 풀 노그(full node) 와 라이트 노드(light node)로 구성하였다. 게이트웨이는 각 노드들과 연결하여 도메인을 형성하고 관리한다. 각 노드들은 블록 저장소인 Wallet를 가지고 있으며 블록들을 체인 형태로 저장한다.

3.1 IoT 블록체인 네트워크 구성원

본 3.1장에서는 IoT 블록체인 네트워크의 구성원들에 대하여 정의한다.

3.1.1 풀 노드(full node)

풀 노드는 블록을 생성 및 합의할 수 있으며, 전체 네트워크에서 생성되는 블록을 Wallet에 저장한다. 게이트웨이 또한 풀 노드의 특성을 가지고 있으며, 추가적으로 토큰 발행 기능을 가진다.

3.1.2 라이트 노드(light node)

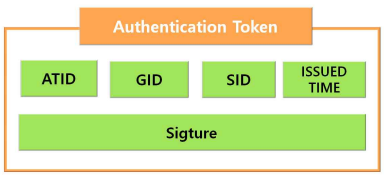
라이트 노드는 풀 노드에 비하여 제한적인 하드웨어 사양을 가진 노드이다. 자체적으로 블록을 생성하고 합의하는 기능이 없으며, 네트워크에서 생성되는 블록 중 해당 라이트 노드와 관련된 블록만을 저장하는 경량 Wallet를 가진다.

3.2 토큰

본 논문에서는 토큰을 인증 토큰과 접근 권한으로 구분한다. 인증 토큰은 노드가 소속되어 있는 도메인의 게이트웨이에게 인증 과정을 수행하고 인증이 완료되면 발급되는 토큰이다[5, 8]. 노드가 특정한 풀 노드나 라이트 노드인 리소스에 대한 접근 권한을 필요로 할 경우에는 인증 토큰을 소유하고 있어야한다. 노드가 리소스에 접근하기 위해서는 해당 리소스가 소속되어 있는 도메인의 게이트웨이에게 인증 토큰과 함께 접근 권한을 요청하면 게이트웨이는 Wallet에 저장되어 있는 블록들을 통해 인증 토큰을 검증하고 해당 리소스에 대한 접근 권한 토큰을 발급한다.

3.2.1 인증 토큰

게이트웨이는 인증 과정을 수행하여 주체에게 인증 토큰을 발급한다. 인증 토큰은 (Fig 2)와 같이 구성하였다.

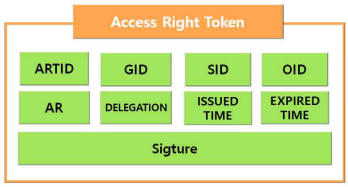


(Fig 2) Authentication Token

ATID를 통해 인증 토큰을 식별하며 GID와 SID를 통해 발급한 게이트웨이와 발급 받은 주체를 식별한다. Sigture를 통해 토큰의 무결성을 보장한다.

3.2.2 접근 권한 토큰

주체가 접근 권한을 요청하면 인증 토큰을 통해 주체를 검증하고 접근 권한 토큰을 발급한다. 접근 권한 토큰은 (Fig 3)과 같다[3].



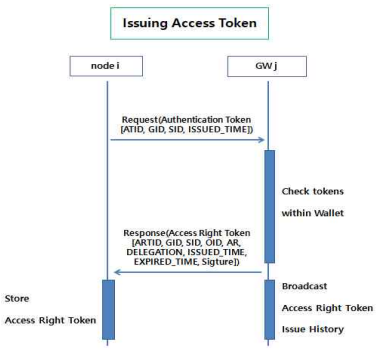
(Fig 3) Access Right Token

ARTID를 통해 접근 권한 토큰을 식별하고 GID, SID, OID를 통해 발급자, 발급 받은 주체와 접근 대상이 되는 객체를 식별한다. AR를 통해 read, write와 같은 접근 권한을 나타내고 DELGATION를 통해 토큰에 대한 위임 가능 여부를 나타낸다. Sigture를 통해 토큰의 무결성을 보장한다.

3.3 접근 권한 토큰 발급

노드가 특정한 풀 노드나 라이트 노드인 리소스에 접근하기 위해서는 우선적으로 소속되어져 있는 도메인의 게이트웨이와 인증 과정을 수행하고 인증 토큰을 발급 받아 소유하고 있어야한다[1, 2].

인증 토큰을 발급 받은 노드는 리소스에 접근하기 위해 접근 권한 토큰을 발급받아야한다. 노드는 접근 권한 토큰을 발급 받기 위해서 (Fig 4)와 같은 절차를 따른다.

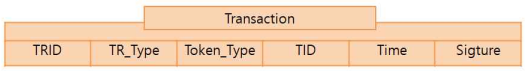


(Fig 4) Issuing Access Token

접근 대상이 소속되어져 있는 도메인의 게이트웨이에게 인증 토큰을 전송하여 접근 권한 토큰을 요청한다. 인증 토큰을 전송 받은 게이트웨이는 Wallet에 저장되어 있는 블록체인에서 해당 인증 토큰의 무결성과 신뢰성을 검증한다. 검증이 완료되면 노드에게 해당 리소스에 대한 권한을 명시한 접근 권한 토큰을 발급한다. 그리고 발급한 내역을 트랜잭션 형태로 구성하여 블록체인 네트워크에 브로드캐스트한다. 노드는 발급 받은 접근 권한 토큰을 저장함으로서 접근 권한을 요청하기 위해 발급 과정을 재차 수행하지 않고 저장한 토큰을 이용하여 지속적으로 리소스에 접근 할 수 있다[3].

3.3.1 트랜잭션과 블록 구조

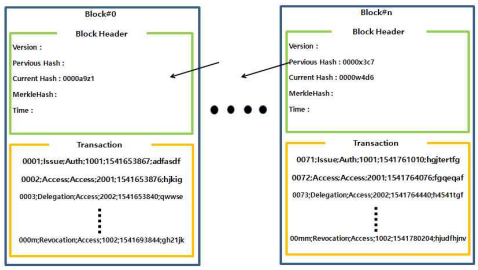
게이트웨이가 노드에게 토큰 발급한 다음 IoT 블록체인 네트워크에 있는 다른 노드들에게 발급 내역을 트랜잭션 형태로 브로드캐스트한다. 트랜잭션을 통해 발급한 토큰에 대한 신뢰성을 보장할 수 있도록 한다. 본 논문에서 정의한 트랜잭션 구조는 다음 (Fig 5)와 같다.



(Fig 5) Transaction Structure

TRID를 통해 트랜잭션을 식별하고 TR_Type를 통해 해당 트랜잭션에서의 행위를 토큰 발급, 사용, 위임, 폐기로 구분한다. Token_Type과 TID를 통해 토큰의 종류을 구분하고 식별한다. Sigture를 통해 토큰의 무결성을 보장한다[4].

트랜잭션을 토대로 헤더를 추가하여 블록을 생성하고 Wallet에 저장 한 다음 체인을 형성한다. 본 논문에서 정의한 블록 구조는 다음 (Fig 6)과 같다.

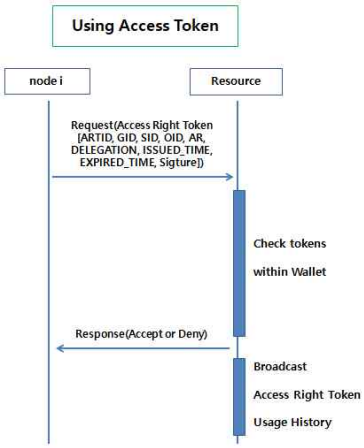


(Fig 6) Block Structure

한 블록 내에는 m개의 트랜잭션을 저장한다. 토큰에 대한 검증이 필요할 경우 해당 토큰에 대한 블록과 트랜잭션을 검색하여 검증을 수행한다.

3.4 접근 권한 토큰 사용

접근 권한 토큰 발급 과정을 통해 토큰을 발급 받은 노드는 리소스에게 접근 요청을 하여야한다. 노드가 리소스에게 접근 요청을 하는 과정은 다음 (Fig 7)과 같다.



(Fig 7) Using Access Token

노드는 리소스에게 접근 권한 토큰을 전송하여 접근 요청을 한다. 접근 권한 토큰을 전송받은 리소스는 Wallet에서 접근 권한 토큰의 무결성과 신뢰성을 검증한다. 검증을 수행한 다음 리소는 검증 결과를 보냄으로서 노드에게 응답한다.

3.5 보안성 분석

인증 토큰과 접근 권한 토큰에 대한 발급 내역을 블록체인에 저장함으로서 악의적인 노드가 토큰을 위조하여 접근 권한을 취득하는 시도를 방지할 수 있다. 그리고 토큰의 Sigture를 블록체인에 저장함으로서 악의적인 노드가 소유자ID 변조 등의 토큰 변조를 통한 접근 권한 취득 시도 또한 방지할 수 있다.

IV. 결론

본 논문에서는 중앙 집중적인 시스템의 문제점을 해결하기 위해 IoT 블록체인을 이용하여 토큰을 관리함으로서 권한 관리를 수행하는 시스템을 제안하였다.

풀 노드와 라이트 노드로 IoT 블록체인 네트워크를 구성하였으며, 인증 토큰과 접근 권한 토큰의 구조를 정의하였다. 인증 과정을 통해 인증 토큰을 발급하고 발급된 인증 토큰을 통해 접근 권한을 요청한다. 인증 토큰을 블록들이 저장되어 있는 Wallet에서 검증한 다음 검증이 완료되면 접근 권한이 명시됭 접근 권한 토큰을 발급한다. 발급한 사실을 블록체인 네트워크에 브로드캐스트하여 토큰에 대한 무결성과 신뢰성을 확보하였다. 리소스에게 접근 권한 토큰을 전송하여 접근 요청을 한다. 리소스는 Wallet에서 해당 토큰에 대하여 검증한 다음 검증 결과를 보냄으로서 응답한다. 블록체인을 통해 토큰에 대한 무결성과 신뢰성을 확보하여 토큰 위변조를 통한 접근 권한 취득을 방지할 수 있다.

향후 연구에서는 토큰에 대한 기밀성과 가용성을 보장하고 중간자 공격을 방지할 수 있는 방법을 연구할 계획이다.

ACKNOWLEDGMENT

본 논문(연구)은 교육부의 재원으로 이공분야기초연구사업의 지원을 받아 수행된 연구임(No. NRF-2018R1D1A1B07051203).

[참고문헌]

[1] Hyung Wook Kim, "A Design of mutual authentication protocol between heterogeneous services in the internet of things Environment", Ph.D. Dissertation, Soongsil University, Korea, 2017.

[2] Sungyuck Hong, Sanghee Park, "The Reserarch on Blockchain-based Secure Iot Authentication", Journal of the Korea Convergence Society vol.8, No 11, pp. 57-62, 2017.

[3] Bum-Ki Lee, Mi-Sun Kim, Jae-Hyun Seo, "Design and Implementation of The Capability Token based Access Control System in the Internet of Things", Journal of The Korea Institute of Informaion Securty & Cryptology vol.25, no.2, April, 2015.

[4] Ali Dorri, Salil S. Kanhere, Raja Jurdak, Praveen Gauravaram, "Blockchain for IoT Security and Privacy: The Case Study of a Smart Home", IEEE PERCOM WORKSHOP ON SECURITY PRIVACY AND TRUST IN THE INTERNET OF THING, March, 2017.

ARP 스푸핑을 방지하기 위한 블록체인 활용 모델 제안*

김광훈*, 염흥열**

*순천향대학교 융합서비스보안학과,**순천향대학교 정보보호학과

A Proposed Blockchain-based Utilization Model to Prevent ARP Spoofing

Kwang Hun Kim*, Heung Youl Youm**

*Department of Convergence Service Security, SoonChunHyang University.

**Department of Information Security Engineering, SoonChunHyang University.

요 약

오늘날 네트워크 상의 공격은 여전히 증가하고 있으며 그 중 대표적인 스푸핑과 스니핑 공격이 복합적으로 사용되어 우리에게 위협을 주고 있다. ARP 스푸핑은 OSI 7 계층 중 2~3계층 사이의 ARP 프로토콜을 악용한 위/변조 공격 기법으로 LAN 상의 영향력을 받는 클라이언트들은 모두 공격자의 표적이 되거나 개인정보 유출, 좀비 PC로도 악용 될 수 있다. 이러한 공격을 방지하기 위해선 실시간으로 ARP 스푸핑의 변조를 탐지하거나 대응하거나 공격 자체의 시도를 불가능하게 해야 한다. 현재 이에 대응하기 위한 솔루션들이 제안되고 있지만 확장성, 의존성, 유연성 부재와 같은 한계가 존재한다. 본 논문에서는 기존의 연구에서 제시한 ARP 스푸핑 대응 방안의 한계점을 식별하고 이를 보완하기 위한 방법으로 블록체인을 활용한 ARP 스푸핑 방지 모델을 제안한다.

I. 서론

오늘날 네트워크 상에서의 공격은 여전히 증가하고 있으며 하나의 공격 기법이 아닌 여러 공격 기법들이 복합적으로 사용되어 네트워크 사용자들에게 위협을 주고 있다. 그 중 스푸핑과 스니핑 공격은 이에 근거한 대표적인 2way 공격 유형이다.[1]

로컬 네트워크 환경이 예전의 허브 중심에서 현재 스위치 기반으로 변화하면서 스니핑 공격이 힘들어지자 이에 공격자들은 ARP 스푸핑을 통한 네트워크의 통제하에 스니핑을 수행하고 있다.[2]

ARP 스푸핑은 현재 네트워크의 OSI 7 Layer 모델 중 2~3계층 사이의 ARP 프로토콜을 악용한 위, 변조 공격 기법으로 LAN 상의 영향력을 받는 클라이언트들은 모두 공격자의 표적이 되어 개인정보 유출은 물론 좀비 PC로서 범죄에 악용될 수 있다.[3] 이러한 공격을 방지하기 위해서는 실시간으로 ARP 스푸핑의 변조를 탐지하여 대응하거나 공격 자체의 시도를 처음부터 불가능하게 해야 한다.

현재 ARP 스푸핑에 대응하기 위한 솔루션 장비와 다양한 메커니즘이 제안되고 있지만 다양한 LAN 환경에 적용하기엔 장비의 의존성, 확장성, 유연성 부재와 같은 어느 정도의 한계가 따른다.[4]

본 논문에서는 기존의 연구에서 제시한 ARP 스푸핑 대응 방안의 한계점을 식별하고 이를 보완하기 위한 방법으로 블록체인을 활용한 ARP 스푸핑 방지 모델을 제안한다. 블록체인은 공개된 규칙을 통한 상호인증과 그 인증 과정을 투명화 함으로서 신뢰를 보장하는 특성을 가진다. 따라서 본 논문의 목적은 ARP 스푸핑의 근본적인 문제인 상호 인증 과정을 블록체인을 활용하여 해결하고자 하며 의존성, 확장성, 유연성을 위한 방향을 함께 제시한다.

II. 기존 연구

2.1 ARP 스푸핑 공격에 대한 이해

* 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "고용계약형 정보보호 석사과정 지원사업"의 연구결과로 수행되었음

보통 ARP 스푸핑 공격은 OSI 7 layer의 이더넷/IP구조를 따르는 LAN 상에서 수행되며, 2~3계층에서 사용되는 ARP 프로토콜의 허점을 이용한다. 본래 LAN 상에서의 통신은 MAC 주소를 기반으로 이루어지며 노드의 할당된 IP와 매칭된 MAC 주소를 찾기 위해 ARP 프로토콜이 사용된다.[5] 하지만 공격자가 악의적인 목적으로 MAC 주소를 자신으로 위/변조한 ARP 응답 메시지를 네트워크 상에 전파할 경우 해당 네트워크상의 모든 노드는 공격자의 영향력 아래에 놓이게 된다. 이후 공격자는 표적의 메시지를 쉽게 도청하거나 가로채어 위/변조 할 수 있다. 이러한 공격 과정을 쉽게 설명한 그림은 다음과 같다.

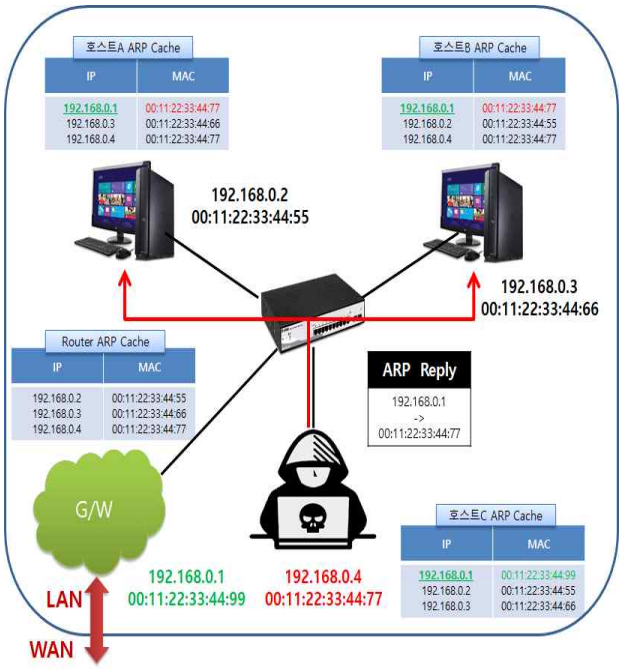


그림 1. ARP 스푸핑의 공격 과정

먼저 공격당하기전 호스트들은 기존의 ARP 캐시 테이블을 참조하여 LAN 상의 호스트 또는 G/W와 정상적인 통신을 수행하며 일정 시간이 지난 후 캐시의 갱신을 위한 ARP 요청 메시지를 전송한다. 공격자는 이때 G/W로 위장한 ARP 응답 메시지를 반복적으로 전송한다. 그 결과 각 호스트들은 원래의 G/W 주소 192.168.0.1 -> 00:11:22:33:44:99가 아닌 위조된 공격자의 주소 00:11:22:33:44:77를 받아들인다. 이 후 G/W로 가야될 메시지는 공격자에게 전달되어 공격자의 원하는 목적에 기여하게 된다.

2.2 ARP 스푸핑 대응 방안

현재 이러한 ARP 스푸핑 공격에 대응하기 위한 여러 가지 방안이 연구되고 있다. 그 중엔 다음과 같은 대표적인 2가지 메커니즘이 주로 사용된다.

2.2.1 시스템 상의 정적 ARP 테이블 구성

로컬 네트워크 상의 호스트들은 그 수가 제한되어 있으며 유동성이 비교적 적다. 따라서 해당 방식은 고정된 호스트들의 IP주소와 매칭되는 MAC 주소를 정적으로 ARP 테이블에 등록하여 관리한다. [6]

```
Example:
 > arp -s 157.55.85.212   00-aa-00-62-c6-09  .... Adds a static entry.
 > arp -a                                    .... Displays the arp table.
```

그림 2. 윈도우 기반의 ARP 정적 테이블 구성[7]

이러한 방식은 ARP 스푸핑 공격을 원천적으로 무시하지만 호스트에 문제가 발생하여 주소를 변경하거나 네트워크가 DHCP 기반의 자동 IP 주소 부여 방식을 사용한다면 매번 수동으로 매칭된 주소를 등록해야하는 번거로움이 따른다. 따라서 유동성이 잦은 네트워크 환경일 경우 해당 방식은 적합하지 않다.

2.2.2 네트워크 장비의 사설 VLAN 기능

해당 방식은 네트워크 장비에서 지원하는 VLAN 기능을 활용한다. 이는 동일 서브넷 상에서 지정한 호스트끼리 통신이 가능하도록 함으로써 공격자의 ARP 스푸핑 대상 범위를 제한하는데 유용하지만 같은 브로드캐스트 도메인 영역 내의 VLAN에 속한 호스트들에게는 여전히 공격이 유효한 한계점이 존재한다. 이 외에도 네트웍 장비의 VLAN 기능 지원유무에 따른 의존성과 네트워크 확장 구성에 따른 추가적인 비용이 소모될 수 있다. [8]

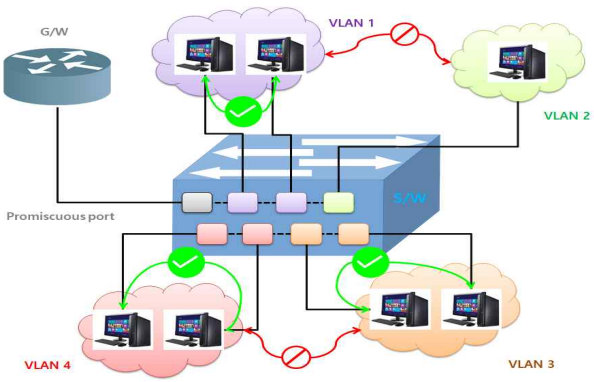


그림 3. 사설 VLAN 구성도

앞선 기존의 연구들은 ARP 스푸핑이 발생하는 근본적인 원인인 ARP 프로토콜의 상호 인증 문제를 배제한채 공격자로부터 ARP 스푸핑이 시도될 경우를 대비한 예방책에 가깝다. 이에 반해 본 논문에서는 ARP 프로토콜의 허점인 상호 인증 문제를 블록체인을 활용하여 해결함으로써 애초에 공격자로부터 ARP 스푸핑 시도를 불가능하게 하는데 목적을 둔다.

III. 블록체인 활용 모델

3.1 개요

블록체인은 관리 대상이 되는 데이터를 체인 형태의 연결고리 기반 분산 데이터 환경에 저장하여 누구든 임의로 수정 할 수 없게 하고 누구나 그 결과를 열람할 수 있는 데이터 위/변조 방지 기술이다.[9] 따라서 이러한 블록체인 특성을 이용하면 ARP 프로토콜의 상호 인증 문제를 해결할 수 있다. 본 논문에서 제안하는 ARP 스푸핑을 방지하기 위한 블록체인 활용 모델은 다음과 같다.

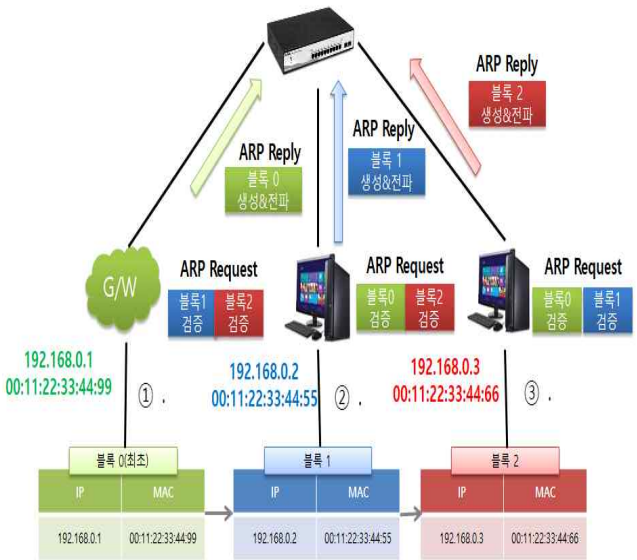


그림 4. ARP 스푸핑 방지 위한 블록체인 모델

해당 모델은 상호 인증 과정을 수행하지 않는 ARP 프로토콜의 허점을 악용한 ARP 스푸핑 공격을 방지하기 위해 기존의 ARP 캐시 내역을 일정한 규칙에 근거하여 블록체인상에 기록한다. 이 후 네트워크상의 호스트들은 블록체인에 기록된 데이터를 검증하고 이를 기반으로 상호인증을 수행함으로써 공격자의 ARP 스푸핑 시도를 무효화 한다.

3.2 블록의 업로드

기존 ARP Reply 대신 수행되는 작업으로 로컬 네트워크상의 호스트가 통신을 위한 기본적인 주소 체계를 최초로 부여받은 경우 매칭된 IP, MAC 주소 및 인증을 위한 정보를 블록체인에 기록하여 네트워크상에 브로드캐스트 한다. 이는 상대방과의 통신을 위한 전제조건으로 호스트 자신을 먼저 증명하는 과정이다. 이때 호스트의 MAC 주소와 일치하는 1개의 IP만을 블록에 기록할 수 있으며, 매칭되는 IP가 변경될 경우 블록을 갱신해야 한다. 블록체인에 기록되는 블록 데이터 구성은 다음과 같다.

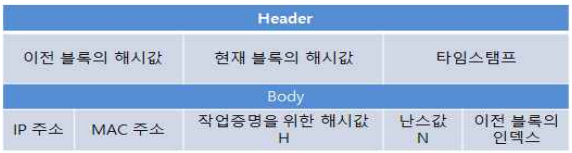


그림 5. 블록 데이터 구성

[표 1] 데이터 구성 요소

요소	내용
이전 블록의 해시값	이전 블록의 현재 블록의 해시 값이다. 이를 통해 체인의 영속성을 유지할 수 있다. (최초의 블록 0에서 이전 블록의 해시 값은 none이다.)
현재 블록의 해시값	이전 블록의 해시 값과 데이터 정보를 합쳐 해시 처리한 결과 값이다.
타임스탬프	해당 블록을 생성한 시각이다. 블록의 갱신 주기는 타임스탬프 값을 따르며 그 주기는 10분 간격으로 갱신된다.(이는 ARP 테이블의 갱신 주기와 동일하다.)
IP	정적, 또는 동적으로 부여받은 호스트의 IP 주소이다.
MAC	해당 호스트 랜카드의 MAC 주소이다.
작업 증명을 위한 해시값	SHA512(MAC, N)를 적용한 해시 값이다. 이를 통해 새로 갱신된 주소가 이전의 호스트가 사용한 것인지 검증 할 수 있다. 이 때 작업 증명 해시 값을 계산하기 위해서는 일정시간의 노력이 필요하며 사용된 난스 값은 해당 호스트가 안전하게 보관하고 있어야한다. (만약 난스가 유출될 경우 자신이 등록했던 매칭 주소가 공격자에 의해 변조될 수 있다.)
난스값	호스트의 매칭된 주소를 새로 갱신하고 증명하기 위해 사용되는 값이다. (최초로 매칭 주소가 기록된 블록의 난스 값은 none이며 갱신된 다음 블록부터 이전의 난스 값이 공개된다.)
이전 블록의 인덱스	이전 호스트의 검증을 위해 현재의 블록과 연결된 이전의 블록 번호이다.

여기서 G/W, Host A, Host B순으로 네트워크가 구성된다고 가정 할 때 초기 네트워크 구성에서 블록을 업로드 하는 과정은 다음과 같다.

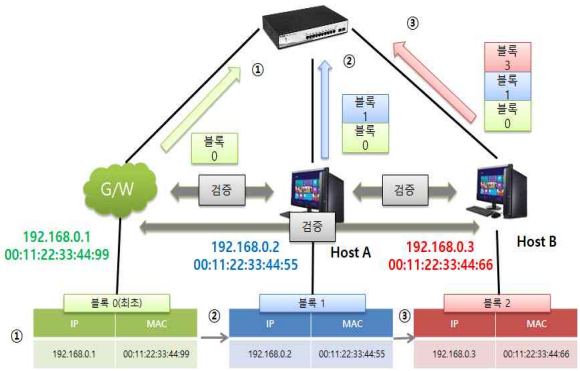


그림 6. 초기 네트워크상의 블록 업로드 과정

1. 가장 먼저 네트워크에 연결된 G/W는 기존의 생성된 블록이 없음을 확인하고 자신의 매칭된 IP, MAC 주소를 기록한 첫 번째 블록0을 생성하여 네트워크상에 브로드캐스트 전송한다.
2. Host A는 3.3의 블록 다운로드 과정을 통해 블록 0이 있음을 확인하고 블록 0에 이은 자신의 매칭된 IP, MAC주소를 추가한 블록 1을 네트워크상에 브로드캐스트 전송한다.
3. Host B도 3.3의 블록 다운로드 과정을 통해 블록 0과 1이 있음을 확인하고 블록 0과 1에 이은 자신의 매칭된 IP, MAC주소를 추가한 블록 3을 네트워크상에 브로드캐스트 전송한다.

결과적으로 G/W와 Host A, B는 서로 동기화된 블록0, 1, 2를 얻게 되며 이 때 해당 블록을 전송한 호스트의 일치여부를 위한 3.5의 블록 검증 과정을 수행한다. 이로써 각 호스트들은 공유된 블록의 데이터를 참고하여 해당 호스트를 신뢰할 수 있다.

3.3 블록의 다운로드

기존의 ARP Request 대신 수행되는 작업으로 초기 네트워크의 호스트를 제외한 새로운 호스트가 네트워크에 추가되거나 기존 블록들이 누락될 경우 해당 호스트는 이전의 블록들을 내려 받기 위한 선행 작업을 수행한다. 그 과정은 다음 그림과 같다.

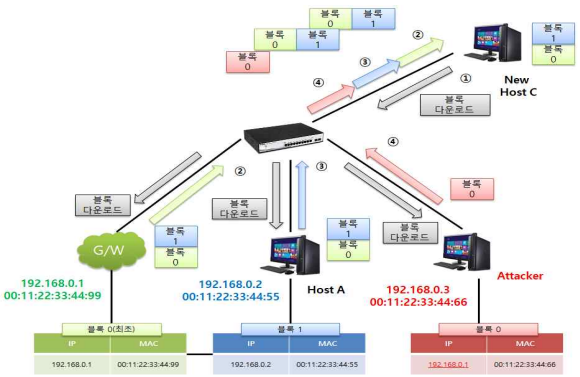


그림 7. 초기 네트워크상의 블록 다운로드 과정

1. 새로운 호스트 C는 네트워크상에 블록 다운로드를 수행하기 위한 패킷을 브로드캐스트 전송한다.
2. 기존의 호스트들은 이전에 기록하고 있던 블록들을 송신자 주소로 유니캐스트 전송한다.
3. 이 과정에서 공격자는 자신이 G/W라고 위조한 블록을 호스트 C에게 전송한다.
4. 새로운 호스트 C는 다운받은 각 블록들의 최종 해시 값을 기준으로 비교 수행하고 그 중에서 일치하는 경우의 수가 많은 블록을 신뢰하여 수신하게 된다. (이 방식은 블록체인 상의 선의의 호스트가 악의적인 호스트보다 많은 경우 유용하다.)

이러한 과정이 끝나게 되면 호스트 C는 기존의 블록에 이어 자신의 할당된 매칭 주소를 추가한 블록을 생성하여 네트워크상에 업로드 한다.

3.4 블록의 갱신

처음으로 블록체인에 기록된 매칭 주소가 네트워크의 사정상 변경되어야하거나 타임스탬프의 갱신 주기 10분에 근접했을 경우 블록의 갱신을 통해 해당 MAC 주소에 해당하는 IP를 수정 또는 유지 할 수 있다. 블록의 갱신을 위해서는 이전에 매칭된 주소를 기록한 블록의 작업 증명 해시 값에 사용된 난스 값을 알고 있어야 한다. 블록 갱신은 3.2 블록의 업로드 과정과 동일하며 추가적으로 다음 블록에 새로 매칭된 주소와 이에 대한 작업 증명 해시 값을 계산한 뒤 난스 필드에 이전 블록의 난스 값을 넣어 갱신한다. 다음의 그림은 실제 블록의 데이터 구성요소를 기반으로 블록 3,4,5를 갱신 하는 과정이다.

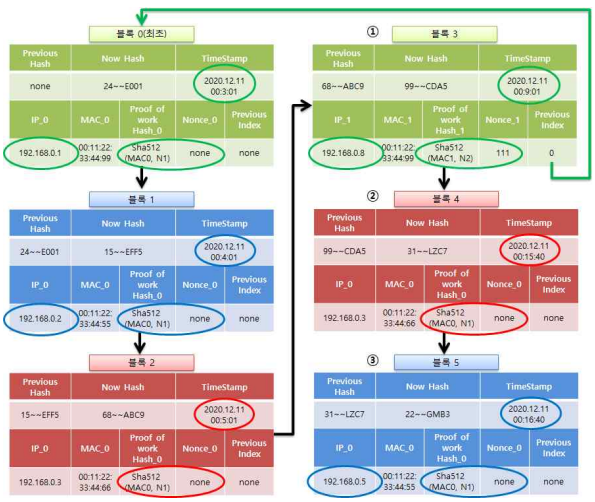


그림 8. 타임스탬프에 따른 블록 갱신 과정

1. 블록0을 최초로 생성한 G/W는 생성 시의 타임스탬프 기준 10분 이내로 IP가 변경된 새로운 블록3으로 갱신한다. 이 때 블록0의 작업증명 해시 값에 사용된 난스 값은 블록3의 난스 필드에 기입하고 이전 블록의 인덱스 필드에는 블록 0을 기입하여 이전 에 사용된 주소를 참조할 수 있도록 한다.

2. 현재 시간 기준 블록2를 최초로 생성한 Host B가 타임스탬프 갱신 주기 10분 이후에 블록4를 갱신한다. 이 때 10분이 지나 갱신하기 전까지의 블록2에 기록된 데이터는 무효화되며 새로 갱신한 블록4는 최초 업로드한 블록2와 동일하며 타임스탬프 값만 새로 갱신된다.
3. 현재 시간 기준 블록1를 최초로 생성한 Host A가 타임스탬프 갱신 주기 10분 이후에 블록5를 갱신한다. 이 때 10분이 지나 갱신하기 전까지의 블록1에 기록된 데이터는 무효화되며 새로 갱신한 블록5는 최초 업로드한 블록5와 동일하며 변경된 IP와 타임스탬프 값만 새로 갱신된다.

여기서 블록을 갱신할 때 사용된 난스 값은 해당 호스트의 이전 주소를 검증하기 위해 사용되며 타임스탬프 값은 실제 사용되지 않은 주소가 블록에 남아 있는 것을 방지하기 위한 용도로 사용된다.

3.5 블록의 검증

블록의 업로드/갱신 과정에서 해당 블록과 호스트의 일치여부를 위한 검증을 수행한다. 구성 요소별 검증 조건은 다음과 같다.

[표 2] 데이터 구성요소에 따른 검증 조건

요소	조건
이전 블록의 해시값	현재 블록의 이전 블록 해시 값은 이전 블록의 현재 해시 값과 동일해야 한다.
현재 블록의 해시값	현재 블록의 헤더와 바디를 SHA512에 적용한 결과 값과 현재 블록의 해시 값은 동일해야 한다.
타임스탬프	현재 블록의 타임스탬프 값은 이전 블록의 타임스탬프 이후여야 하며 현재 시간 기준 10분 이내인 타임스탬프를 포함한 블록만 유효하다.
IP	최근 생성된 블록을 기준으로 하나의 IP당 한 개의 MAC 주소만 유효하다.
MAC	최근 생성된 블록을 기준으로 하나의 IP당 한 개의 MAC 주소만 유효하다.
난스값	SHA512(이전 블록의 MAC, 현재 블록의 난스)결과 값과 이전 블록의 작업 증명 해시 값이 동일한 블록만 유효하다.
이전 블록의 인덱스	인덱스 값이 참조하는 번호의 블록 MAC주소가 현재 블록의 MAC주소와 동일해야 한다.

검증 조건이 완전히 일치한다면 최종적으로 해당 호스트가 존재하는지 확인하기 위한 메시지를 유니캐스트로 전송하여 호스트의 응답 여부를 확인한다. 이상이 없을 경우 해당 블록을 정상적인 것으로 간주하고 수신하게 된다. 이 후 검증된 블록들을 기준으로 각 호스트의 IP에 해당하는 MAC 주소를 찾을 때는 최근 생성된 블록 순으로 찾는다. 이러한 규칙 구조로 인하여 하나의 MAC 주소는 한 개의 IP에만 매칭 될 수 있다. 그 과정을 설명한 그림은 다음과 같다.

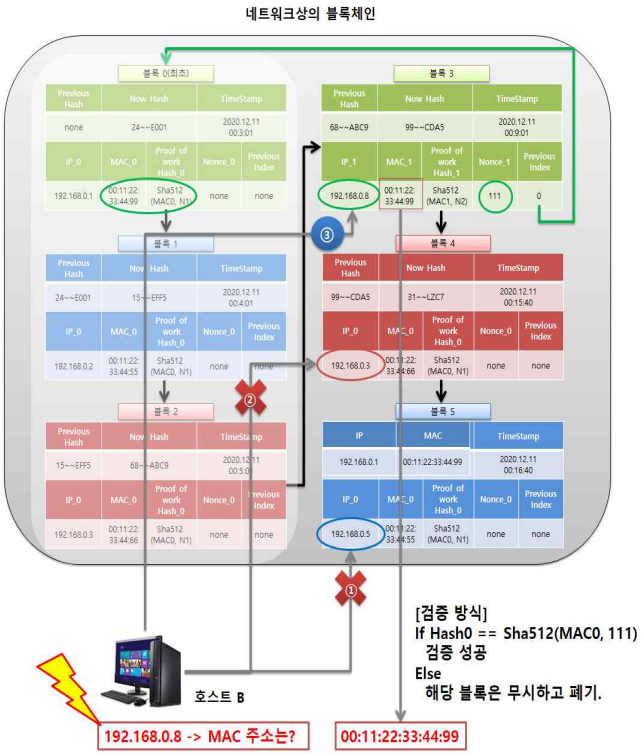


그림 9. 검증된 블록체인에서 호스트 주소 찾기

결과적으로 각 호스트들은 기존의 ARP캐시 테이블 대신 블록체인상의 검증된 데이터를 기반으로 자신과 매칭된 주소를 난스 값을 통해 증명함으로써 네트워크상의 신뢰성을 얻게 된다. 이 후 작업 증명 해시 값의 비교를 통해 호스트간의 인증을 쉽게 수행할 수 있다. 이 때 공격자는 호스트가 블록을 생성할 때 사용한 난스 값을 알지 못하기 때문에 현재 네트워크상에서 사용하고 있는 IP에 대한 위조된 MAC주소로 새로운 블록을 생성 할 수 없다. 만약 공격자가 난스 값을 구하기 위한 시도를 한다면 일방향 해시 함수의 저항성을 깨기 위한 노력이 필요할 것이다. [10]

IV. 결론

본 논문에서는 ARP 스푸핑과 기존 연구의 한계점인 장비 의존적인 문제에 대해서 블록체인 모델을 도입해 호스트간의 상호 인증에 대한 영향력을 분산화 시켜 해결하였다. 또 확장성, 유동성 부재와 같은 문제를 블록의 갱신을 통하여 해결함으로써 변화에 민감한 네트워크를 상황에 맞게 구상할 수 있었다. 이로써 공격자 입장에서 LAN 상의 ARP 스푸핑 공격에 대한 시도가 줄어들 것으로 기대하며 LAN이 아닌 WAN 영역의 다양한 스푸핑 공격 기법들 또한 본 논문의 모델을 응용하여 일정 수준의 대응이 가능 할 것으로 예상된다.

[참고문헌]

[1] Behrouz A. Frouzan, "암호학과 네트워크 보안" 2008년
[2] "ARP Spoofing 공격 및 대책", KISA, http://krcert.or,kr
[3] "ARP Spoofing 악성파일 확산에 따른 예방 조치법", nprotect, http://erteam.nprotect.com/88
[4] "ARP Spoofing 공격 및 대책," KISA, http://krcert.or,kr
[5] "네트워크 통신의 비밀 ARP", 넷센터 http://http://index-of.co.uk/Networking/ARP.pdf
[6] "ARP Spoofing 원리 탐지방법 및 대책" http://blog.naver.com/PostView.nhn?blogId=kimhmin85&logNo=90069864452
[7] "ARP Spoofing 공격 및 대책", KISA, http://krcert.or,kr
[8] "클라우드 환경에서 ARP 스푸핑 방지 메커니즘 구현하기", https://d2.naver.com/helloworld/533511
[9] "블록체인", 위키피디아
[10] "해쉬 충돌 저항성", 위키피디아

블록체인 기반의 산업제어시스템 접근 권한 관리*

조민정†, 이창훈‡
서울과학기술대학교 컴퓨터공학과

BlockChain-Based Industrial Control System Access Control Management

Min-Jeong Cho, Changhoon Lee
Seoul National University of Science and Technology.

요 약

산업제어시스템은 국가 기반 시설 운용에 사용되며 침해사고 발생시 전력마비, 발전소 중지등 심각한 결과를 초래할 수 있다. 안전한 산업제어시스템의 운용을 위해 사용자의 접근제어는 중요하다. 본 논문에서는 블록체인을 이용하여 사용자의 키와 권한을 관리하여 접근제어와 인증서 발급에 사용할 수 있도록 하였다. 시스템의 사용자를 블록체인으로 관리하여 사용자의 역할에 변동이 있는 경우 바로 접근 권한이 변동되도록 한다. 또한 사용자 등록시 키쌍을 생성하여 접근하는 사람에게 인증서를 발급할 때 사용할 수 있도록 하였다. 제안하는 시스템은 산업제어시스템의 물리적인 변경없이 소프트웨어 변경만으로 적용이 가능하다.

I. 서론

산업제어시스템은 산업 분야 및 원자력 발전, 전력, 수력 등 국가 기반 시설 운용에 사용되는 시스템이다[1]. 산업제어시스템에는 다수의 시스템이 포함될 수 있으며 각 시스템에는 여러 사용자가 접근한다[2]. 산업제어시스템에서 비인가된 사용자의 악의적 접근은 금전적 손해뿐만 아니라 인적 손실도 초래할 수 있어 각별히 인가된 사용자만이 접근해야한다.

산업제어시스템에서 기기의 사용자는 언제나 변경될 수 있다. 인가된 사용자만의 접근을 위해서 사용자의 권한에 변동이 있는 경우 접근 제어도 즉시 이루어져야한다. 본 논문에서 제안하는 시스템은 사용자의 역할 및 역할에 따른 권한을 블록체인을 이용하여 관리하여 변동사항이 있을시 즉시 접근 제어에 반영할 수 있다.

사용자 인증을 위해 산업제어시스템에서는 자체 인증서를 사용할 수 있다. 본 논문에서는 블록체인을 이용하여 자체 인증서 서버가 공격당하였을 때 피해를 완화하였다.

II. 산업제어시스템

산업제어시스템은 현장장치 계층, 제어계층, 운영계층으로 구성된다[3].

계층	역할	예시
현장장치 계층	상태 데이터를 계측, 수집	센서, 액추에이터
제어계층	수집된 데이터를 운영계층에 전달	DCS,PLC, RTU
운영계층	필드 장치의 상태를 모니터링하거나 제어 명	HMI,EWS

* 본 연구는 산업통상자원부(MOTIE)와 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다.(원전 비안전등급 제어기기(DCS) 사이버침해 예방 및 탐지 기술 개발, No. 20161510101810).

† 주저자, chomj@seoultech.ac.kr
‡ 교신저자, chlee@seoultech.ac.kr(Corresponding author)

	령을 제어계층에 전달	

표 2 산업제어시스템의 구성

운영 계층에 속하는 HMI(Human Machine Interface), EWS(Enginnerring Workstatino)는 고성능 장치이다. 제어계층에 속하는 DCS(Distributed Contol System), PLC(Programmable Logic Controller), RTU(Remote Terminal Unit)은 산업 프로세스에 특화된 장치로 자원이 한정되어 있다.

III. 블록체인

블록체인은 2009년 발표된 비트코인의 핵심 개념으로 알려졌다[4]. 기존의 시스템들은 TTP(Trusted Third Partay)에 의존하여 모든 거래 내역, 정보 등을 저장하였다. 블록체인은 P2P(Peer to Peer) 분산 네트워크에서 모든 참여자가 검증과 동의를 통해 새로이 생성 및 검증되도록 한 시스템이다[5]. 검증과 동의를 통해 새로이 생성 및 검증을 하는 참여자를 노드 혹은 채굴자라고 칭한다. 장부에 기록되는 내역은 트랜잭션으로 돈거래 내역, 스마트 계약에 따른 실행 내역등이 될 수 있다. 여러 트랜잭션이 모여 하나의 블록이 되며 각 블록은 이전 블록의 해쉬값과 현 블록을 포함한 해쉬값을 저장하여 무결성을 보장하였다.

블록체인은 누구나 자료 열람 및 거래 가능한 퍼블릭 블록체인, 미리 선정된 노드에 의해서 컨트롤되는 컨소시엄 블록체인, 승인 기관의 합의 알고리즘에 따르며 소수 기관의 권한을 통해서만 거리가 검증되고 처리되기 때문에 외부에서는 접근 불가능한 프라이빗 블록체인으로 분류할 수 있다[5].

IV. 사전 연구

JASON PAUL CRUZ, YUICHI KAJI, NAOTO YANAI는 스마트 계약을 이용하여 역할기반의 접근제어를 제안하였다. 기업들 간에 사용자에 대한 정보를 인증하는데 사용하는 스마트 계약이다. 역할 인증은 블록에 등록되며 사용자는 블록에 올라온 인증을 본인의 서명을 이용하여 증명할 수 있다[6].

Ouaddah, A, Abou Elkalam, A, Ait Ouahman는 블록체인을 기반으로 IoT에서 접근 제어를 위한 프레임 워크를 제안하였다. 정책 검색 지점, 접근 제어 정책, 로그등이 트랜잭션 형태로 저장되며 제어 정책은 스마트 계약을 이용하여 구성된다. 액세스 토큰은 UTXO모델을 기반으로 하여 bitcoin대신 엑세스 토큰을 전송한다[7].

Maesa, D. D. F., Mori, P., and Ricci, L은 블록체인을 이용한 접근 권한을 제안하였다. 권한 생성 트랜잭션과 권한 전송 트랜잭션이 블록체인에 저장된다. 접근 제어 정책은 XACML로 작성된다. 그러나 XACML은 형식이 복잡하고 양이 방대하여 블록체인에 저장하기 힘들다. XACML 정책을 외부에 저장하고 블록체인에는 링크만을 저장하여 문제를 해결하였다[8].

V.제안하는 시스템

2.1 시스템 체계

제안하는 시스템에서는 산업제어시스템에서 EWS 및 HMI등 운영계층의 연산장치가 블록체인 노드이다. 각 노드들은 공개키와 개인키가 할당된 계정을 가지며 각 계정은 제네시스 블록 배포시 생성된다. 제네시스 블록은 블록체인의 첫번째 블록이다. 이때 노드들로 구성된 블록체인은 프라이빗 블록체인으로 외부에서 접근이 불가능하다.

각 노드들은 사용자의 계정, 계정의 역할, 역할에 따른 권한을 트랜잭션의 형태로 저장한다. 사용자의 계정을 저장하는 트랜잭션에는 사용자의 공개키를 함께 저장한다.

모든 노드들은 인증서를 발행할 수 있다. 인증서 발행은 스마트 계약을 이용하며 인증서 발급에 대한 스마트 계약은 제네시스 블록 배포시 같이 배포된다. 제어계층의 기기는 전달받은 인증서를 랜덤으로 노드를 선택하여 검증을 요청할 수 있다. 전달받은 제어 명령이 제어기기에 미치는 영향도에 따라 여러 운영 계층의 기기에게 인증서 검증을 요청 할 수 있다.

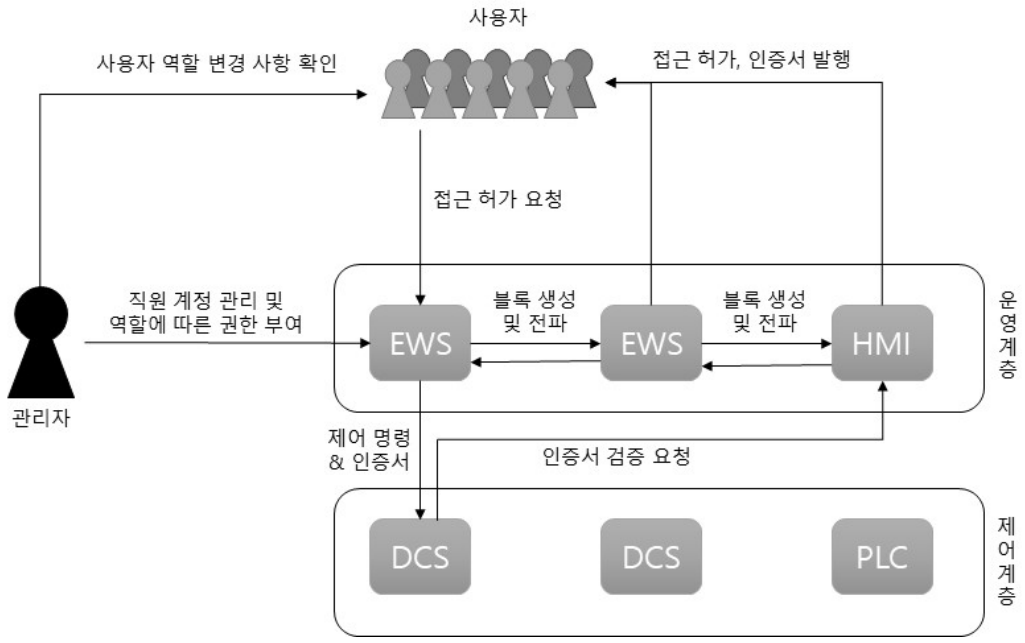


그림 1 시스템 체계

운영계층 기기에 저장된 공개키는 인증서 발행 시 사용한다. 인증발급서버의 개인키는 노드의 개인키이다.

2.2 트랜잭션

제안하는 시스템에서 발행하는 트랜잭션은 사용자 등록 및 변경 , 역할에 따른 권한 등록 및 삭제가 있다. 변경은 삭제 또한 포함한다.

사용자 등록 및 변경은 관리자의 권한이며 관리자 권한은 그림2과 같이 할당 가능하다. 본 논문에서 관리자는 사용자를 관리하는 사람을 의미한다.

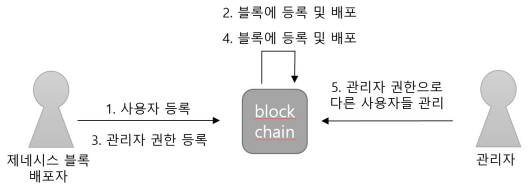


그림 2 관리자 권한 등록 과정

제네시스 블록 배포자는 사용자의 권한을 등록 및 변경할 수 있는 관리자를 등록할 수 있다. 관리자 권한은 제네시스 블록 배포자가 등록한 사용자만 할당받을 수 있다. 사용자는 관리자 권한을 등록하는 트랜잭션이 블록이 배포된 후에 관리자 권한을 사용가능하다.

관리자는 사용자를 등록 할 수 있다. 첫 등록 시에는 블록에 등록 및 배포되면서 블록에 공개키가 등록되고 공개키와 개인키가 사용자에게 전달된다. 전달은 전자메일과 같이 사용자 등록시 입력된 정보로 전달 가능하다.

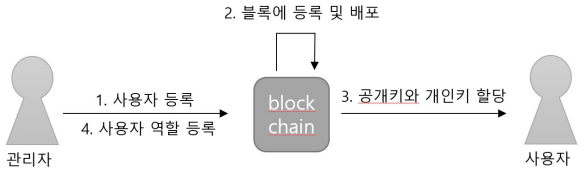


그림 3 사용자 등록 과정

사용자가 등록되고 난 후에 사용자 역할을 등록할 수 있다. 사용자 계정이 배포되고 난 후 역할을 등록하는 것은 등록되지 않은 사용자에게 역할과 역할에 따른 권한이 할당되는 것을 방지하기 위해서이다.

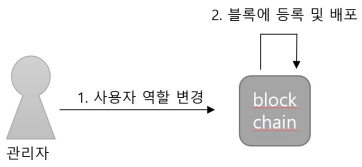


그림 4 사용자 역할 변경 과정

관리자는 사용자의 역할을 변경할 수 있으며 변경된 역할은 블록에 등록 및 배포하여 즉시 사용자의 권한이 변동된다.

사용자 계정의 폐기 역시 사용자 역할 변경과 같은 순서로 이루어진다.

VI. 결론

본 논문에서는 산업제어시스템에서 블록체인을 이용하여 사용자 계정 및 권한을 관리하였다. 저장된 사용자 계정 및 권한을 이용하여 산업제어시스템에서 운영계층에서 제어계층으로 접근할 때 사용할 수 있는 인증서를 발급할 수 있다.

산업제어시스템에서 계정의 관리는 중요한 요소이다. 산업제어시스템을 위한 표준인 NIST 800-82 에서 접근 제어의 복잡성을 줄이기 위한 방법으로 RBAC으로 소개되었다[9]. RBAC은 사용자의 역할 및 권한 정의에 따라 접근 권한이 다르므로 사용자의 역할 및 권한에 대한 변조가 불가능해야하며 변경사항이 즉시 적용되어야 한다. 본 논문에서 제안하는 시스템에서는 역할 및 권한 관리를 블록체인으로 하여 한 시스템이 해킹되어 변조되더라도 다른 노드를 통해 확인이 가능하다.

산업제어시스템에서 사용자의 인증을 하는 방법 중 자체 인증서를 발급하는 방법이 있다. 그러나 자체 인증서 발급 서버의 경우 안전성 및 신뢰성을 확인 할 수 없다. 본 논문에서 제안하는 시스템에서는 랜덤으로 인증서 검증을 요청하여 공격자가 대부분의 운영계층을 해킹하지 않는 이상 공격이 어렵다. 랜덤으로 하나 노드에만 검증을 요청할 수도 있지만 전달된 명령의 중요도에 따라 여러 노드에게 검증을 요청하게 할 수 있다.

산업제어시스템은 시스템 도입 후 장기적인 운영을 하므로 기기의 추가 및 삭제 등의 변경이 어려우나 제안하는 시스템에서는 소프트웨어의 업데이트만으로 사용이 가능하다.

[참고문헌]

[1] 오준형, 유영인, 이경호. 기반시설 침해사고 및 제어시스템 표준 동향. 정보보호학회지, 27(2), 5-11, 2017.

[2] Stouffer, K., Falco, J and Scarfone, K. Guide to industrial control systems (ICS) security. NIST special publication, 800(82), 16-16, 2011.

[3] 조민정, 석병진, 김역, 이창훈. SDN 기반 산업제어시스템 제어명령 판별 메커니즘. 한국디지털콘텐츠학회 논문지, 19(6), 1185-1195, 2018.

[4] NAKAMOTO, Satoshi. Bitcoin: A peer-to-peer electronic cash system. 2008.

[5] 김광훈. 블록체인 기술의 이해 및 적용 현황. ie 매거진, 25(1), 13-19, 2018.

[6] Cruz, J. P., Kaji, Y., and Yanai, N. RBAC-SC: Role-Based Access Control Using Smart Contract. IEEE Access, 6, 12240-12251, 2018 .

[7] Ouaddah, A., Abou Elkalam, A., and Ait Ouahman, A. FairAccess: a new blockchain based access control framework for the Internet of Things. Security and Communication Networks, 9(18), 5943-5964, 2016.

[8] Maesa, D. D. F., Mori, P., and Ricci, L , blockchain based access control. In IFIP International Conference on Distributed Applications and Interoperable Systems pp. 206-220. Springer, Cham, 2017, June.

[9] Stouffer, K. A., Falco, J. A., and Scarfone, K. A.. Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc), 2011

국내 암호화폐 거래소 보안위협 및 대응방안 제시

박현성, 서정택

순천향대학교 정보보호학과

A Proposal of Security Threats and Countermeasures for the Internal Cryptocurrency Exchange

Hyeon-Seong Park, Jung-Teak Seo

Division of information security, Soonchunhyang University.

요 약

최근 암호화폐에 대한 관심과 거래가 증가함에 따라 암호화폐 거래소가 급증하였다. 탈중앙화 분산장부 기술인 블록체인을 활용한 암호화폐의 공급과 수요가 나날이 증가하고 있으며 그에 따라 화폐와 가상화폐를 거래해주는 암호화폐 거래소의 거래량이 증가하였다. 점차 암호화폐 거래소를 목표로 하는 사이버 공격이 증가하였고 더불어 보안의 중요성이 나날이 커지고 있다. 하지만 국내 거래소에서는 거래 서비스의 보안적 측면을 고려하지 않았으며, 이로 인한 거래소 대상 사이버 공격이 주기적으로 발생하였고 큰 피해를 입는 사고가 발생하였다. 따라서 본 논문에서는 암호화폐 거래소의 사이버 공격 사례와 공격시나리오를 바탕으로 사이버 공격에 대한 대응방안을 제시한다.

I. 서론

국내 암호화폐 거래소는 2013년 '코빗'을 시작으로 현재의 주요 암호화폐 거래소인 고팍스, 빗썸, 업비트, 코빗, 코인원 등 약 20여개의 거래소가 설립되어 운영 중이다[1]. 그에 따라 세계 1위의 암호화폐의 거래량을 달성했으며 약 2조원의 거래가 하루 평균 발생하고 있다.

암호화폐 거래소(Cryptocurrency exchange)는 암호화폐와 화폐를 환전해주는 거래소로 달러나 엔화를 원화로 환전해주는 외환 거래소와 비슷한 역할을 한다.

하지만 이러한 거래소들은 해커들의 타겟이 되어오고 있으며, 거래소를 통한 거래가 매우 활발하게 이루어지고 있고 금융과 관련된 분야이기 때문에 이에 따른 피해 규모가 심각하다고 볼 수 있다[2]. 더욱이 사용자의 개인정보와 계좌정보, 암호화폐 보유량 등 개인정보와 금전정보를 수집하고 관리하고 있기 때문에 막대한 경제적 피해와 2차 피해까지 이어질 수 있다. PC와 모바일을 통한 암호화폐 거래가 활발하게 이루어지는 만큼 정부가 앞장서서 보안 강화를 요구하고 있으나 계속해서 피해가 속출하고 있다.

본 논문에서는 과거 국내 암호화폐 해킹 사례를 통해 국내 거래소 보안위협 및 공격 시나리오를 도출하고, 그에 따른 대응방안을 제시하여 국내 암호화폐 거래소 보안을 강화하는데 일조할 수 있을 것으로 사료된다.

본 논문의 구성은 다음과 같다. 2장에서 국내 암호화폐 거래소의 해킹 사례들을 살펴보고, 3장에서 국내 암호화폐 거래소 보안 위협에 대해 분석하며 4장에서 국내 암호화폐 거래소 사이버 공격 시나리오를 도출한다. 5장에서는 국내 암호화폐 거래소 사이버 공격 대응방안과 마지막으로 6장의 결론으로 논문을 마무리한다.

II. 국내 암호화폐 거래소 해킹 사례

국내 암호화폐 거래소를 대상으로 한 주요 사이버 공격 사례는 [표 2]로, 매년 수십억 가량의 경제적 피해가 발생하였다.

발생일시	피해내역
16.07.26.	3억 원 상당의 가상통화 부정인출
17.04.22.	55억 원 상당의 가상통화 부정인출
17.06.28.	70억 원 상당의 가상통화 부정인출 약 3.6만여 건 개인정보 유출
17.09.23.	21억 원 상당의 가상통화 부정인출
17.12.19.	259억 원 상당의 가상통화 부정인출
18.06.10	530억 원 상당의 가상통화 부정인출
18.06.19	350억 원 상당의 가상통화 부정인출

[표 2] 암호화폐 거래소 해킹 사례(경찰청)

2.2 암호화폐 거래소 '빗썸' 2차례 해킹

이메일 피싱을 통한 악성코드로 업무용 PC가 감염되었고, 고객정보가 저장되어 있는 데이터베이스를 목적으로한 공격으로 인해 약 3만 6천여건의 고객정보가 유출되었다[3]. 해커에게 유출된 고객정보는 접속 인증 우회 및 고객에게 2차 피싱등을 통하여 계정에 접근할 수 있었고, 본인 계좌가 아닌 다른 계좌로 인출되는 사고가 발생하였다[4].

1차 해킹을 통해 개인정보 유출과 부정인출로 인한 피해가 발생한 후, 해당 업체는 보안 솔루션을 도입하여 보안을 강화하였다. 그럼에도 불구하고 2018년 6월, 350억 원의 암호화폐의 부정인출과 금전적 피해가 발생하였다.

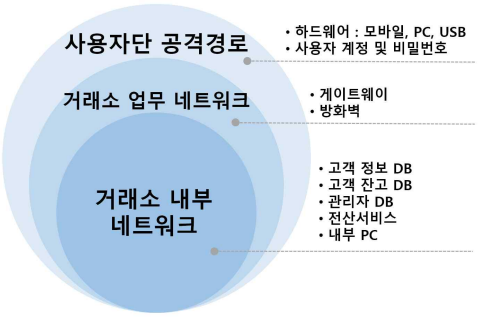
2.3 국내 암호화폐 거래소 보안 점검 결과

A사	· 망 분리 및 시스템 접근통제 관리 미흡 · 가상통화 지갑관리 미흡(보안정책 운영 등) · 정보보호시스템(방화벽 등) 구축 미흡 · 관리자 PC 설치프로그램에 대한 보안관리(유해사이트 접속 차단 등) 미흡
B사	· 망 분리 및 시스템 접근통제 관리 미흡 · 관리자 PC 설치프로그램에 대한 보안관리(유해사이트 접속 차단 등) 미흡 · 외부 개인PC에서 인터넷망을 통해 내부시스템 접근 기능 등 보안사고 우려
C사	· 망 분리 및 시스템 접근통제 관리 미흡 · 가상통화 지갑관리 미흡(보안정책 운영 등) · 정보보호시스템(방화벽 등) 구축 미흡 · 침해사고 대응 절차, 지침 등 미흡 · 주요데이터 백업관리 체계 미흡
D사	· 망 분리 및 시스템 접근통제 관리 미흡 · 정보보호시스템(방화벽 등) 구축 미흡 · 주요데이터 백업관리 체계 미흡
E사	· 망 분리 및 시스템 접근통제 관리 미흡 · 가상통화 지갑관리 미흡(보안정책 운영 등) · 정보보호시스템(방화벽 등) 구축 미흡 · 주요데이터 백업관리 체계 미흡
F사	· 망 분리 및 시스템 접근통제 관리 미흡 · 침해사고 대응 절차, 지침 등 미흡
G사	· 방화벽 등 기본적인 보안시스템은 있으나, 보안정책관리 등 미흡
H사	· 망 분리 및 시스템 접근통제 관리 미흡 · 침해사고 대응 절차, 지침 등 미흡 · 정보보호시스템(방화벽 등) 구축 미흡 · 주요데이터 백업관리 체계 미흡
I사 (폐업)	· 계정관리정책 수립이 되어있지 않음 · 관리자PC가 지정되어 있지 않으며, 노트북과 무선 인터넷을 사용하여 관리수행(망 분리 권고)
J사	· 관리자 PC 설치프로그램에 대한 보안관리(유해사이트 접속 차단 등) 미흡 · 침해사고 대응 절차, 지침 등 미흡

[표 3] 암호화폐 거래소 보안 점검 결과
(과학기술정보통신부, 한국인터넷진흥원)

III. 국내 암호화폐 거래소 보안 위협

급격한 사용자 증가에 따라 확대된 사이버 공격동기 및 공격벡터를 통한 내부 네트워크 불법 침투의 가능성이 증가하였다[5]. 보안 설계가 미흡한 내부 네트워크 특성은 백도어, 스푸핑 및 위·변조 데이터 주입 등 공격 탐지의 취약점이 노출되어 있다.



[그림 1] 국내 암호화폐 거래소 구성(예시)

3.1 사용자단 보안위협

거래소를 이용하는 사용자의 모바일 및 PC는 대중화되어 외부 인터넷에 노출되어 있으므로 비밀번호 유출, 인증서 유출, 펌웨어 감염 및 악성 어플리케이션 설치 등 다양한 공격에 취약하다.

거래소 SW 및 거래소 웹사이트의 취약점으로 인한 보안 위협이 존재한다. 또한 사용자에게 페이크 사이트(Fake Site)의 접속을 유도하는 공격도 가능하다.

3.2 거래소 업무 네트워크 보안위협

외부 인터넷에 연결된 업무용 PC의 사용으로 인한 악성코드의 유입이 가능하다. 방화벽의 접근제어 취약점을 이용하여 정상적인 경로를 통해 공격이 이루어질 수 있다. 또한 주요 전산서버와 연결되어 있는 경우 SQL Injection등과 같은 취약점을 이용하여 데이터베이스로의 접근이 가능하다.

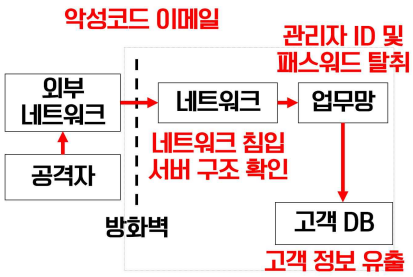
3.3 거래소 내부 네트워크 보안위협

업무망과 내부망의 망 분리 및 주요 자산 시스템에 대한 접근통제 관리 미흡으로 인한 외부 인터넷 연결을 통한 공격이 가능하다.

IV. 국내 암호화폐 거래소 사이버 공격 시나리오

4.1 개인정보 유출 시나리오

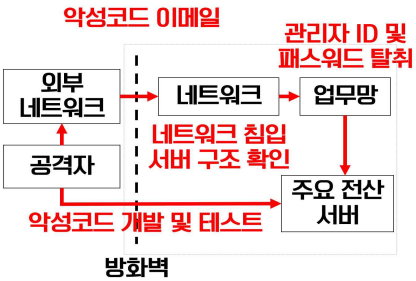
업무용 메신저 주소의 유출이나 직원 계정 탈취를 통하여 악성코드 프로그램이 담긴 이메일을 통한 스피어피싱이 발생한다. 거래소 업무망 내 PC에 접근한 공격자는 네트워크 스캔을 통해 서버 구조를 확인하고 관리자 ID 및 패스워드를 탈취한다. 관리자 계정으로 접속 가능한 고객의 개인정보가 저장된 데이터베이스에서 개인정보를 유출한다.



[그림 2] 개인정보 유출 시나리오 다이어그램

4.2 거래소 서버 장악 시나리오

거래소의 업무망과 전산망(내부망)이 연결되어 있는 PC를 찾기 위해 네트워크 스캔을 통해 해당 구간을 탐색한다. 관리자 권한을 가진 PC를 통해 주요 전산 서버를 장악하고 해당 공격 벡터와 C&C서버를 연결하여 내부망 내의 PC에서 악성코드를 개발하거나 테스트한다. 이 과정에서 업무망의 모든 네트워크 패킷들이 유출된다.

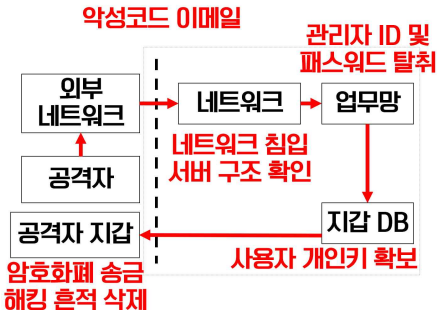


[그림 3] 서버 장악 시나리오 다이어그램

4.1 비대칭키 유출 및 지갑 탈취 시나리오

고객의 계좌에 해당하는 지갑정보가 저장되어 있는 데이터베이스에 접근하여 고객의 개인키를 확보하게 되면, 마지막으로 공격자는 자신의 암호화폐 지갑에 암호화폐를 송신하여 개인자산을 유출한다. 이와 같은 과정이 모두 끝나

면 자신의 해킹 흔적을 은닉하거나 삭제하여 해킹의 흔적이나 유무를 쉽게 발견할 수 없도록 한다.



[그림 4] 비대칭키 유출 및 지갑 탈취 시나리오 다이어그램

V. 국내 암호화폐 거래소 사이버 공격 대응 방안

5.1 망 분리

업무망과 내부망의 망 분리를 통해 고의적이거나 비고의적인 실수와 오류로부터 주요 자산을 보호하고 체계적인 접근제어가 필요하다. 더욱이 자산의 식별과 관리가 효율적으로 이루어질 수 있도록 망 분리 시 고려해야하는 부분과 함께 망 분리가 이루어져야한다.

5.2 주요 데이터 암호화

주요 자산 및 데이터의 접근이나 사용함에 있어서 데이터를 암호화시킴으로써 높은 레벨을 통한 접근권한에서의 자산 관리가 필요하다. 특히 고객 개인정보와 같은 경우, 사용되지 않고 있는 계정의 개인정보는 암호화되어 저장되거나 암호화 백업을 통해 보호되어야한다.

5.3 구간별 보안 강화

주요 데이터의 가용성과 무결성의 침해 위협으로부터 보호하거나 유출되는 정보의 기밀성을 유지하기 위한 보안 기술이나 장치를 추가해야한다.

5.4 ISMS 및 ISMS-P 인증

정보보호 및 개인정보보호를 위한 일련의 조치와 활동이 인증기준에 적합함을 인터넷진흥원 또는 인증기관이 증명하는 제도로 대부분의 국내 거래소는 인증 의무 대상에 포함되어 있다. 이에 따라 적합한 보안 관리체계 수립 및 운영과 보안 기술 적용, 개인정보의 보호조치가 이루어져야한다.

VI. 결론

본 논문에서는 국내 암호화폐 거래소에 대한 보안 위협과 공격 시나리오를 도출하였다. 또 그에 따른 대응 방안을 제시하였다. 가상화폐와 화폐를 교환하고 거래하는 서비스의 보안은 금융서비스의 보안만큼 중요성이 크다. 더불어 개인정보를 통한 거래가 이루어지기 때문에 악의적인 해커에 의해 발생하는 해킹이 매우 큰 금전적 피해를 가져올 수 있어 심각성이 크다고 할 수 있다. 이미 국내 암호화폐 거래소의 시스템 구성의 취약점을 이용하여 과거 수차례의 해킹이 발생하였고, 그에 따른 보안 조치가 이루어지지 않는다면 앞으로도 큰 피해가 발생할 것으로 판단된다. 이러한 공격은 거래소의 보안 관리가 금융서비스와 유사한 방식의 적절한 보안 조치가 필요할 것으로 보인다. 그렇기 때문에 거래소 시스템 구성에 있어서 구조적인 취약점을 방어할 방법을 연구하고 강화해나가야 할 것이다.

[참고문헌]

[1] 송지환, "비트코인 거래소, 해커와 절도범으로부터 안전한가", 월간SW중심사회 2018년 10월호, 소프트웨어정책연구소, pp.36-43

[2] Financial Security Institute , "BlockChain Technology and Security Considerations", August, 2017.

[3] 한국인터넷진흥원, 2017년 3분기 사이버 위협 동향 보고서, 2017

[4] igloosecurity, Monthly Security Report, Feb. 2018

[5] 정용식, 차재상. (2018). 블록체인 기반 가상화폐 거래의 보안 위험 및 대응방안. 한국정보전자통신기술학회 논문지, 11(1), 100-106.

블록체인을 이용한 전자투표 장단점 분석*)

홍지형, 권희원, 김윤정
서울여자대학교 정보보호학과

An Analysis on the Strengths and Weaknesses of Blockchain Electronic Voting

Jihyung Hong, Heewon Kwon, Yoonjeong Kim
Department of Information Security, Seoul Women's University

요 약

블록체인 온라인 투표 시스템에서는, 투표 내용이 블록체인 분산 원장에 기록되며, 투표자들은 자신의 투표 트랜잭션 및 투표 진행상황을 확인할 수도 있다. 장소와 시간에 크게 구애받지 않으며, 투명한 투표를 진행할 수 있다는 점에서 국내외로 블록체인 전자투표 시스템이 적용되고 있다. 그러나 네트워크 상에서 이루어지기 때문에 여전히 전자투표에 대하여 공격 가능성, 기밀성 훼손 등 다양한 문제점도 제기되고 있다. 본 논문에서는 블록체인 전자투표 시스템의 사례들을 분석하여 각 사례별로 장점과 단점을 분류하여 개선방안의 기반을 제안한다.

I. 서론

익명의 개인 혹은 집단인 사토시 나카모토(Satoshi Nakamoto)가 2008년에 제안한 비트코인[1]은 혁신적인 전자화폐로써 크게 각광받았다. 블록체인은 비트코인을 비롯한 가상화폐의 핵심 기술로 활용되고 있는데 가상화폐 뿐만 아니라 점차 확대되어 금융, 물류시스템, 의료, 에너지, 선거 등 다양한 분야에서 시범 및 활용되어지고 있다.

본 논문에서는 블록체인 온라인 투표시스템의 사례를 살피고 제안된 투표시스템의 장단점을 구분하여 논한다. 2장에서는 블록체인 온라인 투표시스템과 장단점을 설명한다. 3장에서는 블록체인 온라인 투표시스템 관련 기존연구를 조사한 내용을 기술하고, 4장에서는 기존 연구의 장단점을 세분화하여 기술한다. 마지막으로 5장에서는 결론을 맺는다.

*) 본 연구는 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(No. NRF-2016R1A2B4011984)과 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2014-1-00636)

II. 블록체인 온라인 투표 시스템

온라인 투표시스템은 일정한 규칙을 준수하여 안전하게 운영되어야 하는데, 완전성, 건전성, 비밀성, 이중투표불가성, 적임성, 공정성, 검증성, 매표방지성, 강인성 등이 전자투표 시스템에 요구된다 [2]. 전자투표는 일반 종이투표와는 달리 장소와 시간에 크게 구애받지 않을 수 있다는 것이 큰 장점이다. 집계 결과 역시 투표가 마감하는 시간으로부터 개표가 완료될 때까지 오랜 시간 기다리지 않아도 신속하게 확인할 수 있다. 또한 선거를 위해 투표 및 개표 장소를 따로 마련하지 않아도 되며, 인력이 동원될 필요도 없어 비용을 크게 절감할 수 있다. 전자투표의 단점으로는, 네트워크를 통해 투표 및 검증을 받게 되는데 이 때 유권자들의 투표 내용, 개인정보 등을 검증하는 기관의 서버가 공격당할 수 있는 가능성이 있다. 또한 투표의 경우 비밀선거의 원칙하에 이루어져야 하나, 전자투표의 경우 투표의 전 과정을 모니터링 할

수 있는 장소와 인력 없이 자유롭게 투표권을 행사할 수 있다는 점을 이용하여 매수 매표의 가능성 또한 매우 크다. 블록체인을 이용한 전자투표도 이와 같은 일반적인 온라인 전자투표의 특성을 갖는다.

한편, 일반적인 전자투표의 경우 유권자의 정보 및 투표내용을 저장할 데이터베이스가 필요하다. 전자투표 시스템에서, 투표자들은 투표 관련 정보가 데이터베이스에 안전하게 저장되는지와, 투표가 끝나고 일정기간 이후에 데이터베이스에 저장된 내용이 완전하게 파기됐는지 알 수 없다. 반면, 블록체인 온라인투표의 경우 참여자 모두가 투표결과를 검증하고 블록체인에 저장된 내용을 동일하게 가지고 있기 때문에 중간에 투표 내용을 조작하기가 어려우며, 기존 전자투표 상 데이터베이스가 하는 역할을 투표자 모두의 분산 원장이 하므로 투명성이 적용된다.

III. 블록체인 온라인 투표 사례연구

3.1 토큰을 이용한 블록체인 전자투표 시스템

노창현 등은 토큰을 이용한 블록체인 전자투표 시스템을 제안하였다 [4]. 해당 전자투표 시스템은 크게 등록-투표-개표 세 단계로 구분되어 있다. 각 서버가 하는 역할이 세부적으로 나뉘어져 있으며, 키와 지갑을 생성 및 전파하고 후보자, 투표자 리스트를 각각 마련해 전송하면 투표자 수만큼 토큰을 생성하여 유효 투표자들의 지갑 주소로 1 토큰씩 전송한다. 투표자는 후보자를 선택해 후보자 정보를 생성하고, 투표 시간과 함께 개표 서버의 공개키로 암호화 한다. 투표서버는 전송받은 트랜잭션과 연접 데이터를 수집해 블록을 생성하고 생성한 블록은 모든 투표 서버에 전파해 검증을 진행한 후 블록체인에 연결되게 한다. 투표가 종료된 후 블록을 받은 개표 서버는 블록에 저장된 암호화된 연접 데이터를 복호화하여 개표를 진행하고 개표가 완료되면 공표한다.

3.2 가상화폐 거래과정을 이용한 블록체인 전자투표 시스템

양준호 등은 가상화폐 거래과정을 이용한 블록체인 전자투표 시스템을 연구하였다 [5].

전자투표 시스템은 Korbit에서 제공하는 가상화폐 제작 API를 활용하여 투표용 코인을 제작했고, 서버는 Node Js를 활용했다. 유권자에 대한 데이터베이스는 Node Js와 호환이 잘 되는 Maria DB를 이용했다. 투표 준비 단계에서는 시스템 관리자가 투표 코인을 생성하고 서버에 유권자 리스트를 전달한다. 투표 진행 단계에서는 유권자 인증이 완료되면 서버는 유권자에게 투표 관련 정보를 제공하고, 투표 코인을 배부한다. 이후 유권자가 코인을 사용하여 투표를 진행한다. 투표 종료 단계에서는 서버가 시스템 관리자에게 투표 종료를 알리면 시스템 관리자는 이를 승인 후에 투표 코인과 인증 소스를 폐기한다. 이후 서버는 유권자에게 투표 관련 정보를 제공한다.

3.3 이더리움 블록체인의 스마트 컨트랙트 방식을 활용한 전자투표 시스템

성기정 등은 이더리움 블록체인의 스마트 컨트랙트 방식의 전자투표 시스템을 연구하였다 [6]. 전자투표 시스템은 이더리움 블록체인의 스마트 컨트랙트 형태로 동작한다. 사용자 인증은 먼저 유권자가 모바일 앱에 자신의 학번 및 비밀번호를 입력하면 해당 정보가 해싱되어 인증기관으로 전송된다. 인증기관에서는 해싱된 개인정보와 교내 DB에 저장된 학생의 신원 정보가 일치하면 해당 학생의 휴대전화로 OTP (One Time Password) 인증을 요청하고 받는다. 유권자는 인증기관의 인증 성공 응답을 받았는지를 전단부에서 안전한 방식으로 검증하며 이후 랜덤한 seed를 생성하고 이로부터 개인키 및 공개키와 이더리움 블록체인의 주소를 생성하며 이는 자동적으로 전단부의 인증과정 후 스마트 컨트랙트에 등록된다. 전자투표 시스템의 후단부에서는 이더리움 기반의 솔리디티 언어를 사용하여 스마트 컨트랙트를 구현한다.

3.4 Public Blockchain을 활용한 블록체인 전자투표 시스템

하현수 등은 Public Blockchain을 활용한 블록체인 전자투표 시스템에 대하여 제안하였다 [7]. 투표 단말기는 투표자의 생체인증을 바탕으로 구성된 Unique Data를 이용하여 전자 서명에 사용될 키를 생성한다. 그 후, 유권자가 전자투표를 진행하게 되면 투표 정보를 담은 트랜잭션이 생성되고 트랜잭션의 내역이 공개되는 Public Blockchain의 특성에 의해 투표자는 자신의 트랜잭션 내역 조회 및 개표 결과를 실시간으로 확인할 수 있다. 하지만 Public Blockchain을 이용할 경우, 자신의 트랜잭션 정보가 공개되어 유권자가 어떤 후보에게 투표하였는지가 노출될 수 있다. 따라서 스텔스 주소 기법을 사용하여 트랜잭션의 목적지인 후보자

(Receiver) 측의 주소를 감춘다. 그 후 트랜잭션의 전자서명을 링 형태로 융합시키는 Ring Signature 기법을 통해서 해당 트랜잭션의 투표자의 익명성을 보장했다. 제안된 트랜잭션 구조는 ECDH(Elliptic Curve Diffie Hellman) 암호기법을 통해 해싱한다. 그 후 이렇게 만들어진 트랜잭션들은 더미로 사용되는 다른 트랜잭션들과 함께 Ring 형태로 묶어 전송한다. Ring 형태로 전송된 트랜잭션은 송신자를 특정할 수 없게 해주며 유권자의 신원을 익명화 시킬 수 있다. 따라서 제안하는 방식에서는 Public Blockchain의 투명성과 검증성을 보장하면서도, Private Blockchain 이상의 익명성을 제공할 수 있다.

3.5 Private 블록체인 기반 전자투표 시스템

권용석 등은 Private 블록체인을 활용한 블록체인 전자투표 시스템을 제안하였다 [8]. Private 블록체인 기반 전자투표 시스템은 투표자 인증과 투표 여부에 관련된 검증 및 합의를 진행하는 Auth Blockchain과 투표 값에 관련된 검증 및 합의를 진행하는 Vote Blockchain을 연결한 이중 블록체인으로 구성된다. 모든 블록체인은 내부 노드 간의 통신과, 블록체인 간의 통신, 블록체인과 투표자 간의 통신은 비대칭키 알고리즘을 통해 암호화되며, 투표자가 실제로 투표하는 경우에는 해당 데이터가 분산형 네트워크 기반의 익명 인터넷 통신 시스템인 Tor 방식을 이용하여 전송된다. 본 시스템은 이중 블록체인의 구성으로 비밀선거, 평등선거의 원칙을 보장한다.

3.6 블라인드 서명을 이용한 블록체인 전자투표 시스템

Yi Liu 등은 Trusted Third Party(TTP)가 존재하지 않으며 블록체인 기술과 블라인드 서명기술을 적용한 전자투표 시스템을 제안하였다 [9]. 투표자인 Alice(A), 투표자가 정당한 투표자인지 검증하고 투표를 조직하는 조직자 Bob(B), 조직자의 과도한 역할을 제한하고 감독하며 투표자와 연관이 있는 감독자 Carol(C)이 있다. A는 자신이 정당한 투표자임을 검증받고자 메시지를 공개키로 서명한 후 B에게 보내면, B는 자신의 개인키로 서명하여 A에게 검증완료의 의미인 서명된 메시지를 재전송한다. A는 C에게도 동일하게 자신을 검증받고자 메시지를 보내고 C의 개인키로 서명된 메시지를 받는다. 블라인드 서명으로 인해 B와 C는 A가 보낸 메시지의 내용을 알 수 없다. B와 C의 키로 서명된 메시지를 모두 가지고 있는 A는, 투표 조직자인 B에게 본인의 투표 내용과 함께 B와 C에게 검증받은 내역까지 함께 보내게 되어 정상적으로 투표를 마친다. 조직자 B는 투표자들이 보낸 투표 내용을 수집했다가, 개표시기에

블록체인 이용 투표시스템	장점	단점
토큰을 이용한 블록체인 전자투표 시스템 [4]	- 서버 및 역할 구분이 뚜렷해 서버 별 관리 용이 - 토큰을 사용해 위조 방지	- 많은 서버 구축 비용 - 다량의 서버 간 데이터 이용
가상화폐 거래과정을 이용한 블록체인 전자투표 시스템 [5]	- 블록체인 서버 구축 비용 절감 - 데이터 이동이 덜 해 빠른 투표처리 가능	- 메일링을 통한 단순한 신원 인증
이더리움 블록체인의 스마트 컨트랙트 방식을 활용한 전자투표 시스템 [6]	- 강압에 의한 매수매표 방지를 위해 재투표 기능 구현	- 투표자 수가 많은 경우는 고려되지 않음 - 단순한 신원인증 과정으로 인한 공격가능성
Public Blockchain을 활용한 블록체인 전자투표 시스템 [7]	- public blockchain을 적용하여 참여자 전원이 블록체인의 모든 트랜잭션 내역을 열람 가능	- Tor의 데이터 릴레이로 인한 속도저하 - 첫 번째 및 마지막 노드 공격가능성
Private 블록체인 기반 전자투표 시스템 [8]	- 인증 및 투표자 정보 저장하는 블록체인과 투표 저장 블록체인을 분리 관리	- 투표 전 과정을 검증할 수 없음 - Tor 속도저하 및 공격가능성 문제
블라인드 서명을 이용한 블록체인 전자투표 시스템 [9]	- Trusted Third Party(TTP)없이 투명한 블록체인 투표 가능 - 유권자의 익명성을 보장	- coercion-resistance를 만족하지 못함

표 1. 블록체인 온라인 투표시스템 장단점 분석

노드들에게 전파하고 투표 내용을 모든 참여자가 검증한다.

IV. 블록체인 온라인 투표시스템 장단점 분석

노창현 등의 사례는 투표, 개표, 관리 단계의 서버가 각각 나뉘어져 있어 서버 별 관리가 용이하다. 그러나 서버가 세분화되어 있기 때문에 서버 구축비용이 많이 들 수 있으며 서버 간에 오가는 데이터의 양이 많아 투표자의 수가 많아지는 경우 이로 인해 문제가 될 수 있다. 양준호 등은 소규모 온라인 투표 사례로 많은 서버를 필요하지 않아 비용적인 측면에서 절감될 수 있으나, 소규모 투표의 경우 기재하는 회원번호의 수가 제한적이거나 타인이 알 수 있어 비밀성이 깨질 수 있다. 예시로 학번을 회원번호로 기재하고 이름, 메일로 받은 개인 인증번호 총 3가지 정보를 기재함으로써 인증을 받게 되어 있는데, 인증과정이 단순하기 때문에 지인이나 부정직한 개인정보관리자의 경우 학번, 이름을 손쉽게 알 수 있기 때문에 이로 인하여 부정투표가 일어날 수 있다. 성기정 등은 강압에 의한 매수매표 방지를 위해 재투표 기능을 구현하였으나 교내 전자투표와 같은 소규모 투표만 고려되지 않았다. 또한 단순한 신원인증 과정 때문에 공격가능성을 가지고 있다. 하현수 등은 Public 블록체인을 적용해 참여자 전원이 투표 과정을 확인할 수 있다는 장점을 가지고 있으나 Tor 네트워크는 데이터의 릴레이로 인해 속도가 현저히 느려질 수 있다는 단점을 가지고 있다. 또한 Tor 네트워크의 우회과정 중에 첫 번째 노드와 마지막 노드의 경우 공격으로 인해 정보가 침해될 수 있는 가능성을 가지고 있다. 권용석 등은 투표자의 정보를 저장하고 인증하는 Auth blockchain, 투표 내용을 저장하는 Vote blockchain이 분리되어 있으나 private 블록체인은 투표 전 과정을 모든 참여자들이 검증하고 확인할 수 없어 검증성이 미달되고 Tor 네트워크 방식의 속도저하 및 공격 가능성 역시 가지고 있다. 마지막으로 Yi Liu 등은 중간 관리자의 역할인 TTP가 없이 투명한 블록체인 투표가 가능하며 전자투표 요구사항을 준수하지만, 블록체인의 투명성으로 인하여 자신이 누구를 투표했는지 검증할 수 없어야하는 coercion-resistance를 만족하지 못한다는 단점이 있다.

V.결론

본 논문에서는 블록체인 전자투표 시스템의 기존 사례를 분석하고 각 사례별 장단점을 논하였다. 본 논문의 연구결과는 중앙 관리센터의 도움 없이 선거의 원칙을 지키며 선거가 수행될 수 있는 '블록체인을 이용한 전자투표 시스템'의 설계, 구현에 기여할 것으로 기대된다. 향후 세밀한 장단점 분석 및 발전된 블록체인 전자투표 시스템에 대한 연구를 진행할 계획이다.

[참고문헌]

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] 허원근, 김희선, 김광조, "전자선거 프로토콜의 요구사항 연구", 정보보호학회지, 10(1), 63-69, 2000.

[3] 권희원, 홍지형, 김윤정, "블록체인 전자투표 사례 분석", 2018 한국소프트웨어종합학술대회, 2018

[4] 노창현, 이임영, "블록체인 기반의 안전한 전자투표 시스템에 관한 연구", 한국정보보호학회 하계학술대회, 2018

[5] 양준호, 진민구, 이경희, 조정원, "블록체인기술 기반 캠퍼스 e-투표 시스템", 한국컴퓨터교육학회 학술발표대회논문집, 22(2), 67-70, 2018.

[6] 성기정, 정채린, 조은아, 이종호, 김희영, 김영우, 이경현, "블록체인 기반 교내 전자투표 시스템", 정보보호학회논문지, 28(4), 779-787, 2018.

[7] 하현수, 이선준, 정구익, 신용구, 김명호, 김영종, "Public Blockchain 기반의 익명성 전자투표 블록체인 플랫폼 모델", 한국정보과학회 학술발표논문집, 1176-1178, 2017.

[8] 권용석, 김수형, 정치영, 주민건, 김태성, 노재원, 조성현, "프라이빗 블록체인 기반 전자투표시스템 구현", 한국통신학회 학술대회논문집, 600-601, 2018.

[9] Liu, Y., Wang, Q., "An e-voting protocol based on blockchain", IACR Cryptol. ePrint Arch., Santa Barbara, CA, USA, Tech. Rep, 1043, 2017.

블록체인 기반 차량데이터 관리 시스템*

박승윤, 오희국*

*한양대학교 컴퓨터공학과

The Manamenet System of Vehicular Data Using Blockchain

Seung Yun Park*, Hee Kuck Oh†

*Department of Computer Science and Engineering, Hanyang University

요 약

중고차 시장의 규모가 커짐에 따라서 차량데이터의 중요성이 부각되고 있다. 하지만 국가적 차원에서 차량데이터를 관리하기 위한 수단이 있음에도 불구하고 차량데이터의 위변조 문제는 잦게 일어난다. 또한, 이러한 차량 데이터의 위변조 문제로 인해서 차량 데이터를 사용하는 시장 즉 중고차 시장에서의 악의적인 사기행각이 발생하고 있다. 본 논문에서는 이러한 차량 데이터 위변조 문제를 해결하고자 기존의 시스템에 블록체인 기술을 사용한다. 또한, 제안하는 시스템을 활용하여 중고차 시장에서 위변조된 차량데이터를 기반으로 이루어지는 사기행각 또한 줄어들 것으로 기대한다.

I. 서론

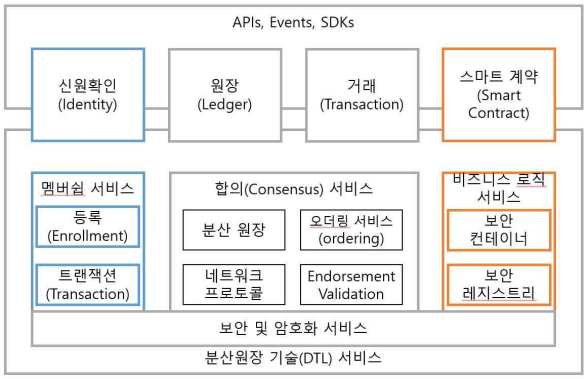
최근 차량의 품질 향상으로 내구성이 좋아지고 신차 가격에 부담을 느껴 중고차를 구입하는 소비자들이 늘어나면서 중고차 거래 관련 소비자 피해도 지속적으로 발생하고 있다. 2016년 말 기준 국내 자동차 등록대수는 총 2,180만대이고, 이 중 이전등록(중고차 거래) 대수는 378만대로 최근 5년간 중고차 거래는 연평균 3.6% 증가 추세에 있다.[1]

성장해가는 중고차 시장 규모에 비해, 중고차 시장에 대한 소비자들의 인식은 좋지 않다. 허위매물 또는 성능 상태 점검내용 상이와 같은 중고차 거래 사기행각이 발생하기 때문이다. 이에 따라 중고차 시장에서 차량 데이터의 중요성이 부각되고 있다. 본 논문에서는 블록체인을 활용하여 중고차 거래 시에 이해 당사자 간의 정보 대칭성을 이루며 신뢰성 있는 거래가 가능하게끔 하는 차량 데이터 위변조 방지 시스템을 제안한다.

본 논문의 1장에서는 각 시장에서의 차량데이터 관리시스템에 대한 전반적 현황에 대해 서술하며 2장에서는 중고차 시장에서의 차량데이터 위변조 문제가 발생하는 것에 대해 서술하고, 3장에서는 본 논문에서 사용되는 블록체인인 하이퍼레저 패브릭에 대해 서술하며, 4장에서는 본 논문에서 제안하는 차량데이터 위변조 방지 시스템에 대해 언급하며 5장에서는 제안하는 시스템의 활용 방안, 6장에서는 기대되는 효과와 7장, 8장에서는 앞으로의 추후 과제

* 본 연구는 미래창조과학부 및 정보통신 기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음(IITP-2017-2014-0-00636)
이 논문은 2015년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2015R1D1A1A09058200)

† 교신저자 : hkoh@hanyang.ac.kr



[Figur 1] 하이퍼레저 패브릭 V1.0 아키텍처

에 대해 언급하며 결론을 맺는다.

II. 중고차 시장에서의 차량데이터 위변조 문제

일반적으로 차량 데이터(주행거리, 사고 이력, 정비 이력, 보험 처리내역, 차량 등록, 이전, 연식 등)는 차량 등록증, 차량 성능점검기록부 시스템에 의해 관리된다. 중고차 거래 시에 딜러는 자동차관리법 제 58조 1항[2] (자동차매매업자가 자동차를 매도 또는 매매의 알선을 하는 경우에는 국토교통부령으로 정하는 바에 따라 다음 각호의 사항을 매매 계약을 체결하기 전에 그 자동차의 매수인에게 서면으로 고지하여야한다.)에 근거하여 차량의 성능상태를 알 수 있는 차량 성능점검기록부를 서면으로 고지한 후 중개를 해야 한다.

이러한 종래의 차량데이터 관리시스템은 지정된 성능 점검장을 통하기 때문에, 중고차 매매업자와 결탁을 해서 차량데이터를 기록하는 차량 성능점검기록부가 위조될 수 있다. 위조된 차량 성능점검기록부를 기반으로 중고차를 구매하는 소비자는 차량 성능점검기록부와는 다른 차량 데이터를 가진 중고차를 구매하여 피해를 입을 수 있다. 본 논문에서는 이와 같은 문제점을 해결하기 위해서, 블록체인 기반 신뢰성 있는 차량 데이터를 제공해 중고차 거래 간 이해 당사자들의 정보 대칭성을 이루고, 중고차 시장에 대한 신뢰성 회복을 기대한다.

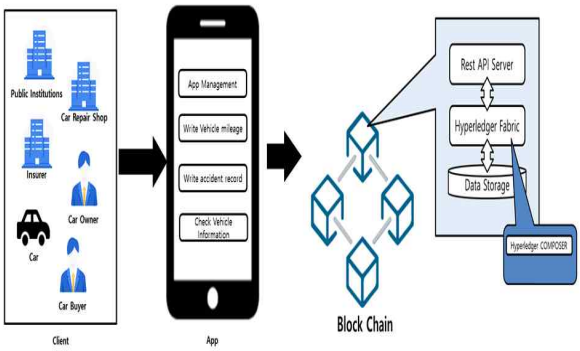
III.하이퍼레저 패브릭

하이퍼레저는 누구든지 참여할 수 있는 공개형(Public)과는 달리, 허가받은 노드들만 참여하는 기업형(Private) 블록체인으로 허가형 블록체인으로도 불린다.[3] 하이퍼레저에는 5개의 프레임이 존재하고 본 논문에서는 그 중 패브릭을 사용하고자 한다. 패브릭의 권한형 블록체인은 **[Figure 1]**와 같이 신원확인(Identity), 분산형 원장(Ledger)과 거래(transaction), 스마트계약(Chaincode)로 이루어져 잇다. 퍼블릭 블록체인과는 달리 패브릭에서는 관련 노드들만 트랜잭션을 조회할 수 있다. 패브릭 내의 노드는 각 역할과 접근 권한을 가지게 되며 이것은 채널 관리로 이루어진다.

패브릭의 체인 형성 과정은 다음과 같다. 사용자(Client)가 웹을 통해 원하는 트랜잭션을 관리하는 피어에 접속하면 각 피어의 스마트코드가 거래를 제안(Endorse)하고, 순서에 따라 실행(Excute)하고, 검증(Validate) 후 승인이 되면 체인에 기록하는 순서를 가진다.

IV.차량데이터 위변조 방지 시스템

앞선 1장에서의 문제점을 해결하고자 본 논문에서는 하이퍼레저 패브릭을 사용한다. 이 기술을 차량데이터 관리시스템에 도입하면 블록체인 네트워크에 참여한 각 노드들은 차량데이터를 체인 내에 기록할 수 있다. 데이터의 위변조가 불가능한 블록체인의 특성을 이용하여 차량데이터의 무결성을 유지한다. 기존의 차량데이터 관리 시스템의 문제점에서 차량데이터(주행거리, 사고 이력, 정비 이력, 보험처리내역 등) 중 주행거리는 차량 내 계기판을 통해 측정되고 사고이력, 정비이력, 보험처리내역 등은 정비업체, 보험사 등 관련 노드들이 기록하게 된다. 즉, 블록체인을 이용해서 체인에 차량데이터를 기록할 때에, 계기판을 조작하여 주행거리가 위조된 데이터가 기록되거나 사고이력에 대한 노드들 간의 상이한 차량 데이터 기록이 가능하다. 이러한 문제점을 해결하고자, 주행거



[Figur 2] 차량데이터 위변조 방지 시스템 아키텍처

리는 블록체인 서버와 연동된 차량시스템을 이용하여 실시간으로 주행거리를 전송하도록 한다. 전송된 주행거리 데이터는 계기판에 기록된 주행거리 데이터와는 달리 블록체인 장부에 기록되었기 때문에 쉽게 위변조가 불가능하다. 장부에 기록된 주행거리 데이터는 등록된 차량의 고유 ID를 통해 조회할 수 있다.

사고와 관련된 모든 데이터의 위변조는 차량 내 충격 감지 센서를 통한 데이터를 장부에 기록함으로서 해결할 수 있다. 센서를 통한 사고 데이터는 블록체인 장부의 트랜잭션을 발행시켜 사고가 발생되었을 때 보험처리를 하지 않는 사고의 누락을 방지하여 악의적으로 사고 이력을 숨기는 식의 방법을 방지한다. 또한, 각 노드들은 이에 따른 사고 처리 내역에 관해 트랜잭션을 발행시켜 장부에 기록한다. 마찬가지로 각 차량에 부여된 고유 ID를 통해 트랜잭션을 조회할 수 있다.

V. 활용 방안

제안하는 시스템은 [Figure 2]와 같다. 본 시스템은 블록체인 기반으로 App와 연동시켜 중고차 시장이나 자동차 보험시장에 이용하도록 한다. 클라이언트에는 차량 등록, 이전을 담당하는 공공기관, 사고 이력을 기록하는 보험사, 제조사, 정비업체 등이 있고 주행거리 정보를 전송할 차량, 마지막으로 기록된 차량 정보를 확인하는 사용자가 있다. 또한, 시스템을 구성하는 노드들은 제조사, 보험사, 공공기관, 정비업체 등 다양한 기관들이 포함된다. [Figure 2]의 App에서 App Management 기능은 Hyperledger Composer와 통신하여 주행기록, 사고기록 등에 대한 정보를 트랜잭션 처리하고 App 내 Membership을 관리한다. Write Vehicle mileage 기능은 각 차량이 주행 중 임의로 스냅숏을 진행하여 주행거리를 App로 전송한다. Write Accident Record 기능은 차량 사고 발생 시 차량의 충격 감지 센서에 의해 파손된 부의를 파악하여 App로 전송하고 차량 사고에 따른 각 보험사, 정비소의 견적 및 사고 내용을 작성한다. Check Vehicle Information 기능은 블록체인 장부에 기록된 차량데이터를 조회하는 기능을 한다.

VI.기대되는 효과

차량 소유주와 정비업체가 동모해서 사고 이력을 위조할 수 있다. 이 문제점을 해결하고자 본 시스템에서는 차량에 사고 발생 시 충격 감지 센서를 사용하여 파손된 부위에 대한 정보를 네트워크로 전송하고, 전송된 정보와 보험사 및 정비업체가 작성한 정보를 대조하여 사고 이력 위조를 방지할 수 있을 것으로 기대된다. 충격을 감지하면 사고 이력을 발생시켜 네트워크에 연결되기 때문에 허위사고 수법도 방지할 수 있다. 이로 인해 투명하고 신뢰성 있는 보험처리가 가능해지기 때문에 자동차 보험 사기에 대한 일부분의 가능성을 배제할 수 있다.

두 번째로, 중고차 시장에서 데이터 위변조 시스템을 활용하면 사고 이력, 주행거리 등 모든 차량 데이터 위변조가 방지되기 때문에 사기 행각뿐 만 아니라 현 시장에서의 정보 비대칭성으로 인한 문제점 및 중고차 시장의 신뢰성 회복에 기여할 수 있다. 판매자는 판매가격을 예측하고 이해할만한 가격에 차량을 처분할 수 있고 악의적인 딜러는 사기행각을 하지 못하게 할 수 있으며 정직한 딜러는 합리적인 기준을 가지고 차량을 매입해 판매할 수 있다. 또한, 구매자는 투명하고 신뢰성 있는 차량 데이터를 조회하며 원하는 중고차를 적절한 가격에 구매할 수 있고 보험사는 차량 이력을 비교해

보험료를 차별화할 수도 있다. 또한, 제조사도 주행거리에 따른 고장, 잦은 노화 부위를 파악해 차량을 분석하여 차 소유주에게 차량 정보를 제공할 수 있다.

VII.향후 과제

사용자들의 입장에서 접근을 할 때에 사용자들이 차량데이터를 검색하는 단계에서 직면하는 문제는 무엇이 있을지 또, 이러한 문제를 해결하기 위해 시스템을 설계하기 위한 요구사항이 무엇인지에 대한 고찰이 필요하다.[4] 향후, 이러한 고려 점을 토대로 구현 방향에서의 기준을 잡아야 한다.

VIII.결론

본 논문에서는 최근 꾸준한 성장세를 보여주는 중고차 시장, 이에 파생적인 모든 시장에서 차량 데이터의 중요성을 언급하였다. 그리고 이에 따른 각 시장에서의 차량 데이터의 위변조 문제점을 해결하고자 블록체인 기술을 이용하여 차량데이터 위변조 시스템을 제안하였다. 본 시스템은 앞으로 중고차 시장 등에서의 신뢰성 회복에 기여할 것으로 기대된다.

[참고문헌]

[1] 이면상. (2017). "중고차 거래, 성능·상태 점검 관련 소비자피해 여전", 한국소비자원 홈페이지,http://www.kca.go.kr/brd/m_32/view.do?seq=2298&multi_itm_seq=0

[2] 국가법령정보센터. (2018). "자동차 관리법 제 58조", 국가법령정보센터 홈페이지, http://www.law.go.kr/lsInfoP.do?lsiSeq=198295&efYd=20181025#0000

[3] IBM. (2018). "HYPERLEDGER",https://www.hyperledger.org/

[4] Zavolokina, Liudmila, Gianluca Miscione, and Gerhard Schwabe. "Buyers of Lemons: Addressing Buyers' Needs in the Market for Lemons with Blockchain Technology."

블록체인 소프트웨어의 취약점을 이용한 인젝션 공격에 대한 연구*

김병국* 허준범**

*고려대학교 컴퓨터정보통신대학원 **컴퓨터학과

Analysis of Blockchain Software Vulnerability against OS Command Injection Attack

Byoungkuk Kim* Junbeom Hur**

*Graduate school of Computer and Information Technology,

**Department of Computer Science and Engineering

Korea University

요 약

블록체인 기술은 비트코인과 함께 알려졌으며, 급진적인 암호화폐의 성장과 함께 주요 기술로 인식되었다. 블록체인은 여러 분야에서 활용하기 위한 노력과 함께 발전하고 있다. 하지만 블록체인의 발전과 함께 블록체인을 위협하는 보안사고도 증가하고 있다. 본 논문에서는 블록체인 소프트웨어 중 오픈소스로 개발되는 암호화폐를 대상으로 소프트웨어의 구현상 문제점을 보안 관점에서 바라보며 소스코드에 존재하는 취약점을 분석하고, 취약점을 이용한 인젝션 공격에 대해 알아보겠다.

I. 서 론

블록체인 기술은 2008년 사토시 나카모토라는 필명으로 발표된 "비트코인 : 개인 간 전자화폐 시스템"[1]이라는 논문과 함께 암호화폐를 이루는 기술로 세상에 알려졌다.

블록체인은 암호화폐 시스템을 구현하기 위해 설계되었으나 작업 증명, 합의 메커니즘 등 블록체인 기술이 가지고 있는 여러 장점을 활용하여 기존의 중앙 집중형 시스템을 개선하기 위한 연구가 다양한 분야에서 진행되고 있다.

대부분의 블록체인 소프트웨어는 오픈소스 프로젝트로 개발이 이루어지고 있어 누구나 참여할 수 있다. 오픈소스는 저작권자가 공개한 소스코드를 자유롭게 활용할 수 있으며, 전 세계의 수많은 개발자가 소스코드의 문제점을 찾아 수정하는 활동을 통해 소프트웨어의 신뢰성을 향상시키는 장점이 있으며 블록체인 소프트웨어도 이러한 활동을 통해 많은 개선이 이루어져 높은 소프트웨어 신뢰성을 가지고 있다.

하지만 이러한 개선 활동은 소프트웨어의 신뢰성을 향상시키는데 집중되어 있으며 보안성에 대한 개선은 부족한 상황으로 OpenSSL, Apache Struts2, Ghost, Venom 등 높은 신뢰성을 가진 오픈소스 프

로젝트에 존재하는 취약점을 이용한 사고[2]가 보고되는 등 오픈소스 프로젝트의 보안성에 대한 문제가 발생하고 있다. 이러한 문제는 소프트웨어 신뢰성과 보안성을 별개로 인식하지 않아 발생하는 문제로 두 속성은 반드시 구별되어 개선 활동을 해야 한다.

다른 오픈소스 프로젝트와 마찬가지로 운영되는 블록체인 소프트웨어도 취약점이 존재할 경우 피해가 발생 할 수 있으며, 시가총액이 300조[3]에 달하는 암호화폐의 경우 문제가 더욱 심각하다.

암호화폐는 블록체인으로 구현된 대표적인 소프트웨어로 화폐와 직접적인 관련이 있어 해커들에게 집중 공격 대상이 되고 있으며 암호화폐와 관련된 소프트웨어의 취약점을 이용하거나 상대적으로 취약한 암호화폐 거래소의 시스템을 해킹하는 등 다양한 방법으로 암호화폐를 탈취하기 위한 시도를 하고 있다.

대부분의 블록체인 시스템은 프로그램 중지 없이 오랜 시간 동안 가동되고 있어 공격에 노출되기 쉽고 구현상의 문제를 발견하더라도 취약한 코드를 패치하기가 어렵다는 문제가 있다.

따라서 이러한 문제를 해결하기 위해서는 선제적 대응으로 개발단계에서부터 블록체인 소프트웨어의 취약점 제거가 필요하다.

본 논문에서는 오픈소스 프로젝트로 구현된 블록체인 소프트웨어 중 암호화폐를 대상으로 하며 소스코드 분석을 통해 존재하는 취약점을 분석하고, 발생할 수 있는 문제에 대해 알아보겠다.

II. 블록체인

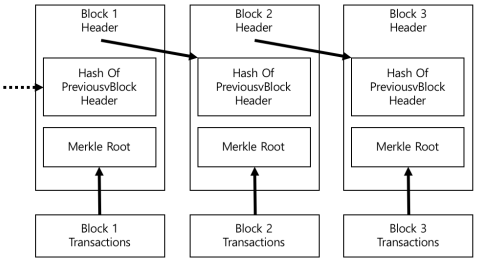
2.1 블록체인 구조

블록체인은 참여하는 모든 노드가 공유하는 데이터베이스이며 트랜잭션을 포함하고 여러 블록이 체인처럼 연결된 블록의 집합체로 모든 블록에는 이전 블록의 해시값이 포함되고 이 해시값은 그림 1과 같이 현재 블록과 이전 블록을 연결해주는 역할을 한다.

각 블록 헤더의 version은 소프트웨어의 버전을 나타내고, previous block header hash는 이전 블록 헤더의 해시값, merkle root hash는 블록에 포함된 모든 트랜잭션 해시값을 2진 트리 형태로 구성하고, 트리 루트에 해당하는 값을 해시 연산한 값이다.

time은 블록이 생성된 시간, nBits는 난이도 수치, nonce는 난이도를 만족시키는 해시값을 찾기 위해 변화하는 값으로 이 nonce를 변경해가며 해시값을 찾는다.

Fig. 1. Blockchain Structure



이렇게 nonce 값을 찾는 것을 작업증명이라 하며 암호화 해시의 특성을 활용한다. 해시 함수는 pre-image, second preimage, collision에 대한 안정성[4]을 가지기 때문에 같은 해시값을 가지는 원래의 메시지를 찾거나 해시값만으로 메시지를 계산하는 것이 불가능하다.

따라서 이를 무력화 시키기 위해서는 51% 이상의 컴퓨팅 파워를 가져야 하며 블록이 일정 시간마다 계속 생성되기 때문에 특정 블록의 트랜잭션을 수정하려면 그 이후에 생성된 모든 블록을 수정해야 한다. 결국 특정 블록의 트랜잭션을 수정하는 것은 불가능하며 작업증명은 완료된 트랜잭션을 되돌릴 수 없게 하는 블록체인의 핵심요소이다[5].

2.2 블록체인 응용 시스템

기존 시스템을 블록체인 시스템으로 대체하기 위해 다양한 분야에서 개발하여 적용하고 있다. 금융권에는 대표적으로 Ethereum을 활용한 청산결제와 Ripple 프로토콜을 활용한 해외 송금 서비스가 있다. 청산결제는 Ethereum 분산 원장을 사용하여 청산결제 구조를 개발하고 스마트 콘트랙트를 사용하여 프로세스를 자동화 하도록 하고 있으며 해외송금은 Ripple 프로토콜의 빠른 전송속도를 이용하여 은행과 국가 간의 자금 이체를 수행하도록 테스트가 진행 중이고 Ripple의 암호화폐인 XRP를 사용한다.

세계 각국의 정부에서도 블록체인 시스템 도입을 하려고 노력하고 있는데 스페인 포데모스 정당, 덴마크 자유 동맹당에서는 블록체인 투표 시스템을 시범적으로 사용하고 있다. 이외에도 기록물 유지관리, 토지 소유권 기록, 전자시민권 공증 서비스 등 다양한 시스템에 블록체인을 적용하기 위한 노력이 이어지고 있다[6].

III. 취약점 분석

3.1 CWE / SANS TOP 25 Most Dangerous Software Errors

CWE / SANS TOP 25 Most Dangerous Software Errors[7]는 CWSS 점수 공식에 따라 계산된 상위 25개 항목들을 정리한 것으로 프로그램 개발상에 발생하는 실수 등 잘못된 사항에 대해 점검을 하는 것에 중점을 두고 보안 약점 및 취약점을 유발하는 가장 중요한 프로그래밍 오류들을 명시하고 있으며 개발상의 실수를 줄이는데 목표를 두고 있다.

3.2 취약점 분석 방법

소프트웨어의 취약점을 분석하고 제거하는 것은 매우 어렵고 큰 비용이 들며 완전히 제거하기가 불가능하다. 그리고 하나의 보안 약점을 이용한 다수의 취약점이 존재하기 때문에 취약점의 원인이 되는 보안 약점을 제거하는 것이 효율적이다.

보안 약점은 정적 분석 도구를 활용하여 효율적으로 분석 및 제거를 할 수 있다. 정적 분석 도구는 소프트웨어의 실행 없이 소스코드와 목적 파일을 분석해 보안 약점을 검출해 내는 역할[8]을 하며 멀티 랭귀지를 지원하는 정석 분석 도구는 표 1과 같다.

Table 1. Static Analysis Tools

Manufacrurer	Product Name
Synopsys	Coverity
Micro Focus	Fortify
Checkmarx	CxSAST
Clint	Clint
Facebook	Infer

다양한 분석 도구 중 Fortify의 정적 코드 분석기는 소스코드의 보안 취약점을 검출해 내고 시큐어 코딩을 적용할 수 있도록 가이드를 제공한다. 또한 25개 이상의 언어와 다양한 플랫폼 및 프레임워크를 지원하며 CWE와 OWASP TOP 10등 다양한 취약점 항목을 수용하여 취약점을 분석해 내는 장점이 있어 이를 활용하기로 했다.

IV. 블록체인 취약점 분석

4.1 분석 방법

블록체인을 활용한 대표적인 분야인 암호화폐를 분석 대상으로 하였다. 그중 오픈소스 프로젝트로 개발되고 있는 암호화폐 중 37.4%와 17.5%의 점유율로 2018년 5월 기준 전 세계에서 가장 많이 사용되는 상위 두 개의 암호화폐인 비트코인와 이더리움을 대상으로 선정하였으며[9], Fortify 정적 코드 분석기를 활용하여 소스코드를 CWE / SANS TOP 25 Most Dangerous Software Errors의 항목을 기준으로 분석해 보았다.

4.2 분석 결과

정적 분석 도구로 소스코드를 점검한 결과 대상 프로젝트 모두에서 취약점이 탐지되었다.

Table 2. 3 high level categories

Project	Group	Total
Bitcoin	Insecure Interaction	1
	Porous Defenses	6
	Risky Resource Management	46
Ethereum	Insecure Interaction	2
	Porous Defenses	0
	Risky Resource Management	2130

탐지 결과를 크게 3가지로 분류하면 표 2와 같다. 안전하지 않은 상호작용은 데이터가 별도의 모듈, 프로그램, 프로세스 또는 시스템 간에 안정하지 않은 방법으로 전송되는 문제이며, 위험한 자원관리는 시스템 리소스의 생성, 사용, 전송 또는 파기를 소프트웨어가 제대로 관리하지 못하는 문제이다. 마지막으로 위험한 자원관리는 사용한 방어기술이 오용, 남용되거나 쉽게 무력화되는 문제를 분류한 것이다.

탐지된 취약점을 CWE / SANS TOP 25 Most Dangerous Software Errors 목록을 기준으로 분석해 본 결과 25개 항목 중 2순위에 해당하는 OS Command Injection에 취약한 코드가 존재하는 것을 확인할 수 있었다.

우선 Bitcoin 프로젝트에서 탐지된 코드를 살펴보면 그림 2와 같다.

Fig. 2. Bitcoin vulnerability code

```
void runCommand(const std::string& strCommand)
{
    if (strCommand.empty()) return;
    int nErr = ::system(strCommand.c_str());
    if (nErr)
        LogPrintf("runCommand error: system(%s)
returned %d₩n", strCommand, nErr);
}
```

runCommand 함수는 입력 값을 빈 문자열 체크만 하고 system 함수를 호출하고 있다. system 함수는 스트링을 command line 명령 프로세서로 전달하는 역할을 하여 프로그램을 실행해 준다. runCommand 함수의 입력 값을 체크 하지 않아 해커가 입력 값을 조절 할 수 있다면 의도되지 않은 악성코드를 실행 할 수 있는 문제가 발생하며 운영체제에 직접 접근하지 못하는 해커도 이 취약한 코드를 통해 접근 할 수 있게 되어 취약한 코드가 높은 권한을 가지는 프로그램에 있다면 공격자가 획득하지 못한 권한으로 프로그램을 실행 할 수 있는 위험에 노출된다.

따라서 허용할 입력 값만을 지정하거나 안전하지 않은 입력 값을 제거하도록 해야 한다.

Fig. 3. Ethereum vulnerability code

```
void getFromExecution(std::string command) {
    Timer timer;
    timer.start();
    // do this using just core stdio.h and stdlib.h, for
portability
    // sadly this requires two invokes
    code = system(("timeout " +
std::to_string(timeout) + "s " + command + " >
/dev/null 2> /dev/null").c_str());
    const int MAX_BUFFER = 1024;
    char buffer[MAX_BUFFER];
    FILE *stream = popen(("timeout " +
std::to_string(timeout) + "s " + command + " 2>
/dev/null").c_str(), "r");
    while (fgets(buffer, MAX_BUFFER, stream) != NULL) {
      output.append(buffer);
    }
    pclose(stream);
    timer.stop();
    time = timer.getTotal() / 2;
  }
```

그림 3은 Ethereum에서 탐지된 취약한 코드이다.

getFromExecution 함수에 존재하는 코드로 system 함수와 popen 함수를 사용하는 부분이다. system 함수와 popen 함수 모두 커멘드를 실행하기 위해 사용한다는 공통점이 있지만, system 함수와는 다르게 popen함수는 실행 후 결과값을 얻을 수 있다는 차이가 있다.

탐지된 코드를 살펴보면 timeout 함수로 다음 명령어의 실행 시간을 제한하고 있다.

실행 시간을 제한하고 있어 입력 값이 의도하지 않은 프로그램을 실행해도 제한 시간 내에 종료되기 때문에 문제가 없어 보이지만 해커나 제한 시간을 조절할 수 있거나 제한 시간 내에 프로그램을 실행해 목적을 이룰 수 있으므로 취약한 코드이며 system 함수와 popen함수를 실행하기 전에 반드시 입력 값을 체크해야 한다.

V. 결 론

본 논문에서는 블록체인 소프트웨어의 취약점을 소스코드 정적 분석을 통해 소개하였다.

암호화폐, 제조, 판매, IoT 등 다양한 블록체인 소프트웨어 중 가장 많이 활용되고 있는 암호화폐를 대상으로 분석한 결과 취약한 코드가 탐지되었으며 취약한 코드 중 높은 위험성을 가지는 OS Command Injection이 가능한 코드가 존재하는 것을 확인할 수 있었다.

블록체인이 가지는 보안성이 뛰어나더라도 이를 활용한 소프트웨어를 개발할 때 발생하는 구현상의 문제를 해결하지 못한다면 블록체인은 지속해서 위협에 노출되며 심각한 보안사고가 발생할 수 있다.

이러한 문제는 다양한 분야에서 블록체인을 활용하려는 노력을 저해할 수 있는 문제로 설계, 구현 단계에서부터 소프트웨어의 보안성을 고려하여 해결해야 한다.

향후 연구에서는 다양한 활용 분야의 블록체인 소프트웨어를 분석해 보고 탐지된 취약점을 이용하여 공격할 수 있음을 재연해보고 공격을 통해 발생할 수 있는 문제에 대해 연구하겠다.

References

[1] Satoshi Nakamoto, "Bitcoin : A Peer-to-P-eer electronic cash system", 2008.

[2] Redhat, Red Hat Product Security Risk Report, 2015. https://access.redhat.com/blogs/766093/posts/2262281

[3] CoinMarketCap. Top 100 Cryptocurrencies By Market Capitalization, https://coinmarketcap.com/ko/

[4] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, 1996.

[5] Li X, Jiang P, Chen T, Luo X and Wen Q, A survey on the security of blockchain systems, 2017.

[6] Financial Security Institute, Development

Status of Block Chain Application Technology and Introduction Example by Industry, 2017.

[7] CWE. 2011 CWE/SANS Top 25 Most Dangerous Software Errors, http://cwe.mitre.org/top25/index.html

[8] A. Gosain and G, Sharma, Static analysis: A survey of techniques and tools, 2015.

[9] Hanwha Investment and Securities, BlockChain Issue, 2018.

EOS 암호화폐의 블록 생성에 대한 인센티브 분석

한성호* 안형철* 김광조*
*KAIST 정보보호대학원

Analysis of Incentive for Block Producer in EOS

Seongho Han* Hyeongcheol An* Kwangjo Kim*
*Graduate School of Information Security, KAIST.

요 약

비트코인이 제안된 이후 다양한 암호화폐가 개발되었다. EOS는 비트코인의 불필요한 에너지 소모와 낮은 거래 처리량, 그리고 제한된 기능 제공을 극복하기 위해 고안되었다. 합의 방식으로 위임지분증명(DPoS) 방식을 채택한 EOS에서는 개별 블록 생성을 위한 경쟁이 없다. 그러나 EOS의 블록 생성자 선거에서 승리하기 위해서는 성능이 좋은 하드웨어를 갖춰야 하므로 참여 비용이 높다. Itay 등은 블록 생성 보상과 비용의 균형이 깨지면 암호화폐의 보안이 취약해질 우려가 있음을 증명하였다. 따라서 보상과 비용의 차이를 계산함으로써 EOS 노드가 블록 생성자가 되려는 인센티브를 분석해야 할 필요가 있다. 본 논문에서는 EOS의 블록 생성 보상과 비용을 정량적으로 계산함으로써 블록 생성자의 인센티브를 분석하였다.

I. 서론

2008년 Satoshi Nakamoto에 의해 비트코인 [1]이 제안된 이후 다양한 암호화폐가 개발되었다. 비트코인을 포함한 퍼블릭 블록체인은 탈중앙화된 경제시스템 구축을 목표로 한다. 퍼블릭 블록체인에서는 서로 다른 노드들이 동일한 데이터베이스를 유지하기 위해 합의 알고리즘을 사용한다. 비트코인은 작업증명 합의 방식(PoW)을 합의 알고리즘으로 채택하고 있다.

합의 알고리즘이 제대로 기능하기 위해서는 노드들의 자발적 참여가 필수적이다. 그러나 작업증명 합의 방식의 경우 불필요한 에너지 소모가 많고, 경쟁이 심해지면 채굴 비용이 증가하여 신규 사용자의 진입 장벽이 높아진다는 단점이 있다. Itay 등 [2]은 장래에 비트코인의 채굴 보상이 줄어들면 채굴자들이 블록 생성에 참여하지 않는 시간이 증가할 수 있음을 증명하였다. 노드들이 채굴에 참여하지 않는 시간이 길어지면 블록체인의 보안이 취약해진다.

EOS [3]는 비트코인의 불필요한 에너지 소모와 낮은 거래 처리량, 그리고 제한된 기능 제공을 극복하기 위해 고안된 암호화폐이다. EOS는 비트코인과 달리 합의 방식으로 위임지분증명(DPoS) 방식을 채택하고 있다. 위임지분증명 방식에서는 선거에서 이긴 노드만이 블록 생성에 참여하기 때문에 개별 블록 생성을 위한 경쟁이 없다. 따라서 시간에 따라 블록 생성에 요구되는 연산량이 증가하지 않는다.

그러나 EOS에서는 블록 생성을 위해 블록 생성자 선거에서 승리해야 한다. 선거에서 승리하기 위해서는 성능이 좋은 하드웨어를 갖춰야 하므로 참여 비용이 높다. Itay 등은 블록 생성 보상과 비용의 균형이 깨지면 블록 생성에 참여하려는 노드의 숫자가 적어져 암호화폐의 보안이 취약해질 우려가 있다고 주장하였다. 따라서 보상과 비용의 차이를 구함으로써 EOS 노드가 블록 생성자가 되려는 인센티브를 분석해야 할 필요가 있다.

본 논문에서는 EOS의 블록 생성 보상과 비

용을 정량적으로 계산한다. 그리고 블록 생성 보상과 비용의 차이를 통해 블록 생성자의 인센티브를 분석한다.

1.1 논문의 구성

본 논문의 구성으로 Ⅱ장에서는 EOS 및 위임지분증명(DPoS) 방식에 대하여 설명하고, Ⅲ, Ⅳ장에서는 각각 EOS의 블록 생성 보상과 비용을 분석한다. Ⅴ장에서는 블록 생성 보상과 비용을 고려하여 블록 생성자의 인센티브를 분석한다. 마지막으로 Ⅵ장에서는 결론과 본 논문 성과를 이용한 추후 연구에 대한 가능성을 제시한다.

II. 배경지식

배경지식에서는 EOS와 EOS에서 사용되는 위임지분증명(Delegated Proof-of-Stake, DPoS) [3, 6]에 대하여 설명한다.

2.1 EOS

EOS는 비트코인의 제한된 기능, 불필요한 에너지 소모, 낮은 거래 처리량을 보완하기 위해 개발된 암호화폐다. EOS를 구현한 소프트웨어는 EOS.IO로 탈중앙화 어플리케이션의 수직 및 수평적 확장이 가능하다. EOS.IO는 계정, 인증, 데이터베이스, CPU 스케줄링 등 수많은 기능을 제공한다. 거래 처리량을 초당 수천개로 유지하고, 거래 수수료가 없다는 것이 장점이다.

2.2 위임지분증명

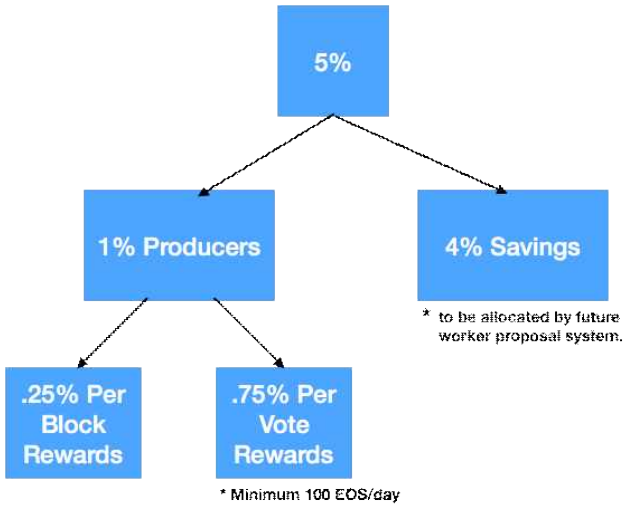
위임지분증명 방식은 작업증명 방식과 달리 블록 생성자가 되기 위해선 투표를 거쳐야 한다. EOS에서는 투표를 통해 21명의 블록 생성자(Block Producer, BP)를 선출한다. 선거 방식은 1인 1표가 아니라 1코인 1표 정책을 선택한다. 누구나 블록 생성자 선거에 참여할 수 있으며, 선거에서 득표 순위가 21위 안에 들 때 블록 생성자가 될 수 있다. 선출된 대표자들은 암호화폐가 정한 계획에 따라 블록을 생성한다. 투표 결과 21위 안에 들지 못했으나 40위 안에 든 사람들에게는 블록 생성후보자의 지위를 부여한다. 장래에는 121위 안에 드는 사람까지 블록 생성후보자의 지위를 부여할 예정이다.

EOS는 블록 생성자들이 서로 협력하는 환경하에서 블록을 생산하기 때문에 이론적으로는 어떤 분기(fork)도 일어나지 않는다. 만약 분기가 발생했을 시 합의는 자동으로 블록이 가장 긴 체인으로 전환된다.

III. EOS 블록 생성자의 블록 생성 보상 체계

EOS 시스템은 2018년 6월 메인넷이 활성화된 시점에 총 10억 개의 토큰을 발행했다. 이후 연간 5%의 인플레이션율에 따라 토큰 발행량이 증가할 예정이다. [그림 1]의 5%는 연간 인플레이션율을 나타낸 것이다.

[그림 1]을 보면 알 수 있듯이 EOS 시스템은 총 EOS 토큰 발행량의 1%를 블록 생성자와 후보자에게 수여한다. 블록 생성자에게 주어지는 보상은 크게 두 종류로 나눠진다. 블록 생성을 수행함으로써 얻는 보상과 투표 득표율에 따라 얻는 보상이 있다. 단, 두 개의 보상을 합쳐 100 EOS 토큰을 초과하지 않으면 어떠한 보상도 주어지지 않는다. 한편 블록 생성후보자는 오직 득표율에 따른 보상만 받을 수 있다. 후보자의 경우 득표율에 따른 보상이 하루에 100 EOS 토큰을 초과하지 않으면 어떠한 보상도 받을 수 없다.



[그림 1] 블록 생성 보상 개념도 [4]

[표 1] EOSUK의 하드웨어 장비 [5]

역할	CPU	RAM	Disc	Expansion
Block Producer Primary	Dual Xeon 3.2GHz 8 Core	128 GB DDR4	512 GB SSD	Yes
Full Node Primary	Dual Xeon 3.2GHz 8 Core	64 GB DDR4	512 GB SSD	Yes

3.1 블록 생성 보상

블록 생성 보상은 블록 생성을 하는 노드에만 주어진다. [그림 1]은 총 EOS 토큰 발행량의 0.25%가 21명의 블록 생성자에게 주어진다는 것을 보여준다. 모든 블록 생성자는 동일한 블록 생성 보상을 받는다. 즉 각각의 블록 생성자에게는 총 EOS 토큰 발행량의 0.0119%가 주어지게 된다. 2018년 6월 기준 EOS의 총 토큰 발행량은 10억이다. 따라서 각각의 블록 생성자들은 1년 동안 119,047 EOS 토큰을 보상으로 받으리라 예측된다. 일별로 보상을 계산해보면 각 블록 생성자들은 하루에 326.16 EOS 토큰을 받을 수 있다. 블록 생성자는 일일 보상이 100 EOS를 초과하므로 항상 보상을 받을 수 있다.

3.2 득표율에 따른 보상

투표 보상에 할당된 총 EOS 토큰은 [그림 1]을 통해 알 수 있듯이 총 EOS 토큰 발행량의 0.75%이다. 각 블록 생성자와 블록 생성후보자는 선거 시 득표한 비율에 따라 보상을 받게 된다. 예를 들어 투표에서 2%의 득표율을 기록한 경우 2018년 기준 1,000,000,000×0.075×0.02인 150,000 EOS 토큰을 받을 수 있다. 이를 하루 평균으로 계산하면 410.96 EOS이다. 즉 득표율이 2%인 경우에는 하루 수익이 100 EOS 토큰을 초과하므로 투표 보상을 받을 수 있다. 블록 생성자로 채택되지 못하고 블록 생성 후보자가 된 경우 투표 보상을 받기 위해서는 득표율이 0.49%를 넘어야 한다. 0.49%라는 임계수치는 총 EOS 토큰 발행량이 변하면 달라질 수 있다.

IV. EOS 블록 생성자의 블록 생성 비용 체계

EOS의 블록 생성에 요구되는 비용으로는 자본 비용과 실행 비용이 있다 [2]. 자본 비용은 CPU, GPU나 RAM같은 하드웨어 장비를 갖추기 위해 소모되는 비용이다. 반면 실행 비용은 실제 블록 생성 시 소모되는 전력에 따른 비용이다.

4.1 자본 비용

EOS 블록 생성자들은 비트코인 채굴자들과 달리 개별 블록 생성을 위해 경쟁하지 않는다. 그러나 EOS 사용자들은 CPU, 네트워크 기능을 블록 생성자로부터 임차하여 사용하기 때문에 블록 생성자에게 높은 하드웨어 수준을 요구한다. 선거 유세 시 유권자들에게 높은 하드웨어 성능을 약속하지 못하는 후보자들은 많은 득표를 할 수 없다. 실제로 [9]는 사용자들이 블록 생성자들에게 고사양의 하드웨어를 갖출 것을 요구한다는 것을 보여주었다. [표 1]은 2018년 11월 기준 득표율이 70~80위에 속한 블록 생성 후보자 EOSUK의 하드웨어 장비를 나타낸 것이다. 이를 통해 개별 블록 생성을 위한 경쟁이 없더라도 블록 생성자가 되기 위한 자본 비용이 높다는 것을 유추할 수 있다.

4.2 실행 비용

블록 생성자가 블록을 생성하고 EOS 사용자에게 편의를 제공하기 위해서는 상시적으로 생성자가 소유한 서버의 전원을 켜놓아야 한다. 그러므로 하루 24시간 내내 전력을 소비할 것으로 예측된다.

V. EOS 블록 생성 인센티브 분석

블록 생성 보상과 블록 생성 비용을 고려하면 블록 생성에 따른 인센티브를 구할 수 있다.

인센티브 = 총 블록생성보상 − 총 블록생성비용

인센티브가 0보다 커야만 자원해서 블록 생

성자가 되려고 할 것이다. 만약 어떤 블록 생성자의 인센티브도 0보다 작으면 블록 생성자가 되려는 사람이 없게 되고, 이에 따라 EOS 시스템 자체가 마비된다.

5.1 가상의 상황을 통한 블록 생성 인센티브 측정

가상의 상황을 통하여 블록 생성 인센티브를 측정해본다. 먼저 투표에서 2%를 획득하여 블록 생성자가 되었다고 가정한다. 그리고 하드웨어 장비는 EOSUK와 동일하게 구비했다고 가정한다. 전력 소비는 영국에서 24시간 동안 했다고 가정한다. 인센티브는 하루를 기준으로 측정한다.

총 블록 생성 보상은 블록 생성자로써 얻는 블록 생성 보상과 투표로 얻는 보상의 합으로 구할 수 있다. 블록 생성자로써 얻는 블록 생성 보상은 326.16 EOS다. 3장 2절에서 보듯이 투표로 얻는 보상은 410.96 EOS다. 즉 총 블록 생성 보상은 737.12 EOS다. 2018년 11월 기준 EOS를 달러로 환산할 경우 5.73의 교환비를 가지므로 [7], 일일 보상은 4223.6976$이다.

한편 총 블록 생성 비용은 자본 비용과 실행 비용의 합이다. 자본 비용은 Intel CPU Xeon E5-2667v4 8Core 3.20GHz 2200$, RAM 128GB DDR4-2666 2900$, Samsung 850 PRO-512GB 150$다. 이때 하드웨어는 1년간 사용한다고 가정하면 하루에 지출하는 비용은 14.38$다. 실행 비용은 EOS가 1시간에 평균적으로 1800KW를 사용하므로 [8], 영국 기준 0.22$/kWh [10]의 전기 요금을 대입하면 9.504$이다. 즉 총 블록 생성 비용은 하루에 23.884$이다.

블록 생성 보상과 비용을 이용하여 일일 인센티브를 계산해보면 4199.8136$이다. 그러므로 블록 생성자는 계속 현재 상태를 유지하려 할 것이다. 그러나 서버에 필요한 컴퓨터를 100대로 유지하면서, 블록 생성자의 지위를 잃고, EOS 시장 가치가 3$로 하락하게 되면 손해를 볼 가능성이 생긴다. 따라서 EOS의 시장 가격이 적절하게 유지되는 것은 EOS 시스템이 원활하게 돌아가는데 필수적이다. 또한, 특정 블록 생성자가 득표율을 지나치게 많이 가져가지 않는 것도 EOS 시스템이 제대로 잘 작동하기 위해 필수적이다.

VI. 결론 및 향후 과제

본 논문에서는 EOS의 블록 생성 보상과 비용을 정량적으로 계산함으로써 블록 생성자의 인센티브를 분석하였다. 현재는 21개의 블록 생성자 노드들이 상태를 유지하려는 경제적 유인이 충분하다고 볼 수 있다. 그러나 외부 조건이 변할 경우 블록 생성자 노드들이 현 상태를 유지하지 않을 가능성도 있다.

향후 연구에서는 블록 생성 후보자들 사이의 경쟁에 따른 블록 생성 인센티브를 분석한다. 그리고 경쟁 조건에 따라서 EOS 시스템이 어떠한 방식으로 작동할지 예측해본다.

[참고문헌]

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system" 2008.

[2] Tsabary, Itay, and Ittay Eyal. "The Gap Game." Proceedings of the 11th ACM International Systems and Storage Conference. ACM, 2018.

[3] https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[4] https://medium.com/eosio/introducing-eosio-dawn-4-0-f738c552879

[5] https://eosuk.io/eos-block-producer-technical-specifications/

[6] Larimer, Daniel. "Delegated proof-of-stake (dpos)" Bitshare whitepaper, 2014.

[7] https://coinmarketcap.com/

[8] https://medium.com/genereos/eos-energy-consumption-vs-bitcoin-and-ethereum-2d1bb31ed72f

[9] https://steemit.com/eos/@blockmaker/eos-io-block-producer-guesstimated-specsl

[10] https://en.wikipedia.org/wiki/Electricity_pricing

블록체인 기반 사용자 계정 관리 기술 동향 조사

이대화[1], 김베드로[1], 지우중[1], 김형식[1], 우사이먼[2]
성균관대학교 전자전기컴퓨터공학과[1], 한국뉴욕주립대학교 컴퓨터과학과[2]

Survey on Blockchain Based Identity management system

Daehwa Rayer Lee[1], Bedeuro Kim[1], Woojoong Ji[1],
Hyoungshick Kim[1], Simon S Woo[2]

Department of Computer Science and Engineering,
Sungkyunkwan University, Korea[1]
Department of Computer Science,
The State University of New York, Korea[2]

요 약

블록체인은 투명성과 무결성을 보장하는 신기술로서 서로 신뢰할 수 없는 이종 도메인 간 아이디 관리 시스템에 결합하여 중립성, 공정성, 프라이버시, 인증 속도 및 효율의 증대와 서버 단일 실패점 완화, 서버 유지비용 감소 등의 장점을 개선할 수 있다. 본 논문에서는 이종도메인 간 아이디 관리 시스템인 OAuth(Open Authorization), OpenID(Open Identity), SAML(Security Assertion Markup Language)와 블록체인 기반 아이디 관리 시스템 프로젝트인 Banksign, Sorvin, Bitnation을 조사하였다.

I. 서론

2008년 나카모토 사토시가 금융기관인 제삼자를 거치지 않고 당사자 간의 개인 거래를 통해 이루어지는 암호 화폐 시스템인 비트코인은 오늘날 디지털 금이라고 불리우며 많은 사람들에게 관심을 받고 있다. 또한 블록체인은 비트코인의 기반 기술로 탄생하여, 이제는 투명성과 무결성을 보장하는 신기술로 주목받고 있다[1]. 또한 서로 신뢰할 수 없는 이종 도메인 간 아이디 관리 시스템은 Trusted Third Party를 통해 인증하게 되는데, 이때 자신의 신원을 인증하는 절차는 매우 높은 투명성과 무결성을 필요로 하게 된다. 그렇기 때문에 블록체인을 통해 구현하는 아이디 관리 시스템은 이미 많은 연구가 진행되고 있다. 본 논문에서는 기존 아이디 관리 시스템과 블록체인을 통해 구현하는 아이디 관리 시스템의 기술 동향을 조사할 것이다. 이를 통해 현재 연구되고 있는 기술들의 컨셉을 이해하고 추후 Single Sign On과 같은 기술도 구현가능하다는 가능성을 보여준다.

본 논문의 구성은 다음과 같다. 2장에서 블록체인에 대해 간단히 설명하고, 3장에서 기존에 사용되는 이종 도메인 간 아이디 관리 시스템을 소개한다. 4장에서 블록체인을 통해 강화할 수 있는 특징들을 정리한 뒤, 5장에서 블록체인 기반 아이디 관리 시스템을 소개하고, 6장에서 논문의 결론을 내린다.

II. 블록체인

블록체인은 모든 정보를 블록이라는 단위로 나누어 담아서 체인처럼 이어 붙인 뒤, P2P 네트워크를 기반으로 연결된 모든 노드에 공유하는 분산 원장 시스템이다. 블록체인의 특징은 누구나 변경의 결과를 열람할 수 있는 투명성과 아무도 조작할 수 없는 무결성 두 가지이다. 블록체인은 P2P 네트워크를 기반으로 하기 때문에 발생할 수 있는 문제는 크게 비잔틴 장군 문제[2]와 이중 지불 문제가 있다. 비잔틴 장군 문제는 분산 컴퓨터에서 발생할 수 있는 신뢰와 합의에 대한 문제로서, 시스템이 멈추거나 응답이 없는 경우 혹은 에러가 발생하거나 공격을 받아 잘못된 값을 전달하는 경우에 안전하고 신뢰할 수 있는 분산 컴퓨팅 환경을 만들기 위해 정의된 것이다. 반드시 2/3 이상의 노드가 정상적으로 동작해야 한다는 것이 증명되었다. 다음 이중 지불 문제는 하나의 금액을 여러 곳에서 사용할 수 있기 때문에 발생하는 문제이다. 이 문제는 모든 노드가 하나의 같은 원장을 갖지 못하고 각자의 원장만을 주장하는 것이 원인이다. 그래서 이러한 두 문제를 해결하기 위해서는 모든 노드가 인정할 방법을 통해 신뢰할 수 있는 트랜잭션 로그를 생성하고 모두에게 전파하여 공유하는 것이 중요한데 이때 사용하는 방법이 합의 알고리즘이다. 일반적으로 많이 사용하는 합의 알고리즘인 PoW 합의 알고리즘은 블록을 생성하기 위해 일정한 작업량을 수행했음을 증명하는 방식으로 이루어지며 주로 높은 Time complexity 혹은 Space complexity가 요구되는 문제를 통해 작업을 만들어낸다.

III. 이종 도메인 간 아이디 관리 시스템

이종 도메인 간 아이디 관리 시스템을 이해하기 위해 본 논문에서는 기존에 사용되고 있는 방식인 OAuth(Open Authorization)[3], OpenID(Open Identity)[4], SAML(Security Assertion Markup Language)[5]를 조사하였다.

3.1 OAuth(Open Authorization)

기존의 인증 방식에서는 매번 계정을 만들어 사용하기 때문에 아이디와 비밀번호의 관리가 어려울 뿐만 아니라 개인정보 유출에 따른 피해가 증가되는 문제가 야기되었다. 이를 해결하기 위해 제안된 OAuth는 웹과 앱 서비스 사이에서 제한적인 권한 요청을 통해 발급받은 키를 이용하는 표준화 인증 방식으로서, 보안 수준이 검증된 도메인의 API를 이용하여 보안성과 편의성을 개선하였다. 이는 권한 획득 방식에 따른 클라이언트의 유형, 자원에 대한 장기적 접근 여부, 제공 서비스의 특성에 따라 4가지로 분류된다.

3.1.1 권한 코드 승인

클라이언트가 자원 소유자에게 권한을 부여하지 않고, 자원 소유자가 직접 권한 서버에서 인증 및 권한 허가를 받게 되는 방식이다. 이는 토큰 전달과정에서 생길 수 있는 정보 유출 문제를 방지할 수 있다는 장점을 가지고 있다.

3.1.2 암묵적 승인

권한 코드 승인 과정에서 권한 코드 없이 액세스 토큰이 발급되는 방식으로 인증 과정이 간소화되어 사용자의 편의성은 높지만 보안성이 낮아지는 문제를 가지고 있다.

3.1.3 자원 소유자 비밀번호 자격증명 승인

클라이언트에 아이디와 비밀번호를 저장하여 직접 액세스 토큰을 받아오는 방식이다. 이 과정에서 자원 소유자는 비밀번호가 애플리케이션에 노출되는 피싱 위험에 노출되는 문제를 가지고 있다.

3.1.4 클라이언트 자격 증명

클라이언트가 자기를 인증할 수 있는 정보를 권한 서버에 보내어 액세스 토큰을 요청하게 된다. 클라이언트 인증만으로 액세스 토큰이 발급되기 때문에 안전한 인증 방식 사용 및 인증 정보의 규칙적인 변경이 필요한 방식이다.

3.2 Open Identity(OpenID)

하나의 아이디로 서로 다른 도메인에서 사용자를 인증하는 방식으로 아이디 소유자의 정보는 URL 형태(http://이용자ID.openid.net)로 표

현된다. OpenID는 한번 입력된 개인정보를 다른 도메인에서 요청하면 재입력 없이 사용자의 허가만으로 이용이 가능하기 때문에 전송 정보 및 여부에 대한 유출의 위험이 적다. 이는 OAuth(Open Authorization)와 같이 HTTP를 사용한다는 공통점 가지고 있지만, Open Identity는 인증의 목적, 즉 Open Identity Provider에서 사용자의 인증을 처리하는 목적이 있고 Open Authorization은 허가의 목적, 즉 API를 호출할 수 있는 권한이 있는 사용자인지를 확인하는 목적이 있다는 차이점이 있다.

3.3 SAML(Security Assertion Markup Language)

이종 도메인 간 ID 인증 및 권한에 대한 데이터를 교환하기 위한 XML 기반 프레임 워크이다. 이는 사용자가 일정 기간 인증을 서비스 업체에 위임하고 단일 로그아웃 기능을 사용하여서 한 사이트에서 로그아웃하면 다른 모든 서비스 업체에서도 함께 자동으로 로그아웃 되도록 한다.

IV. 블록체인을 통해 개선할 수 있는 이종 도메인 간 아이디 관리 시스템의 특징

블록체인을 통해 개선할 수 있는 이종 도메인 간 아이디 관리 시스템의 특징은 다음과 같다.

4.1 중립성과 공정성

기존 시스템은 한 회사가 인증 서비스를 독점하기 때문에 이종 도메인 간 중립성이나 공정성의 문제가 발생할 수 있지만, 블록체인은 운영의 주체가 없기 때문에 보다 중립적이고 공정한 시스템을 구성할 수 있다.

4.2 서버 단일 실패점의 보완

기존 시스템은 단일 서버의 인증 오버헤드가 가중되거나 해커로부터 공격을 받게 되면 서버가 중단되어 사용이 불가하지만, 블록체인은 모든 인증 정보를 공유하기 때문에 피어 노드가 로컬에서 인증 정보를 처리할 수 있는 구조로 서버가 중단되어도 사용할 수 있다.

4.3 서버 유지비용의 감소

기존 시스템은 중앙 서버가 백업 및 서버 관리에 대한 모든 비용을 부담하지만, 블록체인은 피어 노드가 백업 및 서버 관리에 대한 모든 비용을 나누어 부담하는 구조로 서버 유지비용을 감소할 수 있다.

4.4 프라이버시 강화

기존 시스템은 Trusted Third Party에 대한 신뢰를 가정하지만, 블록체인은 신뢰할 Trusted Third Party의 역할을 없애고 가명 계정 생성이 가능하므로 보다 높은 프라이버시 강화가 가능하다.

4.5 인증 속도 및 효율의 증대

기존 시스템은 인증의 주체가 트랜잭션을 Trusted Third Party에게 보내서 인증 정보를 받은 뒤 그것을 사용하는 절차로 이루어져있는데 블록체인에서는 인증의 주체가 바로 블록체인에서 전자 서명을 통해 자신을 인증한 뒤 사용하는 절차로 속도가 빠르고 방식의 효율성이 증가했다.

V. 블록체인을 통해 아이디 관리 시스템을 구현한 프로젝트

기존 아이디 관리 시스템은 신뢰할 수 있는 Trusted Third Party에 자신의 공개키를 등록하고 공개키에 대한 인증서를 발급받아 사용하는 PKI 기반 인증이다. 그러나 블록체인 기반 아이디 관리 시스템은 블록체인에 자신의 공개키를 등록하는 방식으로 탈 중앙형 PKI를 구현하는 것이 핵심이다. 우리는 현재 블록체인을 통해 아이디 관리 시스템을 서로 다른 목적을 위해 구현한 서비스들인 Banksign[6], Sorvin[7], Bitnation[8]을 조사해보았다.

5.1 Banksign

기존 은행권에서 한 은행으로 발급받은 공인인증서를 다른 은행에서 사용하려면 다시 발급받거나 타인증서 등록 절차를 거쳐서 사용할 수 있었던 문제를 앞서 말한 방식과 같이 공개키를 은행들의 컨소시엄이 만든 블록체인에 저장하고, 비밀키를 일반 NPKI 폴더가 아닌 개인 휴대폰 보안영역에 저장하여 보안성과 사용자

편의성을 높인 프로젝트이다. 기존 공인인증서와 차이점은 한 번의 등록만으로도 컨소시엄에 가입된 모든 은행에서 통합하여 사용할 수 있고 1인 1단말기 1인증서 정책으로 인증서 무단 복제를 방지하였으며 스마트폰에만 저장되는 방식으로 개인키 도난을 방지하였다는 점이다.

5.2 Sorvin

운전 면허증, 회사 신분증, 여권 등 퍼블릭 환경에서 다양하게 자신의 신원을 증명할 수 있도록 하는 인증정보를 블록체인을 통해 구현한 것으로, 공인기관을 일부에서만 인정하는 문제를 해결하여 전 세계에서 같은 수준의 인증을 받을 수 있도록 하는 프로젝트이다. 블록체인에 저장된 공개키 주소인 Decentralized Identifier를 이용하여 자신의 신원 정보를 암호화폐 지갑에 넣어두고 직접 안전하게 통제할 수 있도록 하여 보안성을 높이고 개인키가 있어야만 동작하도록 하여 위조나 변조가 어렵도록 보안성을 높였다.

5.3 Bitnation

거버넌스 플랫폼으로 블록체인상 신원등록을 통해 세계 시민권의 개념을 확립하고자 하는 목표로 헌법에 서명함으로써 가입할 수 있고 블록체인 여권이나 블록체인 결혼 증명서와 같은 형태로도 사용될 수 있다. Bitstation Refuge Emergency Response라는 식별 시스템을 사용하여 기존 신분증이 없는 사람들도 블록체인에 가입하여 신분증을 받을 수 있도록 허용하였고, 이것을 사용하여 사회적 지원이나 금융 서비스를 받을 수 있도록 하였다.

VI. 결론

우리는 본 논문에서 블록체인의 원리와 기존 아이디 관리 시스템 조사를 통해 블록체인으로 개선할 수 있는 특징들에 대해 정의 내렸고, 실제 블록체인 기반 아이디 관리 시스템 조사를 통해 기술이 어떻게 적용되는지를 알아보았다. 추후 이를 바탕으로 Single Sign On과 같은 기술을 블록체인으로 구현할 예정이다.

Acknowledgments. 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-01369, O2O 서비스를 위한 무자각 증강인증 및 프라이버시가 보장되는 블록체인 ID 관리 기술 개발)

[참고문헌]

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf

[2] Leslie Lamport, Robert Shostak, Marshall Pease, The Byzantine Generals Problem, https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

[3] Mark Atwood, Richard M. Conlan, Blaine Cook, Leah Culver, Kellan Elliott-McCrea, Larry Halff, Eran Hammer-Lahav, Ben Laurie, Chris Messina, John Panzer, Sam Quigley, David Recordon, Eran Sandler, Jonathan Sergent, Todd Sieling, Brian Slesinsky, Andy Smith, OAuth Core 1.0, https://oauth.net/core/1.0/

[4] Fitzpatrick, Brad, Distributed Identity: Yadis, Live Journal. Archived from the original on 2006-05-04. Retrieved 2008-03-20.

[5] OASIS, Security Assertion Markup Language (SAML) Version 1.0 an OASIS Open Standard, http://xml.coverpages.org/ni2002-11-12-b.html

[6] Banksign, http://www.banksign.or.kr/index.php

[7] Sorvin Foundation, Sovrin: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust, https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf

[8] Bitnation, https://github.com/Bit-Nation/Pangea-Docs/raw/master/BITNATION%20Pangea%20Whitepaper%202018.pdf

블록체인 기반의 소셜 미디어 플랫폼 분석을 통한 교내 탈중앙화 소셜 미디어 보상체계 제안

김범진*, 김은수*, 조수빈*, 이지수*, 김진우*, 반주형*,
이경모**, 노시완**, 이경현*

*부경대학교 IT융합응용공학과

**부경대학교 정보보호학협동과정

A proposal of decentralization social media reword system in campus through analysis of social media platform based on blockchain

Bumjin Kim*, Eunsu Kim*, Su-bin Jo*, Ji-soo Lee*, Jin woo Kim*, Juhyoung Ban*,
KyungMo Lee**, Siwan Noh**and Kyung-Hyune Rhee*

*Department of IT Convergence and Application Engineering, Pukyong National University

**Interdisciplinary Program of Information Security, Graduate School, Pukyong National University.

요 약

소셜 미디어는 소셜 네트워크를 기반으로 사용자간의 의견 및 정보 등을 공유하는 온라인 플랫폼이다. 사용자는 이를 통해 다양한 정보와 사용자들을 접할 수 있다. 교내 소셜 미디어 플랫폼도 기존의 소셜 미디어 플랫폼을 이용하여 운영하고 있다. 하지만 이러한 순기능과 달리 플랫폼의 구조적인 문제로 중앙화된 구조, 보상체계에 관한 문제점도 존재한다. 한편 블록체인 기반의 소셜 미디어 플랫폼인 엑스퍼티와 스팀잇은 사용자에 대한 보상체계가 존재하며 블록체인을 통해 탈중앙화를 이루었다. 본 논문에서는 중앙화된 소셜 미디어 플랫폼의 문제를 해결하기 위해 블록체인 기반의 소셜 미디어를 사용하여 보상체계를 제공하는 교내 소셜 미디어 플랫폼을 제안한다.

I. 서론

소셜 미디어(social media)는 소셜 네트워크를 기반으로 개인의 다양한 생각이나 의견, 정보 등을 공유하고 관계를 확장시킬 수 있는 온라인 플랫폼을 칭한다. 대표적인 플랫폼으로써 페이스북의 경우 사용자는 자신의 공간에 다양한 글을 포스팅하고 이를 다른 사람과 공유하는 것이 가능하다. 때로는 개인의 작은 사연이 소셜 미디어를 통해 전 세계에 공유되어 큰 파급력을 가져오기도 한다[1].

하지만 이러한 순기능과 달리 출처가 불분명한 잘못된 정보가 유포되거나 정부와 같은 중앙 기관의 규제로 공유하고자하는 정보가 검열되거나 [2] 중앙화된 플랫폼 구조의 문제로 개인정보가 오용 혹은 유출되기도 한다[3]. 많은 국내 대학 및 기업들에서도 별도로 이러한 소셜 미디어 플랫폼을 운영하고 있으나 앞서 언급한 문제와 동일한 문제를 가지고 있다. 때문에 본 논문에서는 이러한 중앙화된 소셜 미디어 플랫폼의 문제를 해결하기 위해 블록체인 기반 소셜 미디어 플랫폼인 엑스퍼티[Experty][4]와 스팀잇(stemmit)[5]을 사용하여 관리자가 없는 탈중앙화된 교내 소셜 미디어 플랫폼의 개발을 목적으로 사용자들이 자발적으로 잘못된 정보가 공유되는 것을 막고 커뮤니티 전체에 이익이 되는 컨텐츠의 생산을 독려하기 위한 사용자에 대한 보상체계에 대한 분석을 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 블록체인 기반 소셜 미디어 플랫폼인 엑스퍼티와 스팀잇에 대한 분석과 보상체계 및 교내 적용의 문제에 대해 서술하고 3장에서는 적용 가능한 다른 보상체계에 대한 분석과 교내 적용에 대한 문제에 대해 고찰하고 4장에서 결론을 맺는다.

II. 블록체인 기반 소셜 미디어 플랫폼

1). 엑스퍼티(Experty)[4]

엑스퍼티(Experty)는 블록체인 기술을 이용하여 전 세계 어느 곳에서나 지식 제공자와 지식을 찾는 사람들을 연결하도록 설계된 지식 교환 플랫폼이다. 두 명 이상의 사람들이 음성 또는 화상 통화를 통해 참여자들 사이에 유료 기반 전문 교환을 할 수 있도록 하는 것이 이 플랫폼의 목적이다. 엑스퍼티에서 지식 제공자는 직접 통화 연결을 수행하거나 QR코드를 통해 서비스를 광고할 수 있으며, 소셜 미디어 플랫폼에 쉽게 배치가 가능하다. 통화가 끝났을 시에 플랫폼의 미리 정해진 시스템에 따라 요금이 정산된다.

엑스퍼티에서 응용 프로그램 내의 오디오 및 비디오 통신은 WebRTC 프로토콜[6]을 사용한다. 또한 언제 어디서나 전문가가 유료 통화 링크를 공유가능하게 하기 위해서 분산화를 통해 통화를 연결하고, 제 3자 서비스 투명성을 확보하기 위해서 타 웹사이트에서 시작된 통화 추적을 가능하게 한다.

엑스퍼티에서는 지식 제공자가 미국 달러로 상담 요금을 설정하면 지식 탐색자는 USD/EXY 전환율에 따라 EXY토큰 양을 계산하여 자신의 계좌에 돈을 입금하고, 통화를 시작하여 지식을 획득한다. 통화가 끝난 후에 지식 제공자는 EXY 토큰을 지급 받는다.

엑스퍼티에서 지식을 제공하는 사람들은 그 분야에서의 전문가들이다. 따라서 지식을 신청한 사람들은 인터넷에서 단순히 검색을 했을 때보다도 훨씬 더 능동적으로 지식을 획득할 수 있으며, 통화 중에 모르는 것이 있다면 전문가에게 실시간으로 질문할 수 있다. 문서화된 정보보다 더 풍부한 정보를 획득할 수 있는 것이다. 그리고 지식 제공자는 실물 보상을 획득함으로써 더 많은 참여를 기대할 수 있게 된다. 표-1에 Token Generation Event Distribution summary를 보인다.

Maximum cap on crowdsale	33,000 ETH
Total token supply	**100 Million** (100,000,000) EXY, of which: ◊ 33% Crowd Sale Tokens ◊ 33% Company Tokens (vested for 3 years*) ◊ 30% Partner Tokens (locked for 18 months) ◊ 1% to cover tokensale costs ◊ 3% Airdrops and bounties
Ethereum ERC20	Experty is using the **ERC223 standard**, an updated version of ERC20. ERC 223 requires less Gas than ERC20, prevents the risk of tokens being lost when they are sent to an incompatible contract, and is also fully compatible with ERC20 for the purpose of exchange listings.
Purchase methods accepted directly	ETH
Purchase methods accepted via Bitcoin Suisse	ETH, BTC, CHF, USD, EUR, JPY

*Every month, beginning on February 2018, the contract will release 1/36 of the total EXY company tokens. All EXY company tokens will be released by February 2021.

[표-1]. Token Generation Event Distribution Summary

2). 스팀잇(Steemit)

스팀잇(steemit)[5]은 기존 소셜 미디어의 여러 가지 문제점들을 보완한 블록체인 기반 소셜 네트워크 서비스(Social Network Service)이다. 2016년 4월 서비스를 시작하였으며 개발자 스스로 컨텐츠의 공정한 보상을 위해 스팀잇을 만들었다고 밝힌 만큼 탈중앙화된 보상체계를 보유하고 있다.

스팀잇은 암호 화폐 블록체인인 스팀(STEEM)을 사용하고 블록체인은 20명의 증인(witness)들과 80명의 대기증인(backup witness)들에 의해 유지·관리된다. 사용자가 생산한 컨텐츠의 이미지와 동영상이 블록체인에 모두 기록되는 것이 아닌 컨텐츠의 URL만이 블록체인에 기록되고 이미지와 동영상은 아마존의 클라우드 서비스에 저장된다. 때문에 증인들이 운영하는 서버에는 상대적으로 작은 크기의 포인터 정보를 포함하는 블록체인의 사본들이 저장된다. 때문에 사용자가 컨텐츠를 스팀잇에 포스팅하면 이는 아마존 클라우드 서버에 저장되고 최종적으로 스팀 블록체인에 URL과 함께 기록되어 블록체인의 비가역성

(immutability)으로 인해 나중에 컨텐츠를 수정하는 것이 불가능해진다.

사용자들은 양질의 컨텐츠를 생산하거나 다른 사람의 컨텐츠에 대해 추천하는 행위(voting)를 통해 보상을 받을 수 있다. 양질의 컨텐츠에 대해 사용자들은 추천을 통해 보다 많은 사람들에게 공유할 수 있고 많은 추천을 받은 컨텐츠는 표-2과 같이 암호 화폐인 스팀코인으로 보상을 지급받는다. 이는 집단지성이 더 좋은 콘텐츠를 양산하는 선순환의 구조의 생태계를 구축한 사례로 볼 수 있다.

인플레이션율	증인	스팀 파워	저자	추천인	합계
	0.1	0.15	0.75		1
			0.5625	0.1875	
10%	0.01	0.015	0.05625	0.01875	0.1

[표-2]. 스팀잇 보상체계

하지만 본 논문에서 목적으로 하는 교내 소셜 미디어 플랫폼 구축에 암호 화폐를 사용한 보상체계를 적용하기에는 분명한 한계점이 존재한다. EXY토큰 과 스팀코인의 경우 거래소 등을 통해서 실물 화폐로 교환하거나 일부 매장에서 현금처럼 사용할 수 있다. 하지만 교내에서 이러한 별도의 암호 화폐를 만들어 배포할지라도 실물 보상으로 혹은 다른 보상으로 이를 활용하는데 있어서는 주변 상권이나 학교의 협업이 필수적이나 이 경우 탈중앙화된 플랫폼에 관리기관의 참여가 불가피해지므로 기존과 동일한 중앙화 이슈가 발생한다. 때문에 교내에 적용을 위해서는 소셜 미디어 플랫폼 사용자들 사이에서 자체적으로 외부의 제3자의 참여를 필요로 하지 않는 보상체계에 대한 분석이 필요하다.

III. 타 보상체계 분석

2장에서는 엑스퍼티와 스팀잇의 암호 화폐를 사용한 기존 보상체계가 교내 환경에 적용하기에는 적절하지 않음을 확인했다. 이 장에서는 교내 탈중앙화된 소셜 미디어 플랫폼의 활성화 방안을 위해 기존에 알려진 몇 가지 보상체계의 사례와 각 사례의 교내 적용에 대한 분석을 수행한다.

1) 네이버 지식IN

네이버가 운영하는 QnA 서비스로 2002년 이후부터 현재까지 운영되고 있으며 질문자가 질문을 올리면 이에 대해 누구나 답변을 할 수 있다. 하지만 이 역시 출처가 불분명한 정보나 누가 봐도 거짓인 정보를 답변으로 추가하는 문제가 발생하였고 네이버에서는 이를 해결하기 위해 답변자에 대한 등급제를 실시하였다. 질문자는 질문에 대한 답변들 중 하나를 채택하고 자체적인 마일리지 포인트인 내공을 답변자에게 줄 수 있는데 답변 채택율이 높고 채택 답변수가 일정 수 이상이며 보유한 내공이 많은 답변자의 경우 하수~절대신 사이의 등급을 부여받는다. 실제로 내공이나 등급에서 얻는 혜택은 크지 않으나 이러한 등급제는 외에도 많은 QnA 서비스에서 채택하고 있는 일종의 명예적 보상체계이며 일부에서는 우수한 사용자를 선정하여 물질적 보상을 지급하기도 한다.

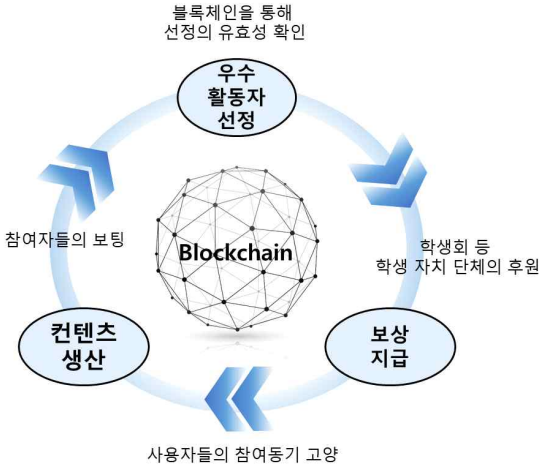
2장에서는 교내 소셜 미디어 플랫폼의 경우 중앙기관이 플랫폼에서 과도한 권한을 가지는 것을 방지하기 위해 자체적인 보상체계가 필요함을 서술했었다. 이러한 명예적 보상은 별도의 보상 지급 기관이나 추가 비용 없이 자체적으로 운용 가능하므로 실제 구현 관점에서 현실적이나 네이버의 사례에서 전체 사용자의 참여 유도는 불가능하여 알려진 우수답변자 그룹 외에는 참여 활동이 전무한 상태이다.

2) 인센티브 제도

인센티브 제도의 경우 앞선 명예적 보상과 달리 보상지급의 주체(중앙관리기관)이 존재하고 스팀잇의 보상체계와 달리 활동이 우수한 사용자 중 일부를 선정하여 인센티브를 제공한다. 하지만 선정 기준이 중앙 기관에 의존적이고 선정 과정이 투명하지 않다는 문제가 존재하지만 실물 화폐를 인센티브로 제공하기에 사용자의 참여유도 동기 정도는 1)의 명예적 보상보다 높으며 보상을 지급하는 중앙 기관이 존재하는 국내 정부부처 등에서 가장 많이 활용하는 보상체계이다.

앞서 서술한 두 가지 방법 1) 명예적 보상, 2)

인센티브 제도를 교내 소셜 미디어 플랫폼에서 양질의 컨텐츠의 생산과 참여를 유도하기 위한 보상으로 사용하기 위해서 본 연구에서는 그림 1과 같은 보상체계를 제안한다.



[그림-1]. 제안하는 보상체계의 선순환 구조

사용자들은 컨텐츠를 생산하고 컨텐츠에 대해 다른 사용자들은 양질의 컨텐츠의 경우 추천을 통해 다른 사용자에게 공유한다. 일정기간 마다 블록체인에 기록된 개인 사용자별 추천 기록을 확인하여 우수 사용자를 선정하여 플랫폼에서 보여지는 명예적 보상인 개별등급을 부여하고 이 중 우수 사용자(상위 10% 이내 등)에 대해서는 학생 자치단체 혹은 후원 단체 등에서 실물 보상을 지급하는 복합적인 보상체계를 제안한다. 보상의 규모 및 지급 기준, 지급 시기 등은 후원 및 자치 단체의 예산에 따라 상이하나 학교 본부 등 중앙기관의 참여 없이 가능하다는 점에서 보다 현실적인 구현이 가능하다.

IV. 결론

본 논문에서는 블록체인 기반의 소셜 미디어 플랫폼 엑스퍼티와 스팀잇을 사용하여 탈중앙화 되고 보상체계가 존재하는 교내 소셜 미디어 플랫폼을 제시한다. 이 교내 소셜 미디어 플랫폼은 참여자의 추천을 통해 우수 활동자 선정 및 일부 우수 활동자에게는 학생회 등 학생 자치 단체에서 후원을 하는 것으로 학교 본부와 같은 중앙기관의 참여 없이 현실적인 구현이 가능하다.

향후 본 연구에서 제시된 보상체계를 통해 블록체인 기반의 교내 소셜 미디어 플랫폼을 구현할 것이다.

[감사의 글]

본 논문은 부경대 링크플러스 사업단의 지원을 받아 수행된 연구결과 입니다.

[참고문헌]

[1]http://www.sisapress.com/journal/article/155804

[2]https://ko.wikipedia.org/wiki/%EC%8B%9C%EB%82%98_%EC%9B%A8%EC%9D%B4%EB%B3%B4

[3]https://www.boannews.com/media/view.asp?idx=67851

[4] Experty-whitepater

[5] steem-whitepaper

[6]https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols

딥러닝을 이용한 자동 취약점 분석1)

이재융*, 석재혁*, 이동훈*
*고려대학교 정보보호대학원 정보보호학과

Automatic vulnerability analysis using Deep Learning

Jae-Yung Lee*, Jae Hyuk Suk*, Dong Hoon Lee*
*Korea University, Graduate School of Information Security.

요 약

IT기술의 발전에 따라서 사람들에게 편의성을 제공하기 위한 수많은 소프트웨어가 쏟아지고 있다. 기술에 발전과 같이 해커들의 기술또한 발전을 하여 다양한 취약점들을 발견하고 이를 이용한 공격들은 사람들의 개인정보를 탈취해간다. 이러한 취약점들은 보안지식을 가진 전문가들이 발견할 수가 있지만 발견하기 까지 많은 시간이 소요되고 인력 비용 또한 만만치 않다. 따라서 본 논문에서는 이러한 단점들을 개선하기 위한 딥러닝을 이용한 자동 취약점 분석들을 소개한다.

I. 서론

매년 사람들에게 편의성 혹은 다른 목적을 위해서 생산되는 소프트웨어양이 기하급수적으로 많아지고 있다. 하지만 생산되는 양만큼 취약점들 또한 나날이 진화하고 발견된다. 또한 사람들이 수동으로 취약점을 분석하기에는 매우 많은 소프트웨어가 있고, 인건비 그리고 취약점을 분석하는 시간도 고려해야함으로 매우 비효율적이다. 따라서 자동으로 취약점을 분석하는 시스템이 필요로 하고 전 세계적으로 연구가 활발히 진행 중에 있다. 최근 자동 취약점 분석연구에 눈에 띄었던 활동이라면 당연히 CGC(Cyber Grand Challenge)라고 할 수가 있다. 사람의 개입이 전혀 없고 오직 기계가 바이너리 파일을 분석을 하며 자동으로 공격 쉘코드를 작성한다. 이러한 연구는 자동 취약점분석에 가능성을 보여주었고 전 세계 소프트웨어 보안 연구원들에게 촉진제가 되었다. 하지만 CGC에서 보여주었던 자동 취약점 분석에는 아직까지 많은 단점들이 있다. 따라서 본 논문에서는 어떤 단점들이 있고. 이러한 단점들을 극복하기 위해서 나온 딥러닝을 활용한 자동 취약점 분석을 소개하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 AEG에 대해 설명하고, 3장에서는 딥러닝을 이용한 자동 취약점 분석을 설명하고 4장에서 마무리를 짓는다.

II. AEG(Automatic Exploit Generation)

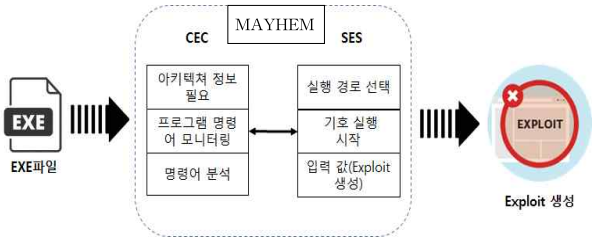
2016년 미국 DARPA에서 주최한 CGC에서는 기계가 자동으로 프로그램의 취약점을 찾아내고 exploit까지 만들어내는 기계설계 대회를 열었다. 이 대회에서 우승팀의 MAYHEM[3]은 다음 [그림 1]과 같이 구현을 하였다.

1. MAYHEM에서는 익스플로잇을 생성하기 위하여 기호실행[7]을 이용한 두 개의 모듈을 사용한다: Concrete Executor Client(CEC)는 명령어를 제어흐름을 변경하는 명령어(Ex: jmp, call)가 실행될 때 마다 이를 분석 모듈인

1) 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

Symbolic Executor Server(SES)에 전달하며, SES에서는 전달받은 명령어를 중간언어로 변환한다.

2. SES는 변환된 중간언어에 만족하는 실행경로식[5]을 생성하고 이를 기호실행에 입력하여 익스플로잇을 생성한다.



[그림 1] MAYHEM의 동작 과정

MAYHEM과 같은 방식의 자동 취약점 분석 같은 경우에 높은 정확도와 기계의 완성이후에 사람개입이 적게 되는 장점이 있다. 하지만 이러한 기계를 완성하기 위해서는 소프트웨어에 깊은 지식이 필요로 하고 많은 시간 그리고 비용 면에서 단점을 가지고 있다. 따라서 이러한 단점들을 개선하기 위해서 딥러닝을 이용한 자동 취약점 분석이 필요하다.

III. 딥러닝을 이용한 자동 취약점 분석

인공지능분야와 보안 분야가 서로 아무 의미가 없는 거 같지만. 최근에 보안 분야에서 딥러닝을 이용하여 다양한 연구결과 물들이 나왔었다. 예를 들어, 딥러닝을 활용한 악성코드 탐지, 악성코드 분류등 많은 연구가 진행되었었다. 특히 NDSS 및 이외의 학회에서 나온 딥러닝을 이용한 자동 취약점 분석은 소프트웨어 보안 분야에 큰 이슈를 몰고 왔다. 여기서 취약점 기준으로 탐지와 분석에 대하여 간단하게 설명하자면 다음 [표 1] 와 같다.

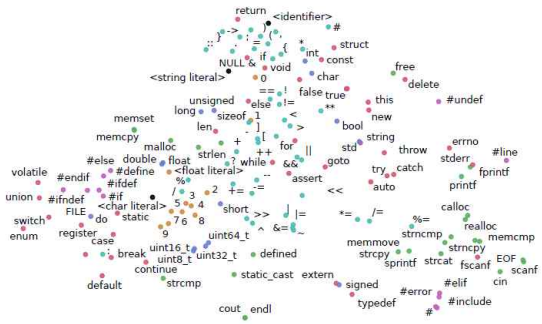
취약점 탐지	취약점 분석
검사하려는 코드가 악성인지 아닌지 판단만 하는 과정. 어디에서 취약점이 발생하는지는 고려를 안 한다	검사하려는 코드가 악성인지 아닌지 판단함과 동시에 취약점이 어디에서 발생하는지 알 수가 있다

[표 1] 취약점 탐지와 분석의 차이

딥러닝을 이용한 취약점 분석방법을 알기 전에 몇 가지 배경지식을 짚고 넘어간다.

1. Word2Vec

발음과 동일하게 word to vector, 즉 단어를 벡터화 시킨다는 알고리즘이다. 단어자체는 사람이 이해할 수가 있지만 컴퓨터가 이해하기에는 어려움이 있어 이를 보완하기 위해 단어를 벡터화 시키는 과정이다. [그림 2]은 C언어에 나오는 단어들을 word2vec을 적용한 그림이다.



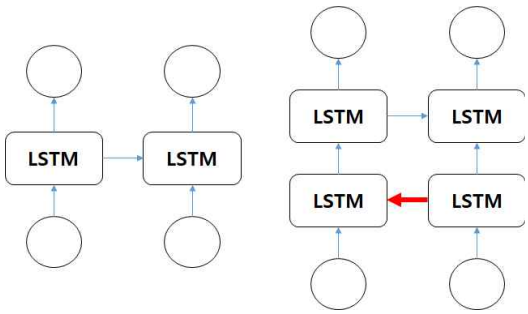
[그림 2] C언어 Word2Vec 적용결과

2. Bag of Words

word2vec과 다르게 Bag of Words는 단순하게 단어의 빈도수만을 고려하여 단어를 벡터화 시킨다. 따라서 Bag of Words는 간단하게 단어사전이라고 생각하면 편하다.

3. BLSTM

BLSTM은 Bidirectional Long Short Term Memory의 약자로 자연어 처리 알고리즘중 하나인 LSTM[8]에서 진화된 딥러닝 알고리즘이다. 이는 LSTM에서 앞의 노드가 뒤에 노드에만 영향을 미치는 구조를 바꿔서 양방향성으로 영향을 미치게끔 만든 알고리즘이다. [그림 3]은 LSTM과 BLSTM의 구조 그림이다.



[그림 3] LSTM(좌), BLSTM(우) 구조도

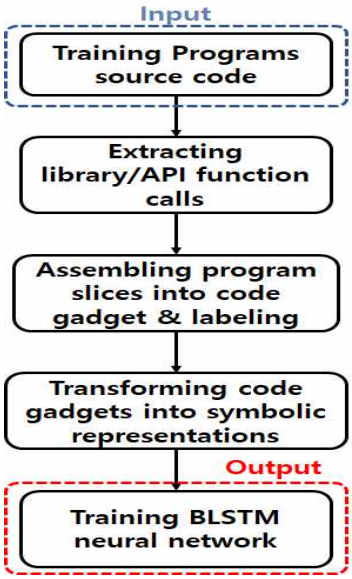
3. CNN[6]

CNN은 Convolutional Neural Network의 약자로 딥러닝 분야에서 흔히 이미지 분류를 할 때 쓰이는 알고리즘이다. CNN에서는 convolution 이라는 과정을 이용하여 사람이 직접 특징점을 잡

아내지 않아도 기계가 계산을 통하여 이미지의 특징점을 잡아낸다. 이후 pooling작업을 거쳐서 여러 layer들을 연결하여 유의미한 데이터를 추출한다.

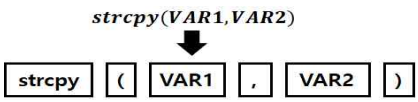
3.1 BLSTM을 이용한 취약점 분석

NDSS에서 발표된 연구에 따르면 다음과 같이 자동취약점 분석 도구를 만들었다고 한다.

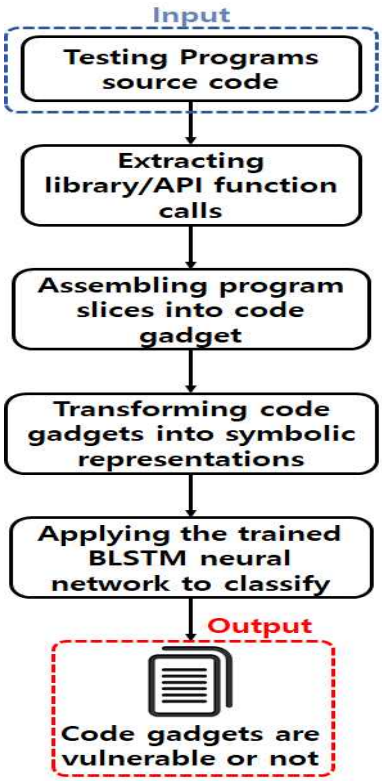


[그림 4] [4]의 학습 과정

[4]에서 제안하는 모델의 학습 과정도는 [그림 4]와 같다. 먼저 딥러닝 모델에 학습시킬 데이터들을 준비한다. 이때 데이터 셋은 소스코드로 이뤄져있고 정상코드 취약점이 내포된 코드 두 개로 나눠진다. 두 번째로 소스코드 내부에 API함수들을 추출하고 세 번째로 추출한 소스코드들을 gadget형식으로 분리하고 라벨링(정상:0, 취약점내포:1)을 진행한다. code gadget처리 과정은 [그림 5] 와 같이 진행된다. 네 번째로 분리된 code gadget들을 word2vec을 이용하여 벡터화 시킨 뒤 BLSTM 딥러닝 모델을 이용하여 학습 시킨다.



[그림 5] Code gadget 처리 과정



[그림 6] [4]의 검증 과정

앞에서 설명한 학습과정을 마친 뒤 테스트 과정은 [그림 6]와 같다. 세 번째 과정에서 라벨링 작업만 빼면 테스트과정이 된다. 추출된 code gadget을 이용하여 학습된 모델에 넣어주고 결과로 해당 code gadget이 취약점을 내포하는지 알 수가 있다.

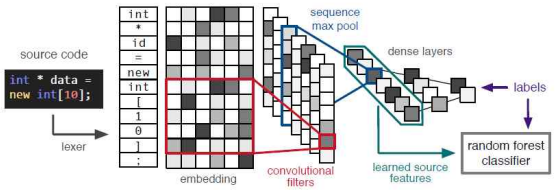
3.2 TextCNN을 이용한 취약점 분석

소스코드를 이용하여 코드 단어들을 모아둔 리스트를 사용하여 사용한 단어는 word2vec으로 학습하여 코드 단어들을 값으로 설정하여 여러 줄의 코드를 하나의 이미지로 설정한다.

이미지로 만들기 전에 소스코드에는 다양한 데이터 형식이 있으므로 이를 통일하는 과정이 필요하다. 따라서 중복되는 데이터 형식은 하나로 통일한다. 예를 들어 C언어에서 u32, uint32_t, UINT32, uint32, DWORD같은 경우에는 똑같이 32비트 unsigned데이터 형식이기 때문에 하나로 통일한다.

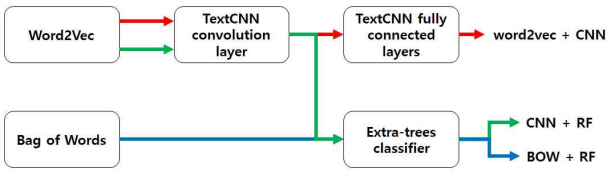
word2vec과 TextCNN을 사용한 자동 취약점 분석을 두 가지 방법론을 설명하겠다. 첫 번째 방법론 같은 경우에는 TextCNN과 Random

Forest모델을 사용하여 취약점 유무를 판단하였다. 해당 과정은 [그림 7]와 같다.



[그림 7] [1]의 모델 구조도

두 번째 방법론 같은 경우에는 첫 번째 방법론과 유사하지만 이외로 Bag of words라는 벡터화 알고리즘도 사용하여 다양한 실험결과를 제시한다. 본 논문에서는 [그림 8]와 같이 총 세 가지 방법을 이용하여 실험을 진행했는데 이는: 1. word2vec과 CNN을 이용한 실험, 2. CNN과 Random Forest을 이용한 실험, 3. Bag of words와 Random Forest를 이용한 실험을 소개한다.



[그림 8] [2]에서 실험한 모델

결과적으로 [표 3]와 같이 CNN와 Random Forest를 이용한 실험이 가장 효율이 좋게 나왔다.

Model	ROC AUC	P-R AUC
BoW + RF	0.85	0.44
W2V+CNN	0.87	0.45
CNN + RF	0.87	0.49

[표 3] [2]의 정확도 결과 표

IV. 결론

본 논문에서는 여러 가지 방법을 이용한 자동 취약점 분석에 대해서 알아보았다. 활발한 연구가 있었지만 아직까지도 완벽하지 않으며 무한한 발전가능성이 있는 연구 분야이다. 앞에서 설명한 여러 방법들의 단점들을 개선시키는 건 필수이며 이후 자동으로 탐지된 취약점들을 수정하는 것 까지 연구가 필요하다고 생각된다.

[참고문헌]

[1] Rebecca L. Russell, Louis Kim, Lei H. Hamilton, Tomo Lazovich, Jacob A.Harer, Onur Ozdemir, Paul M. Ellingwood, Mare W. McConley. Automated Vulnerability Detection in Source Code Using Deep Representation Learning, arXiv 2018

[2] Jacob A. Harer, Louis Y. Kim, Rebecca L. Russell, Onur Ozdemir, Leonard R. Kosta, Akshay Rangamani, Lei H. Hamilton, Gabriel I. Centeno, Jonathan R. Key, Paul M. Ellingwood, Erik Antelman, Alan Mackay, Marc W. McConley, Jeffrey M. Opper, Peter Chin, Tomo Lazovich. Automated software vulnerability detection with machine learning. arXiv 2018

[3] S.K Cha, T.Avgerinos, A.Rebert and D.Brumley. Unleashing MAYHEM on Binary Code. IEEE Symposium on Security and Privacy, 2012

[4] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng and Yuyi Zhong, VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. Network and Distributed Systems Security(NDSS) Symposium, 2018

[5] www.seas.harvard.edu/courses/cs252/2011sp/slides/Lec13-SymExec.pdf

[6] https://en.wikipedia.org/wiki/Convolutional_neural_network

[7] https://en.wikipedia.org/wiki/Symbolic_execution

[8] https://en.wikipedia.org/wiki/Long_short-term_memory

딥러닝 기반 이미지 스팸 검출 방법 연구

이정열, 김형식
성균관대학교 컴퓨터공학과

요 약

기술이 발전됨에 따라 사람들은 시간과 장소에 제약 없이 정보를 접하고 있다. 이에 따라 사람들이 원하지 않는 스팸에 접하는 횟수도 많아지고 있다. 일반적으로 필터링이 발달한 스팸 문자에 비해 스팸 이미지는 이미지를 변형함으로써 원본 이미지와 구별하기 힘들어진다. 본 논문은 구글의 딥러닝 오픈소스인 Tensorflow 기반의 스팸 이미지를 구별하는 모델을 설계한다. CNN으로 모델을 설계해 스팸 이미지를 구별하였으며, layer개수를 조절하여 8개의 모델로 실험한 결과 86%의 정확도를 얻을 수 있었다. 향후 더 많은 데이터 학습, 스팸 이미지의 특징을 잡아낼 수 있는 알고리즘 보완을 통해 스팸 이미지를 구별 할 수 있을 것으로 기대된다.

I. 서론

기술이 발전하고 정보화시대가 진행되면서 사람들은 스마트폰, PC 등을 이용해 시간과 장소에 제약 없이 정보를 접하고 있다. 정보를 접하는 기회가 많아짐에 따라 광고, 바이러스 등을 목적으로 하는 스팸에 노출되는 횟수도 많아지고 있다. 최근 방송통신위원회의 2018 상반기 스팸 유통현황 발표에 따르면, 17년 하반기 대비 휴대전화 문자 스팸의 양이 538만건에서 632만건으로, 이메일 스팸의 양이 4618만건에서 5109만건으로 각각 17.4%, 10.6% 증가된 것으로 나타났다 [1]. 문자열 필터링을 이용해 이를 차단하는 노력이 진행되고 있지만, 이미지 변형으로 인한 이미지 스팸의 대처에는 어려움을 겪고 있다.

최근 인공지능 기술이 발달하고 있고, 다양한 딥러닝을 활용하는 라이브러리가 오픈소스로 공개되고 있다. 이 중 구글에서는 딥러닝과 심층신경망 연구용인 Tensorflow를 오픈소스로 공개하였고, 본 논문에서는 Tensorflow 기반의 Convolution Neural Network 모델을 구성하여 스팸을 구별하는 학습 모델을 설계하였다 [2]. 1 ayer개수를 조절하여 8개의 모델로 실험한 결과 86%의 정확도를 얻을 수 있었다. 더 많은 학습으로 높은 정확도로 스팸 이미지를 구별한다면, 다양한 방식으로 증가하고 있는 이미지 스팸에 대한 대처가 가능하고, 스팸 이미지 유통을 줄일 수 있을 것이다.

II. 관련 연구

2.1 텍스트 스팸

텍스트 스팸 필터는 특징 단어를 필터링하는 방식을 기반으로 한다. 많은 특징 단어로 인해 광범위한 용량과 계산 시간이 필요하고, 성능을 향상시키기 위한 연구가 진행 중이다. 첫 번째로 특징 단어의 용량을 줄이기 위해 의미를 추출해, 여러 단어를 하나의 제목으로 묶어 계산 시간과 성능을 향상 시키는 것이다 [3]. 두 번째는 Tensorflow 기반으로 특징 단어를 효과적으로 분석하기 위한 단어 순서 기반의 OPCNN

딥러닝으로 기존 CNN보다 정확도를 높이는 연구이다 [4]. 세 번째는 BP neural network 모델을 이용해 전자 메일 텍스트를 분류해 딥러닝하는 연구가 진행되었다 [5].

이런 연구들을 비교해보면, 특징 단어를 필터링 하는 방식을 기반으로 여러 네트워크 모델로 딥러닝을 이용해 성능을 향상시키는 연구가 진행됨을 알 수 있다. 이처럼, 이미지 스팸도 특징 이미지를 잡아 필터링을 한다면, 딥러닝을 이용해 스팸을 분류할 수 있을 것이다.

2.2 이미지 스팸

초기 스팸은 텍스트 기반으로, 필터로 쉽게 검색 할 수 있었다. 이런 탐지를 피하기 위해 이미지에 텍스트를 포함시키는 이미지 스팸이 생성되었다 [6]. 대부분 기존 텍스트 스팸 필터링 기술로 잡을 수 없는 스팸이고 점점 정교해지기 때문에 이를 분석하는 연구가 필요하다.

2.3 Convolution Neural Network

CNN은 이미지의 특징을 추출하는 Feature Extraction과 그 특징을 분류하는 Classification으로 나누어진다. Feature Extraction은 Convolution Layer와 Subsampling Layer로 이루어져 있다.

2.3.1 Convolution Layer

이미지는 픽셀 하나하나가 RGB 3개의 실수로 표현된다. Convolution Layer에서는 각 채널별로 이미지 각각의 픽셀을 실수로 표현하고, Fig 1과 같이 일정 크기의 필터와 Convolution이 실행된다.

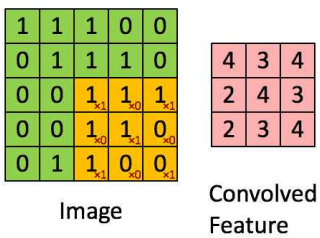


Fig 1. Convolution [7]

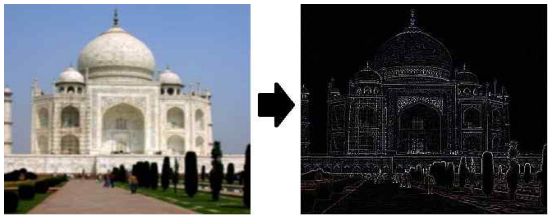


Fig 2. Convolution Result [8]

이 Convolution 과정을 통해 필터의 개수에 따라 이미지의 특징을 추출하게 되고, 실제 이미지는 Fig 2처럼 이미지의 윤곽을 구별하는 이미지로 변형된다.

2.3.2 Subsampling Layer

Subsampling Layer에서는 Convolution Layer의 크기를 줄이거나, 특징을 강조하는 시행이 이루어지고, CNN에서는 Fig. 3처럼 Convolution Layer의 출력 값에서 가장 큰 값을 선택하는 Max Pooling이 이루어진다. 이 과정에서 기존 이미지와 크기 변형이 이루어지고, 패딩을 통해 기존 이미지 크기와 같아진다.

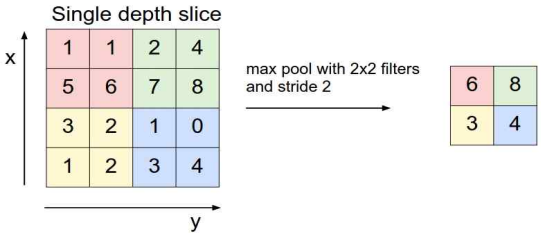


Fig 3. Max Pooling [9]

2.3.3 Fully-connected Layer

여기서 추출된 특징을 Classification에서 분류하게 된다. Fully-connected Layer에서는 사용자가 원하는 특징 수로 분류를 한다. Feature Extraction에서 Convolution과 Max Pooling 과정을 반복하면 점점 많은 특징으로 나누어지고, 특징이 세분화된다.

III. 본론

3. 1 Data set

학습에 사용할 spam 이미지 7,852개와 일반 이미지 917개를 각각 다른 소스로 수집하였다[10][11]. 이 이미지들을 학습 용도와 증명 용도로 분류 한 뒤 cifar-10 dataset과 같은 형태인 Label+RGB로 만들어 train_data.bin, eval_data.bin 파일로 생성하였다 [12].

3. 2 Convolution Neural Network Model 구성

Fig 4. Convolution Layer

```
# 첫번째 합성곱
w1 = weight_variable([kernel_size,kernel_size,
                      channel,img_size])
b1 = bias_variable([img_size])
L1 = tf.nn.relu(conv2d(images,w1)+tf.cast(b1, tf.float32))
L1 = max_pool(L1,pool_size,pool_size)
prev_img_size = img_size
img_size *=2
# 두번째 합성곱
w2 = weight_variable([kernel_size,kernel_size,
                      prev_img_size,img_size])
b2 = bias_variable([img_size])
L2 = tf.nn.relu(conv2d(L1,w2)+tf.cast(b2, tf.float32))
L2 = max_pool(L2,pool_size,pool_size)
prev_img_size = img_size
img_size *=2
# 세번째 합성곱
w3 = weight_variable([kernel_size,kernel_size,
                      prev_img_size,img_size])
b3 = bias_variable([img_size])
L3 = tf.nn.relu(conv2d(L2,w3)+tf.cast(b3, tf.float32))
L3 = max_pool(L3,pool_size,pool_size)
```

본 논문에서는 칼라 이미지를 사용하였기 때문에 채널수는 3, pool_size를 2로 설정하여 layer가 늘어남에 따라 이미지 크기가 절반으로 줄어들도록 설정하였다. Convolution의 Layer의 층수에 관계없이 출력 이미지의 개수는 입력 이미지의 2배로 설정하였다. 다양한 성능을 확인하기 위해 layer의 개수는 3~6으로 조절하였다.

Fig 5. Full_connected Layer

```
# full_connected,
w_fc1 = weight_variable([16*16*128,fc_size])
b_fc1 = bias_variable([fc_size])
x_fc1 = tf.reshape(L3,[-1, 16*16*128])
L_fc1 = tf.nn.relu(tf.matmul(x_fc1, w_fc1)
        +tf.cast(b_fc1, tf.float32))
keep_prob = tf.placeholder(tf.float32)
L_fc1 = tf.nn.dropout(L_fc1,keep_prob)
w_fc2 = weight_variable([fc_size,2])
b_fc2 = bias_variable([2])
```

Full connected Layer에서는 Convolution Layer에서 나누어진 Shape를 입력 값으로 받고, 최종적으로 분류할 특징의 수로 출력 값을 정한다. 본 논문에서는 스팸인지 아닌지를 구분하기 위해 출력 값을 2로 설정하였다. Convolution Layer에서 나눠진 특징을 딥러닝하여 스팸인지 아닌지 구분 할 것이다.

3. 3 실험 결과

주어진 공통 데이터로 모델을 구성할 때 layer의 개수에 따라 특징을 잡는 기준이 달라져, 정확도가 차이가 있을 수 있기 때문에 본 논문에서는 Table 1과 같이 Convolution Layer의 개수를 3~6개로, Full connected Layer의 개수를 1~2개로 조절하며 학습하였다.

설계한 모델로 10,000번의 학습을 진행하였고, 이 결과를 Tensorboard로 출력하였다. 모든 모델들이 처음에는 높은 loss값을 가졌지만, 학습이 진행됨에 따라 fig 8처럼 0에 가까워지게 좋은 학습 결과를 가져왔다.

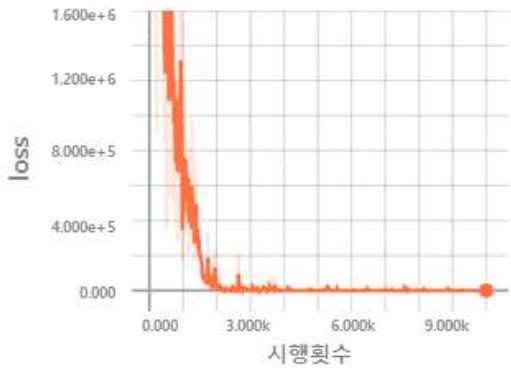


Fig 6. 시행횟수에 따른 loss값

좀 더 세밀한 성공률을 확인하기 위해 Accuracy와 Precision, Recall을 계산하였다. 학습된 데이터를 바탕으로 검증을 실행한 결과 86%의 정확도를 가지는 모델을 설계할 수 있었다. Layer의 개수를 바꿔가며, 학습하고 정확도를 비교한 결과는 Table 1과 같다.

Table 1. 실험결과

conv/fc	accuracy	precision	recall
3/1	**86.0%**	**78.9%**	**88.9%**
3/2	30.0%	12.6%	28.3%
4/1	74.3%	58.7%	75.3%
4/2	27.1%	13.3%	19.1%
5/1	44.9%	29.1%	46.9%
5/2	31.9%	22.6%	45.2%
6/1	41.1%	21.8%	40.7%
6/2	66.7%	26.2%	73.9%

정확도를 높일 수 있는 첫 번째 방법은 더 많은 학습 데이터양이다. 스팸 이미지는 공개되어있는 이미지보다 개인 문자, 이메일 등 Private한 곳으로 전해지기 때문에 학습 데이터를 수집하기에 어려움이 있다. 두 번째 방법은 명확한 특징 잡기이다. 스팸 이미지는 텍스트 이미지와 다르게 명확한 특징을 잡기 어렵다는 것이다. 텍스트를 추가한다거나, 선정적인 그림을 넣는 것 등 다양한 이미지 특징이 있기 때문에 단순한 색, 모양적인 특징으로 특징 이미지를 정의하기 힘들다.

IV. 결론

본 논문에서는 구글의 딥러닝 오픈소스인 Tensorflow 기반의 Convolution Neural Network 모델을 설계하고 딥러닝을 통해 스팸 이미지를 얼마나 정확하게 구별할 수 있는지 분석하였다. 총 8개의 모델을 설계하여 분석하였고, 최고 86%의 정확도를 가지는 모델을 획득하였다. 더 높은 정확도를 얻기 위해서는 많은 Dateset 확보를 통한 많은 딥러닝 학습이 이루어져야 할 것이다. 학습을 통해 더 높은 정확도를 가질 수 있다면, 스팸 이미지의 확산을 예방 할 수 있을 것이다. 향후 연구에서는 스팸 이미지는 다른 이미지와 다르게 색, 모양 등으로 특징을 잡기 힘들기 때문에, 주어진 이미지를 스팸 이미지로 구별 할 수 있는 알고리즘을 개발하여 학습을 좀 더 쉽고 정확하게 할 수 있는 방법을 연구할 것이다.

[참고문헌]

[1] Korea Communication Commision. https://kcc.go.kr/user.do?mode=view&page=A05030000&dc=K05030000&boardId=1113&cp=1&boardSeq=46251

[2] Tensorflow. https://www.tensorflow.org/

[3] Wei Hu, Jinglong Du, and Yongkang Xing. Spam Filtering by Semantics-based Text Classification, in: 2016 Eighth International Conference on, pp 89-94

[4] Siyuan Zhao, Zhiwei Xu, Limin Liu, Mengjie Guo, Jing Yun. Towards Accurate Deceptive Opinions Detection based on Word Order-preserving CNN, in: Mathematical Problems in Engineering, Vol 2018 (2018)

[5] Qiuming Ma, Zhiguang Qin, Fengli Zhang, Qiao Liu. Text Spam Neural Network Classification Algorithm, in: 2010 International Conference on Communications, Circuits and Systems (ICCCAS), pp 466-469

[6] Khanum, Mohammadi Akheela. Trends in Combating Image Spam E-mails, in: ICFIT 2012

[7] Deep Learning Stanford. http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

[8] WILDML. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[9] Stanford. http://cs231n.github.io/convolutional-networks/

[10] Mark Dredze, Reuven Gevaryahu, Ari Elias-Bachrach. Learning Fast Classifiers for Image Spam: Conference on Email and Anti-spam(CEAS) 2007

[11] Y. Gao, M. Yang, X. Zhao, B. Pardo, Y. Wu, T. N. Pappas, A. Choudhary, Image spam hunter, in: International Conference on, IEEE, 2008, pp 1765-1768.

[12] cifar 10. https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10

Long Short-Term Memory를 포함한 Autoencoder 기반의 네트워크 침입 탐지 시스템

안선우, 이하윤, 김기열, 김현준, 백윤흥*

*서울대학교 전기정보공학부

Autoencoder-based Network Intrusion Detection System Using LSTM

Sunwoo Ahn, Hayoon Yi, Giyeol Kim, Hyunjun Kim, Yunheung Paek*

*Department of Electrical and Computer Engineering, Seoul National University.

요 약

네트워크 기반 침입 탐지 시스템(NIDS)에 인공신경망을 쓰는 연구들이 증가해왔다. 하지만 인공신경망을 사용하는 기존의 NIDS들은 지도학습을 사용하거나, 시계열 정보를 잘라 사용하는 문제들이 있었다. 본 연구에서는 비지도학습으로 학습할 수 있는 인공신경망의 일종인 autoencoder를 LSTM 기반으로 구성하여, 이러한 문제점을 해결하고자 했다. 또한, 네트워크 패킷에서 데이터를 추출하여 제시된 모델을 학습할 수 있도록 하였다.

I. 서론

스마트폰 등장 이후 사용하는 대부분의 전자기기들이 네트워크를 사용하고 있다고 해도 과언이 아니다. 이에 따라 네트워크를 통한 공격이 더 다양해지고 복잡해지고 있다. 네트워크를 통한 공격을 방어하기 위해 흔히 쓰이는 방법 중 하나가 네트워크 기반 침입 탐지 시스템(NIDS)이다.

지난 10년간 기계 학습 기술이 크게 발전함에 따라, 많은 분야에 기계 학습이 도입되고 있다.[1][2][3] NIDS는 기계 학습이 도입되어 활발히 연구되는 사례 중 하나이다. 특히, 기계 학습 중에서도 가장 활발히 적용되고 있는 인공신경망(ANN) 역시 NIDS에 적용하려는 시도들이 있다. 하지만 기존의 ANN 기반의 NIDS는 (1) 지도학습을 통해 모델을 생성하기 때문에, 전문가가 데이터에 labeling을 하는 과정이 필요하여 human effort를 줄이려는 기계 학습 취지에 맞지 않으며, (2) 시계열 정보인 네트워크 패킷을 해석하기 위해 패킷을 특정한 윈도우 단위로 자르게 되는데, 이 때문에 연관된 패킷이 멀리 떨어진 경우 침입을 탐지하기 어렵다.

[4]에서는 (1)의 문제를 해결하기 위해 autoencoder를 도입하였지만, 여전히 (2)의 문제는 남아있다. 본 연구에서는 이를 해결하기 위해 Long Short-Term Memory(LSTM)[5] 기반의 autoencoder를 NIDS에 도입하고자 한다. LSTM을 이용하면 윈도우를 두지 않고 시계열 정보를 해석할 수 있어, (2)의 문제를 해결할 수 있다. 따라서, 본 연구는 LSTM을 NIDS의 데이터와 목적에 맞게 autoencoder에 포함시켜 신경망을 구성하는 것에 초점을 맞췄다.

II. LSTM 기반의 autoencoder를 이용한 NIDS

이 장에서는 본 연구에서 제시하는 LSTM 기반의 autoencoder의 신경망 구성과 이를 이용

특성 1	특성 2	특성 3	특성 4	특성 5
timestamp delta	패킷 크기	프로토콜	특수 IP 종류	IP class A
특성 6	특성 7	특성 8	특성 9	특성 10
IP class B	IP class C	IP class D	외부 port	호스트 port

표 1 네트워크 패킷에서 추출한 feature 목록

한 NIDS에 대해 소개한다. 또한, 제시된 NIDS가 어떻게 동작 과정도 소개한다.

2.1 개관

본 연구에서 제시하는 NIDS는 네트워크 패킷에서 추출한 데이터로 모델을 학습 및 추론을 하여 동작한다. 이 NIDS는 크게 세 가지 요소로 구성된다.

- **패킷 분석기**: 네트워크 패킷을 받아 분석하여 전처리기로 보내준다. 외부 라이브러리인 tshark[6]를 사용한다.
- **데이터 전처리기**: 패킷 분석기로부터 받은 데이터를 가공한다. 이 과정에는 네트워크 패킷을 분류하는 과정, 원하는 특성을 추출하는 과정, 이상 행위 탐지기의 입력 데이터 형태에 맞게 고치는 과정이 포함된다.
- **이상 행위 탐지기**: 데이터 전처리기로부터 받은 입력 데이터로 이상 행위를 탐지를 학습/시행한다.

패킷 분석기는 외부 라이브러리를 사용했기 때문에 데이터 전처리기와 이상 행위 탐지기에 대한 설명에 집중한다. 또한, ANN 기반의 이상 행위 탐지에서 데이터 전처리와 이상 행위 탐지기 역할을 하는 모델의 학습/추론은 일반적인 행위이므로, 본 논문에서는 네트워크 패킷을 분류하는 방법, 어떠한 특성을 추출하였는지, 모델의 구성에 대하여 설명하는 데 집중한다.

2.2 데이터 전처리기

데이터 전처리는 패킷 분석기에서 받은 데이터를 이상 행위 탐지기의 입력 데이터 형태에 맞게 고쳐주기 때문에 필수적이고, 어떤 특성을 학습에 이용할지 정하기 때문에 매우 중요하다. 이 과정에서 해결해야 할 문제 중 하나는 네트워크 패킷을 분류하는 작업이다. NIDS가 동작하는 기기에 다양한 포트로 들어오는 네트워크 패킷을 구분 없이 시계열 데이터로 만든다면, 서로 다른 목적을 가지고 들어온 네트워크 패킷이 한데 섞여 학습이 어려워지기 때문이다.

문제 해결을 위해 네트워크 패킷을 단순히 포트별로 분류하는 방법이 있다. 네트워크 패킷은 기기에 도착하면, 패킷에 있는 기기의 포트로 전달되고, 기기의 운영 체제는 해당 포트에 도착한 데이터를 프로세스에서 읽어가도록 한다. 반대의 과정이 전송할 때 일어난다. 악성 프로그램은 기기에서 프로세스로 동작할 때 악성 행위를 할 것이기 때문에, 네트워크 패킷을 포트별로 분류하는 방식은 타당해 보인다. 하지만, 프로세스가 fork하여 자식 프로세스를 만드는 경우 복합적인 행위를 해석하기 어려워진다.

본 연구에서는 이를 해결하기 위하여 네트워크 패킷을 두 가지 방식으로 분류하고, 둘 모두를 학습/추론에 이용하였다. 첫 번째 방법은 앞서 언급한 단순하게 포트로 분류하는 방식이다. 이 방법에 대한 타당성은 앞서 언급하였다. 두 번째 방법은 같은 프로세스 트리에 있는 프로세스에 연결된 포트로 들어오는/나가는 네트워크 패킷을 같이 분류하는 방법이다. 이 방법은 앞서 말한 첫 번째 분류 방법의 단점을 보완해 준다.

네트워크 패킷 분류가 끝나면 패킷 분석기에서 string으로 넘겨준 데이터를 표 1과 같이 구성된 특성을 가진 vector로 변환하여 준다. IP 정보는 기기의 IP가 없다면 기기에 들어오지 않을 것이기 때문에 포함하지 않았지만, 기기의 포트 정보는 두 번째 분류 방법에서 유의미한 정보가 되기 때문에 포함했다. 네트워크 패킷의 다른 정보들은 행위에 대한 정보가 없거나, 우회가 쉽거나, 혹은 특정 상황을 상정하는 정보가 되기 때문에 사용하지 않았다.

2.3 이상 행위 탐지기

본 연구의 NIDS의 이상 행위 탐지기의 동작을 정의하는 것은 결국 ANN의 구성이다. 본

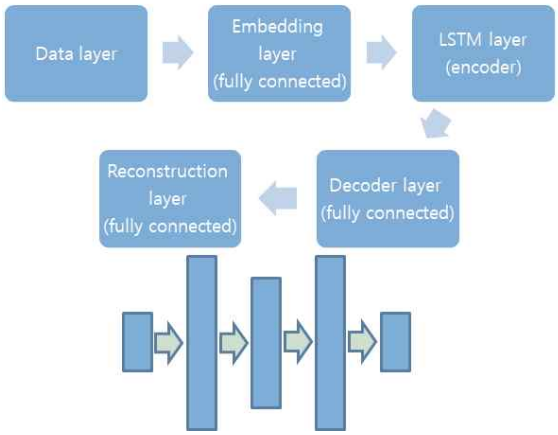


그림 1 이상 행위 탐지기의 ANN 구성

연구의 ANN은 그림 1과 같이 구성된다. 이 ANN은 다음과 같이 동작한다.

1) 데이터 전처리기에서 받은 feature vector, $\vec{v_i}$를 extender 층을 넣어 크기를 $\frac{1}{\beta} \cdot \dim(\vec{v_i})$ $where$ $\beta \in (0,1]$로 늘린 $\vec{v_e}$를 만든다.

2) $\vec{v_e}$를 LSTM 층에 넣어 LSTM의 hidden size로 줄인 $\vec{v_l}$로 만든다.

3) $\vec{v_l}$을 decoder 층에 넣어 $\vec{v_e}$를 재생성하는, 즉 $\vec{v_e}$와 크기가 같은 $\vec{v_e}'$을 만든다.

4) $\vec{v_e}'$을 decoder 층에 넣어 $\vec{v_i}$를 재생성하는, 즉 $\vec{v_i}$와 크기가 같은 $\vec{v_i}'$을 만든다.

이제 이 과정의 타당성에 관하여 논해보자. 우선 ANN의 입력 형태가 vector이고 이 값은 연속적이기 때문에, 기존의 자연어 처리(NLP)에서 사용하듯이, 그 갯수에 제한이 있는 target word를 두고 이를 예측하는 방식의 학습은 불가능하다. 따라서 학습을 위해 학습이 잘 되었는지 나타내는 새로운 기준이 필요하다. 이를 위하여 autoencoder를 사용하였고, autoencoder를 사용함으로써 1에서의 입력과 4에서의 출력의 mean squared error(MSE)를 새로운 기준으로 삼을 수 있었다.

이 autoencoder의 입력은 네트워크 패킷에서 추출한 feature vector, 즉, 시계열 데이터이기 때문에, encoder 단에는 LSTM이 포함되어야 한다. 반면, decoder 단에는 LSTM에 의해 시간 정보가 포함된 채 encoding된 vector가 입력으로 들어오며, feature vector의 재구성은 이전의 재구성과 무관하게 일어나기 때문에 LSTM을 포함시키지 않았다. 학습의 목표를 extender의 출력 vector와 decoder의 출력 vector의 MSE로 설정하면, extender가 constant한 vector를 출력하여 ANN의 입력과 상관없이 MSE를 줄이는 방향으로 학습될 수 있다. 따라서 학습의 목표는 ANN의 입력인 feature vector가 포함되어야 하며, 이에 따라 학습의 목표를 feature vector와 reconstructor의 출력 vector의 MSE로 설정하기 위하여 reconstructor 층도 필요하게 된다.

Autoencoder는 vector가 압축이 된 후에 다시 더 큰 vector로 투사하는 과정이 필요하다. 즉, 정보 손실을 의도한 뒤, 더 적은 정보로부터 원래 정보를 유추해내도록 학습을 시켜야 한다. 그런데, 앞 문단에서 밝혔듯이 encoder에는 LSTM이 포함되기 때문에, encoder의 출력 vector 크기는 LSTM의 hidden size로 정해지게 된다. 이 hidden size가 feature vector의 크기보다 크고 이를 그대로 LSTM의 입력으로 넣으면 feature vector가 압축되지 않기 때문에 encoder 본래의 목적을 달성하지 못한다. 따라서 LSTM 층 전에 extender 층을 두어, feature vector를 더 큰 vector에 투사한다.

이로써 1)-4)에 대한 타당성을 모두 논하였고, 이 ANN을 이상 행위 탐지기로 이용하였다.

2.4 학습과 추론

본 단에서는 2.3에서 제시한 모델을 학습 및 추론하는 과정에 대해 설명한다. 학습에서는 ANN에서 사용하는 optimizer의 한 종류인 Adam optimizer를 이용한다. Optimizer를 정하였으면 optimizer가 최소화할 값을 정하여야 하는데, 이는 2.3에서 논의했듯이 MSE를 이용하면 된다. 학습에 데이터를 사용하는 방법은 데이터 전처리기에서 나온 데이터를 stream 데이터로 넣지 않고, 2.2에서 설명한 분류법으로 분류가 완료된 시퀀스 데이터를 모델에 입력으로 넣는 방법이다. 모델은 입력을 받으면 출력 결과가 나올 것이고 이를 optimizer가 보고 모델을 학습시키게 된다.

추론 과정은 실제 네트워크 패킷을 실시간으

로 분석을 해야하기 때문에, 학습 시 데이터를 사용하는 방식과 다르게 이루어져야 한다. 모델에 전처리기에서 나온 데이터를 입력으로 넣어, feature vector와 재생성된 vector 사이의 MSE와 LSTM 층에서 나오는 context vector, 두 가지를 출력 값으로 내게 만든다. 이 출력 값은 데이터 전처리기에서 구분된 기준으로 저장을 해두고, 같은 종류로 구분된 데이터가 들어오면 저장해둔 context vector를 LSTM의 context 입력으로 넣어주어 모델을 동작하게 한다.

III. 결론

본 논문에서는 LSTM 기반의 autoencoder를 이용한 NIDS를 제시하였다. autoencoder에 LSTM을 포함시킴으로써, [4]의 문제점이었던 시계열 데이터를 자르는 윈도우를 사용하지 않고 비정상 탐지할 수 있다. 이는 본 연구에서 제시한 NIDS가 시간차 공격에 대하여 더 내성 있는 NIDS임을 의미한다.

IV. ACKNOWLEDGMENT

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원(No.2018-0-00230, (IoT 총괄/1세부) IoT 디바이스 자율 신뢰보장 기술 및 글로벌 표준기반 IoT 통합보안 오픈 플랫폼 기술개발 [TrusThingz 프로젝트])과 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구이며 (No.2016-0-00078, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발), 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2017R1A2A1A17069478)을 밝힙니다.

[참고문헌]

[1] Dimitrios Damopoulos, Sofia A Menesidou, Georgios Kambourakis, Maria Papadaki, Nathan Clarke, and Stefanos Gritzalis. Evaluation of anomaly-based ids for mobile devices using machine learning classifiers. Security and Communication Networks, 5(1):3 - 14, 2012.

[2] Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Systems with Applications, 39(1):424 - 430, 2012.

[3] Supranamaya Ranjan. Machine learning based botnet detection using real-time extracted traffic features, March 25 2014. US Patent 8,682,812.

[4] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. Network and Distributed Systems Security Symposium (NDSS) (2018).

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735 - 1780, 1997.

[6] Borja Merino. Instant Traffic Analysis with Tshark How-to. Packt Publishing Ltd, 2013

AI칩과 머신러닝 프레임워크, 암호 구현의 새로운 Tool: 격자 기반 암호를 중심으로

정치곤*, 전성일, 백윤흥**

*서울대학교 응용공학과, **서울대학교 전기정보공학부

AI Chip and Machine Learning Framework, a New Tool for Implementation of Cryptography: Focus on Lattice-based Cryptography

Chi-Gon Jung*, Seongil Jeon, Yunheung Paek**

*Department of Engineering Practice, Seoul National University.
**Department of Electrical and Computer Engineering, Seoul National University

요 약

지능화 서비스의 실시간 제공을 위해 Edge-Device에 인공지능 연산 가속기(AI칩)가 내장되고 있다. 인공지능 처리를 위한 데이터가 Device에 저장되면서 보안 이슈는 더욱 커졌지만, AI칩 사용을 위해 전력 및 메모리 등의 자원이 추가로 소모되면서 암호 활용에 할애 할 자원은 더욱 적어졌다. 이에 따라, 암호 설계 연구는 다양해지고 있지만, 암호 구현은 가속기 설계 또는 소프트웨어 최적화를 벗어나지 못하고 있어 변화를 수용하는데 제한적이다. 본 연구에서는 AI칩의 특징을 살펴보고 GPU를 활용하여 격자 기반 암호의 구현함으로써 AI칩이 암호 구현에 효율적으로 사용될 수 있음을 보인다.

I. 서론

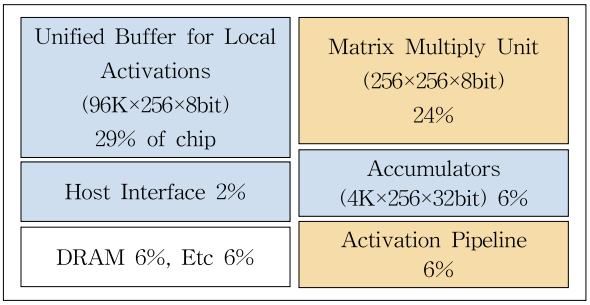
AI칩은 딥러닝 연산 가속을 위해 FPGA나 ASIC 또는 GPU에 기반한 별도의 칩(SoC) 등으로 제작된 프로세서를 의미한다[1]. AI칩은 주로 대규모 데이터센터에서 사용되어 왔으나, 최근에는 자율주행자동차, 드론 등 실시간 지능 서비스 수요가 증가하면서 Edge-Device용 AI 칩이 부각되고 있다. 데이터를 기기 내에서 처리하게 되면서 각 기기들의 보안 이슈도 중요해지고 있다[2]. 이런 변화는 암호 설계 및 실용화에도 많은 영향을 미친다. 자원(전력, 메모리 등) 제약된 Edge-Device에 AI칩까지 내장되면서 암호 활용에 사용할 수 있는 자원은 더욱 적어지기 때문이다. 한편, 급속도로 발전하고 있는 양자컴퓨터와 양자알고리즘은 표준 공개키 암호를 해독 할 수 있을 것으로 예상된다. 이에, 새로운 공개키 암호 표준화를 위한 NIST 주관 공모전이 진행되고 있다.

자원 제약적 환경에서의 활용 가능성과 양자컴퓨터에 대한 안전성을 만족하기 위해 다양한 암호들이 새롭게 설계되고 있다.

이처럼 암호 설계는 컴퓨팅 환경 변화에 따라 민첩하게 대응하고 있으나, 구현 연구는 범용 CPU의 병렬처리 명령어 set 활용, FPGA 등 가속기 이용 등으로 고착되는 경향을 보이고 있다.

이제 암호 구현도 변화를 맞이할 때가 되었다. Edge-Device에 강력한 연산기인 AI칩이 내장되기 시작했고, 더욱 늘어날 것이다[2]. AI칩의 핵심기능은 행렬 곱셈을 최적화하는 것으로, CPU 대비 15~30배 빠른데도 30~80배 적은 전력을 소모한다[3]. AI칩을 암호 구현에 활용하면 효율적 연산이 가능할 것으로 예상됨에도 현재까지 관련 연구는 이루어지지 않았다.

본 연구에서는 AI칩과 머신러닝 라이브러리의 주요 특징을 살펴보고, GPU와 Google의 Tensorflow를 활용한 격자 기반 암호 구현 결과를 제시함으로써, AI칩이 암호 구현에 효율적으로 사용될 수 있음을 보인다.



[그림 1] TPU 칩의 면적 분포[3]

II. 배경 지식

2.1 AI칩의 구조와 개발 프레임워크

AI칩은 딥 러닝의 학습(training) 및 추론(inferencing) 연산을 가속하는 프로세서로 FPGA, ASIC, GPU 기반 SoC 등 다양한 형태를 갖는다.

구글의 AI칩인 TPU(Tensor Processing Unit)은 ASIC 형태로, 2015년부터 데이터센터에서 사용되고 있다[3]. TPU는 [그림 1]과 같이 약 6만 5천개의 곱셈기(Matrix Multiply Unit)와 내부 저장공간(Unified Buffer) 24MB을 포함해 칩 면적의 약 67%가 연산과 관계된 기능들이다. 곱셈기는 Systolic 모델로 실행되어 병렬 처리되며, 내부 저장공간은 연산할 데이터 또는 중간 연산결과를 저장함으로써 외부 메모리 interface에서의 병목을 최소화하여 효율성을 높였다.

이처럼 AI칩은 단순히 보면 엄청나게 많은 곱셈기와 Buffer를 갖은 연산 가속기로 그동안의 공개키 암호 하드웨어 구현에서 고민했었던 곱셈기 구현과 연계성이 있다고 볼 수 있다.

Edge-Device용 AI칩은 Apple(A11), 퀄컴(snapdragon845), 화웨이(Kirin 970) 등 글로벌 기업들이 모바일용 SoC로 출시하기 시작했다. 구글의 Edge-TPU는 Cloud 및 Tensorflow 프레임워크와 연계된 IoT용 ASIC으로 올해 10월부터 일부에게 Early access를 허용하고 있다. Edge-TPU는 부동소수점 대신 8bit・16bit 정수 처리에 적합하게 설계되어 딥러닝 추론 단계를 최적화한다. 암호 연산은 대부분 정수에 기반하기 때문에 추론 단계에 최적화된 칩 사용은 더욱 효과적일 것으로 예상된다.

구글 Tensorflow는 Python과 C++로 사용할 수 있는 프레임워크로 딥러닝 연산의 대부분을 차지하는 행렬 연산에 필요한 다양한 함수를 제공한다. 예를 들어, 행렬의 내적을 계산하는 *matmul()*은 매개변수 행렬의 전치 여부, sparse 여부 등을 True, False로 입력 받도록 만들어져 있어 코드 간결화와 행렬 고속 연산을 지원한다. 다만, 암호 구현에서 많이 사용하는 Loop명령어 중 *for*문은 지원하지 않고 *while*문만 사용할 수 있다. 특히, Python 구현의 경우 Loop명령어의 효율이 C++에 비해 나쁘기 때문에, unrolling 등 다른 방법이 필요하다.

2.2 격자(Lattice) 기반 암호

자원 제약적 환경에서의 활용 가능성과 양자 컴퓨터에 대한 안전성 측면에서 다양한 암호들이 경쟁하고 있는데, 격자 기반 암호알고리즘은 두 가지 성질을 두루 겸비하여 각광 받고 있다.

격자 기반 암호는 Worst-case Hard Problem인 Shortest Vector Problem으로부터 reduction 되는 Learning With Error (LWE) 문제를 주로 사용한다. LWE 문제에 기반한 암호 알고리즘의 주요 구성은 행렬/벡터 간 곱셈과 에러 샘플링으로, 이를 효율적으로 수행하기 위한 다양한 변형 스킴들이 연구되었다.

Ring-LWE[4]는 행렬/벡터를 다항식으로 변형하여 곱셈 등 연산을 효율화 하였고, LWR (Learning With Rounding)[5]은 에러 샘플링 대신 반올림을 통해 에러의 효과를 주었다.

2.2.1. LIZARD[6]

LIZARD는 국내에서 개발한 LWE 및 LWR 기반 암호로, NIST의 Post Quantum Cryptography 표준화 공모전에서 60여 개 암호들과 경쟁 중이다. LIZARD는 암호화 과정에서 LWR을 사용하여 에러 샘플링을 하지 않고, Secret vector를 Sparse한 Binary로 구성하여 효율성을 높였다. 또한, Ring-LWE 스킴으로도 변형할 수 있어 자원이 제약된 Device에도 사용할 수 있다.

III. 암호 구현방법의 제한사항

3.1 소프트웨어 구현

소프트웨어 구현은 Intel, AMD, ARM 등과 같은 범용CPU의 병렬 처리를 활용한 AVX (Advanced Vector eXtensions) 명령어 셋 등을

활용하는 방법이 대부분이다. 격자 기반 암호도 AVX를 활용한 구현기법들이 많이 연구되었다. Ring-LWE 스킴은 8bit AVR에서 구현될 수준으로 최적화 되었지만, LIZARD와 같은 LWE 스킴에 대한 연구결과는 알려지지 않았다.

하지만, AVX 등 CPU의 병렬화 명령어 셋은 플랫폼에 의존적이어서 SoC나 ASIC 형태의 AI 칩이 내장된 Edge-Device에서의 활용 가능성은 미지수이며, 프로그래밍 기술과 컴퓨터 구조에 대한 상당한 수준의 지식을 필요로 하여 암호 사용의 대중화 측면에서도 부정적이다.

3.2 하드웨어 구현

하드웨어 구현은 대부분 FPGA나 ASIC 등 가속기를 설계하는 연구로, 소모전력이나 지연시간 최소화를 목표로 한다. FPGA는 프로그래밍 가능하다는 장점이 있어 하드웨어가 갖는 민첩성의 제한을 극복할 수 있다고 여겨지고 있다.

하지만, 클라우드 같은 가상화 환경에서는 원격으로 기기내 소프트웨어를 수정하기 원하고, FPGA가 갖는 민첩성은 의미가 퇴색된다[7]. 또한, 이미 AI칩을 내장함으로써 자원 소모가 상당한 상태에서 하드웨어를 추가하는 것은 비용과 전력 소모 측면에서 매우 어렵다.

IV. GPU와 Tensorflow를 활용한 구현

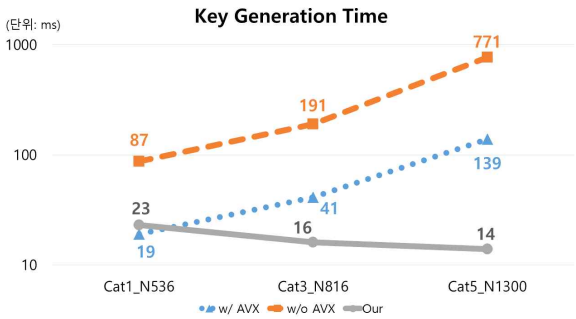
본 절에서는 AI칩이 격자 기반 암호 구현에 효율적인지 평가하기 위해, GPU와 Tensorflow를 사용하여 LIZARD -CPA-PKE 스킴의 암호키 생성 알고리즘을 구현하였다.

Edge-Device용 AI칩은 ASIC 또는 SoC형태이나, 개발 Kit 제공이 제한적임에 따라 PC용 GPU를 사용하였다. 일반적으로 ASIC은 GPU보다 우수한 성능을 갖으며, 구글의 Edge-TPU 등은 GPU와 달리 정수 연산을 채택하고 있기 때문에 GPU에서 좋은 성능을 보인다면, AI칩에서는 더욱 효율적일 것으로 판단할 수 있다.

우리의 구현은 https://github.com/wjdclrhs/LIZARD_GPU_TENSOR 에서 확인할 수 있다.

4.1 구현 환경

구현 환경은 Windows10(64bit)에서 i5-8500 (3.00GHz) CPU, Geforce GT 1030(2GB) GPU,



[그림 2] GPU - Tensorflow(Python) 구현결과와 CPU - C 구현결과 비교

Cuda9.2, CuDNN 7.1.4, Jupyter notebook Python 3.6.5, Tensorflow 1.10.0 이다.

성능 비교를 위해 [6]의 오픈소스 C 코드를 동일 환경에서 GCC 6.3.0로 실행하였다. 컴파일 옵션은 [6]과 같은 AVX 사용 환경(*-O3 -fomit-frame -pointer -msse2avx -mavx2 -march=native -std=c99*)과 미사용 환경(*-O3 -fomit -frame -pointer -std=c99*)으로 구분하였다.

파라메타는 NIST post quantum cryptography 공모전 참조 문서[8]에 포함된 category1(N536)과 category3(N816), category5(N1300)을 사용하였다. category가 1에서 5로 올라갈수록 안전성은 높아지지만 dimension도 커져 matrix 곱셈 등 연산량이 많아진다.

4.2 구현 결과 및 평가

Cat1에서는 AVX를 적용한 C 코드와 유사한 성능을 보이지만, Cat3에서는 약 2.5배, Cat5에서는 약 10배 빠르다. 특히, CPU가 AVX를 지원하지 않는 상황을 가정하면 효율이 매우 떨어지므로, AI칩을 사용하는 것이 효과적이다.

특히, 보안강도가 훨씬 높은 Cat5에서 우리의 구현이 Cat1의 C 코드보다 우수한 것으로 나타난 바, 더 높은 보안수준을 더 빠르게 사용할 수 있는 이상적인 결과가 도출되었다.

Cat1에서 Cat5로 파라메타가 커질수록 [6]의 실행 결과들은 시간이 증가한 반면, 우리의 구현은 오히려 속도가 빨라진다. 이 결과는 Tensorflow와 Python의 loop문 처리 특성에서 비롯된다. 우리의 구현에서 소비 시간의 대부분은 에러 샘플링 단계로, Cat1 22ms, Cat3 15ms, Cat5

13ms 이다. [6]은 bit 연산으로 에러를 구하는 함수를 만들고, 에러 함수를 *for*문으로 looping 하면서 call하여 matrix를 생성한다. 하지만, Python은 loop문과 함수 콜을 인터프리터 하는 오버헤드가 크고, Tensorflow에서는 *for*문을 지원하지 않기 때문에, 우리는 행렬의 원소들을 각각 비교하는 방법을 사용하였다. 우리의 구현은 샘플링 분포의 표준편차가 클수록 비교 단계가 많아지는 특징이 있다. Dimension이 큰 파라메타(Cat5) 일수록 표준편차가 작은 분포를 갖으므로 빠른 속도가 나온 것으로 분석할 수 있다.

실제로, 키 생성시간 중 에러 샘플링을 제외한 matrix간 곱셈은 matrix 크기가 커지는 것에 영향 없이 약 1ms 정도만 소요되는 것을 알 수 있다. 그러므로, 샘플링을 효율적으로 수행하는 다른 구현 방법이나 알고리즘을 연구한다면 키 생성시간은 비약적으로 줄어들 것으로 예상된다.

V. 결론

본 연구에서는 Edge-Device의 실시간 지능화 서비스를 위해 개발되고 있는 AI칩을 격자 암호 구현에 활용시 매우 효율적임을 보였다. 그 간의 암호 구현은 별도의 비용을 수반하는 제한사항이 있었으나, AI칩에 내장된 곱셈 연산능력과 라이브러리를 활용하면 많은 암호 알고리즘들의 연산 가속 효과를 누릴 수 있을 것이다.

향후에는 이번 연구에서 병목 구간으로 밝혀진 에러 샘플링에 대한 효율화 방안에 대한 연구와 ECC, AES와 같은 표준 암호들의 가속 성능을 실험해볼 필요가 있으며, ASIC형태의 AI칩에 실제 검증해보고, 부채널 공격 이슈에 대해서 검토해보는 연구도 필요할 것이다.

ACKNOWLEDGEMENT

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임. (No.2016-0-00078, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안기술 개발)

[참고문헌]

[1] 최새솔, 인공지능 반도체 산업동향 및 이슈 분석, ETRI Insight Report 2017-37, December, 2017.

[2] ROMAN, Rodrigo; LOPEZ, Javier; MAMBO, Masahiro. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, 2018, 78: 680-698.

[3] JOUPPI, Norman P., et al. In-datacenter performance analysis of a tensor processing unit. In: Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on. IEEE, 2017. p. 1-12.

[4] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," EUROCRYPT 2010, Springer, pp. 1-23, Monaco, Monaco, May 2010.

[5] BANERJEE, Abhishek; PEIKERT, Chris; ROSEN, Alon. Pseudorandom functions and lattices. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2012. p. 719-737.

[6] Cheon JH, Kim D, Lee J, Song YS. Lizard: Cut off the Tail!//Practical Post-Quantum Public-Key Encryption from LWE and LWR. IACR Cryptology ePrint Archive. 2016;2016:1126.

[7] Nejatollahi H, Dutt N, Cammarota R, editors. Trends, challenges and needs for lattice-based cryptography implementations: special session. Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion; 2017: ACM.

[8] https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Lizard.zip

SDR을 활용한 실용적인 LTE 재밍 연구

양호준, 손민철, 노상미, 김용대
KAIST 전기 및 전자공학부

Practical LTE jamming with SDR

Hojoon Yang, Mincheol Son, Sangmi Noh, and Yongdae Kim
Department of Electrical and Engineering, KAIST

Abstract

LTE standard is the most widely used wireless network these days. LTE adopts Orthogonal Frequency Division Multiplexing (OFDM) to modulate the signal. While the OFDM is robust to multipath fading effects, it is not adequate to the jamming attack. In order to reduce power consumption of the jammer, many power-efficient jamming techniques have been proposed, but these techniques have limitation to perform the attack practically. In this paper, we implemented synchronized jammer against commercial LTE base station using USRP board to show the feasibility of the synchronized jamming attack. Our implementation also deals with clock synchronization issue and resolves it by using GPS disciplined oscillator (GPSDO). To the best of our knowledge, this issue was never taken into consideration in previous studies. Finally, we verified our synchronized jamming attack.[1)]

I. Introduction

Wireless mobile network adopts Long Term Evolution (LTE) standards which use OFDM as a modulation technique. Although the OFDM is robust to multipath fading effects, it is not adequate to the environment where an intentional jamming signal exists. LTE systems, which also use OFDM modulation, are vulnerable to intentional jamming signals. Since information about carrier frequency and bandwidth of every LTE systems is open to the public, the malicious node can send the jamming signal on the frequency bands of an LTE system. Jamming is an intentional radio signal that blocks the communication between legitimate transmitter and receiver. Attacker sends a noise-like signal so that signal to noise ratio (SNR) at the legitimate receiver becomes too low to correctly decode the message. Generally, jamming signal lasts for a long time, overlays all time and spectrum, which is called barrage jamming. For the attacker, however, it is not efficient to jam the whole spectrum. Thus, the attacker can selectively jam the spectrum to reduce the power consumed while maintaining the effectiveness of the jamming attack. Many kinds of power-efficient jamming attacks have been studied and implemented in hardware [1, 2]. However, both [1] and [2] have limitation to perform the attack practically. In order to perform the synchronized jamming attack, the attacker needs to synchronize with the time and frequency of the target base station. However, Krenz et al [1] didn't consider the time and frequency synchronization with the target base station and Rao et al [2]

1) This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00831, A Study on Physical Layer Security for Heterogeneous Wireless Network)

considered time synchronization, but not frequency synchronization. Frequency synchronization is also important since it will be drifting as time goes by.

In this paper, we implement the practical synchronized LTE jammer on the SDR platform and perform the jamming attack on the commercial base station within the Faraday cage. We implement the frequency synchronization process using GPSDO to avoid frequency drifting. The jammer automatically can find the target base station and perform the time and frequency synchronization.

II. Implementation

In this section, we describe how we implement the synchronized jammer in detail. There are two main categories of implementing synchronized jammer, Synchronization part and Jamming part.

2.1 Synchronizing with Target base station

The synchronized jammer needs to synchronize with the target base station which is connected to the target user equipment (UE) to jam the target UE. We assumed that the attacker already knows the target base station connected to the target UE. This assumption is not quite tough to the attacker. Because there are several achievable methods to identify the target base station. If the attacker knows target UE's the phone number, the attacker calls the target UE periodically to invoke the periodic call traffic. The attacker observes each radio network temporary identifier's (RNTI) traffic variance in the cell by using a cell measurement tool such as Airscope [3] to identify the target UE. Another possible method is to observe the debugging information which is supported by all LTE devices. If the attacker knows what LTE service provider the target UE is using, the attacker may want to observe the debugging information of UE serviced by the same operator as the target UE. Because the two UEs which are located close to each other are highly likely to be connected to the same cell. The attacker now obtains the target base station information and have to figure the target base station's transmission timing out. The timing can be obtained by observing a periodically transmitted primary synchronization signal (PSS), secondary synchronization signal (SSS) sequence from the target base station. However, the timing obtained by the attacker indicates when the signal transmitted by the target base station reaches the attacker. Thus, in order to obtain the exact timing that reaches the target UE at the same time as target base station, the attacker needs to add or subtract some propagation delays to the obtained timing depending on the distance among the attacker, target base station, and target UEs. But, from the jamming point of view, the propagation delay is negligible. Because, even if the distance between the jammer and target UE is 1 km away, the propagation delay is only 3.4 microseconds, which is negligible compared to 10 ms which is an LTE frame time.

2.2 Jamming

We implemented the synchronized jamming program based on the srsLTE [4] to run it via SDR platforms such as USRP or BladeRF. Especially, we modified the pdsch_enodeb.c file which runs a simple downlink only eNodeB. In order to generate Gaussian the jamming symbols, we used rand()%256. We implemented five jamming modes, which are a barrage jammer, channel reference signal jammer, physical broadcast channel (PBCH) jammer, physical downlink control channel (PDCCH) jammer, and PDSCH jammer. The maximum bandwidth of the jammer is 20MHz and it depends on the target base station bandwidth. For a barrage jammer, we allocated the jamming symbols into all LTE symbols in an LTE frame. Similarly, for other jammers, we allocated the jamming symbols into each corresponding symbols in an LTE frame. Our synchronized jammer takes input for which jamming mode the attacker would like to execute among five jamming modes. There are two main phases in our jammer, receiving phase and transmitting phase. In the receiving phase, the jammer takes the PSS and SSS sequence from the target base station to synchronize

with the target base station and it stores the received time (t0) after successful synchronization. In the transmitting phase, the jammer transmits the jamming signal at the next LTE frame transmission timing (i.e. t0 + 0.01).

III. Implementation Issue

While we developed the jammer, we encountered an implementation issue we have never considered. To the best of our knowledge, this issue never has been addressed in previous studies.

3.1 Clock synchronization

All RF radio device contains at least one oscillator to generate the frequency. Even oscillators operating on the same clock have a slight difference. However, its small variance becomes large variance since it will be added up as time goes. Thus, even if two devices have same clock frequency and start the clock at the same time, they will drift apart from each other as time goes. This clock drifting phenomenon applies equally to the jammer as well. For successful synchronized jamming, it is most important to keep synchronize between the jammer and the target base station. In many synchronized applications, the clock synchronization between two devices can be achieved by sharing one clock source. However, this technique cannot apply to this case since the jammer cannot share the clock source with the target base station. Another clock synchronization technique is to use GPS disciplined oscillator (GPSDO). The GPSDO outputs the very accurate 10MHz reference clock which is continuously trained by the GPS signal for clock synchronization. Thus, two devices can achieve clock synchronization if they equipped GPSDO. The target base station is known to use reference clock disciplined by GPS signal [5].

IV. Experimental Setup

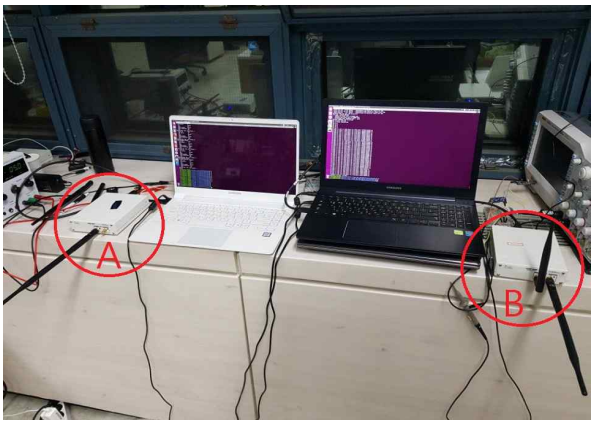


Figure 1 Experimental setup. (A) indicates the target UE and (B) indicates the jammer.

Fig. 1 shows the overall experimental environment. The target UE is pdsch_ue.c program supported by the srsLTE [4]. The target UE keep decoding the PDSCH data against C-RNTI 0xffff which means system information block, especially SIB1. It also shows center frequency offset (CFO), PDCCH missing rate and PDSCH block error rate (BLER) against the connected base station and constellation plot for PDCCH, PDSCH, and PSS, SSS. Thus, we can easily observe which channel is affected by the jammer and how much the jammer affects it. In this paper, we didn't use commercial LTE UE but pdsch_ue.c program because the commercial UE does not show the detailed PHY information including PDCCH missing rate or BLER which are needed to measure the jamming performance. For the target base station, we used commercial LTE eNodeB whose band is 7, cell id is 439, earfcn is 3200 and its bandwidth is 10MHz. For the Jammer, we used our synchronized jammer. We assumed that the target UE and jammer have fixed position. We placed the jammer about 1 meter away from the target UE. We cannot increase the distance between them anymore because their power is too low.

V. Verification

Unlike the traditional jammer which jams whole spectrum and time domain, synchronized jammer have to jam the specific LTE channel such as PDCCH, RS, PCFICH,

PDCCH, and so on. To verify that our synchronized jammer jams specific LTE channel, we use the pdsch_ue program provided by srsLTE, which is a dummy UE that shows channel response, PSS/SSS cross-correlation value, and PDCCH/PDSCH equalized symbols. For verification, our jammer jam the RS channel only. If the jammer works correctly, the channel response graph on the UE side will significantly vary and other channel states should be stable as before. Fig. 2 shows the channel response graph and PSS/SSS cross-correlation graph of the UE side before and after jamming. Before jamming, the PDCCH-miss rate of the UE is 0.23% and PSS cross-correlation value reaches to 1 (i.e. the UE can decode all data and PSS finding is good, see Figure 2.A). After jamming, however, the PDCCH-miss rate of the UE is 100%, but PSS cross-correlation value still reaches to 1 (i.e. the UE cannot decode any data at all, but PSS finding is good, see Fig 2.B). It means that our RS only jamming works correct because our jammer jams the UE successfully while it does not interfere with the PSS/SSS channel which is not our target channel.

VI. Conclusion

In this paper, we investigated synchronized jamming on the LTE system. We implemented synchronized jammer against commercial LTE base station using USRP board to show the feasibility of the synchronized jamming attack. Our implementation deals with clock synchronization issue which was never mentioned in the previous works and resolves it by using GPSDO. Furthermore, unlike the previous studies, which skipped the synchronization procedure and physically connected the jammer and the transmitter using RF cable, we transmitted the jamming signal through antennas within the Faraday cage. Implementation with the USRP board showed the feasibility of our attack on the commercial base station and it was verified by RS jamming experiment.

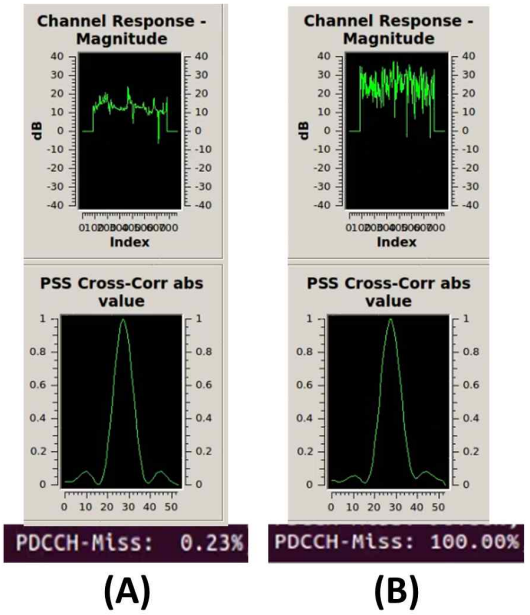


Figure 2 Channel Response, PSS cross-cor graph and PDCCH Miss rate of UE. (A) is before jamming and (B) is after jamming.

[References]

[1] R. Krenz, and S. Brahma. Jamming LTE signals. IEEE International Black Sea Conference on Communications and Networking, pp. 72-76, May, 2015.

[2] R.M. Rao, S. Ha, V. Marojevic, and J.H. Reed. LTE PHY Layer Vulnerability Analysis and Testing Using Open-Source SDR Tools, IEEE MILCOM 2017, 23-25 Oct. 2017.

[3] Airscope, http://www.softwareradiosystems.com/tag/airscope/

[4] srsLTE, http://github.com/srsLTE/srsLTE

[5] http://www.furuno.com/en/gnss/case/furuno02

LTE 제어평면 메시지 암호화에 대한 고찰+

손민철[1], 배상욱[1], 김홍일[1], 이은규[1], 양호준[1], 이지호[2], 심규찬[2], 손수엘[2], 김용대[1]

KAIST [1]전자전기공학부, [2]정보보호대학원,

A Study on LTE Control Plane Message Encryption

Mincheol Son[1], Sangwook Bae[1], Hongil Kim[1], Eunkyu Lee[1], Hojoon Yang[1], Jiho Lee[2], Kyuchan Shim[2], Sooel Son[2], Yongdae Kim[1]

[1]School of Electrical Engineering, [2]Graduate School of Information Security, KAIST.

요 약

이동통신망에서는 악의적인 공격자로부터 사용자 및 망의 고유 기술/정보를 보호하기 위하여 민감한 내용을 담고 있는 메시지들을 암호화하여 사용자 단말과 서로 주고받는다. 본 논문에서는 국내외 5개 이동통신사에서 실제로 주고받는 이동통신 사용자의 이동성 관리 및 통신망 접속 관리에 사용되는 제어평면 메시지들을 수집한 뒤, 각 통신사별로 메시지들의 암호화 여부를 조사하였으며, 암호화가 되지 않은 메시지에서 유출될 수 있는 정보들에 대해 분석하였다. 또한, 메시지가 암호화되지 않은 원인을 분석하고 일부 이동통신 사업자 및 표준의 문제를 서술한다.

I. 서론

이동통신 시스템에서는 공기 중으로 데이터를 전송하기 때문에 악의적인 공격자에게 사용자와 이동통신망 사이에 서로 주고받는 데이터가 노출되기 쉽다. 이러한 노출되는 정보는 사용자의 개인정보 및 데이터뿐만 아니라 망의 구성정보와 같은 고유 정보들을 포함 한다. 따라서 사용자와 이동통신망 사이에 주고받는 메시지에 대한 암호화 및 무결성 검사는 사용자뿐만 아니라 이동통신사에게도 매우 중요하다. 이를 위해, 3G와 4G LTE 네트워크의 규격을 담당하는 3rd Generation Partnership Project (3GPP)에서는 메시지들의 암호화 및 무결성을 보호하기 위한 과정과 사용 가능한 알고리즘에 대하여 정의하고 있다.

본 논문에서는 미국 및 한국의 주요 5개 통신사들에서 실제 사용자와 주고받는 제어평면 메시지 중 하나인 Non Access Stratum (NAS) 메시지들에 대한 암호화 적용 여부를 점검하였다. 또한, 암호화되지 않은 제어평면 메시지들에서 어떠한 정보들이 노출되고 있는지에 대해 살펴보고자 한다.

본 논문에서는 세 가지 단계로 분석을 진행한다. 먼저, 각 통신사별로 실제 상용 단말과 주고받는 제어평면 메시지들을 수집한다. 다음으로, 실제 전송될 때 암호화가 되지 않은 메시지를 구별하고 메시지 안에 담겨있는 정보들을 수집한다. 마지막으로, 암호화가 되지 않아 공격자에게 노출될 수 있는 메시지의 내용 중 노출되지 않아야 하는 데이터가 있는지 확인한다. 이 과정에서, 많은 양의 정보를 일일이 검사하는 방법을 벗어나, 통신사 간의 비교 분석을 통해 분석 범위를 효율적으로 줄이고, 메시지가 암호화되지 않는 원인을 조사하였다.

본 논문에서는 실제 데이터의 분석을 통하여 통신사마다 NAS 메시지들의 암호화 여부는 다르게 나타나며 암호화되지 않는 메시지 중 공격자에게 노출되어서는 안 되는 정보를 포함하는 메시지가 있음을 보여 준다. 이러한 문제를 방지하기 위해서는 이동통신 표준에서 제시하는 내용에 대해 이동통신망 사업자의 올바른 이해와 선택이 필요하며, 표준 또한 명확한 정의가 필요하다.

+ 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2015-0-00403)

II.LTE 제어평면 메시지 암호화 과정

이동통신망은 사용자의 이동성 및 망 접속 관리를 위해 망 내의 장비인 MME+와 단말 사이의 제어평면 프로토콜인 NAS 프로토콜을 사용한다. 사용자는 망에 접속할 때 망과 암호키 및 암호화 알고리즘을 공유하는 동작을 수행하며, 초기 접속의 자세한 과정은 그림 1과 같다.

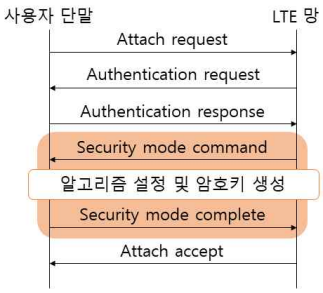


그림 1 LTE 초기 접속 절차

사용자로부터 접속 요구를 받은 LTE 망 내의 MME는 상호 인증과정 및 암호키 생성 과정을 진행한다. 사용자는 *Attach request* 메시지를 보낼 때, NAS 암호화 과정에서 단말이 사용할 수 있는 알고리즘에 대한 정보를 LTE 망에 보낸다. 망은 알고리즘 중 하나를 선택하여 *Security mode command* 메시지를 통해 사용자에게 알려준다. 단말은 LTE 망으로부터 받은 알고리즘을 사용해 암호키를 생성하며, 이후 과정부터 NAS 메시지는 암호화되어 전송된다. 이때, LTE NAS 암호화를 위해 사용되는 암호화 알고리즘의 종류에는 세 가지가 있다. 알고리즘은 EPS++ Encryption Key 0 (EEA0), EEA1 그리고 EEA2 라고 명명하며, 각각 NULL ciphering algorithm, SNOW 3G, AES 알고리즘을 나타낸다. 중요한 점은, 망이 암호화과정에서 사용될 알고리즘을 결정한다는 것이다.

III.통신사별 메시지 암호화 여부 분석

3.1 메시지 수집 및 암호화 여부 판단

실제로 통신사의 망과 단말 사이에서 주고받는 제어평면 메시지들의 암호화 여부를 알아보기 위해, 한국 및 미국의 5개 통신사를 대상으로 총 39만개의 제어평면 메시지를 수집하였다.

통신사	A	B	C	D	E
메시지 수 (천)	20	44	39	97	193

표 1 통신사별 수집된 메시지 수

다음으로, 각 통신사별로 수집된 NAS 메시지를 대상으로 각 암호화 여부를 분석하는 작업을 수행하였다. 암호화 여부 판단은 단순히 메시지의 암호화 여부를 알려주는 헤더 정보를 확인하는 것이 아닌, 실제 Man in the Middle (MitM)환경에서 메시지 내용의 노출 여부를 기반으로 판단하였다. 암호화 이전의 NAS 메시지의 값과 암호화 이후 실제 사용자가 기지국에 전송하는 데이터의 값을 비교함으로써 NAS 메시지 별로 암호화 여부를 판단하였다.

분석 결과, 이동통신망을 사용하기 위한 *Attach request* 메시지부터 사용자의 고유 정보를 전송하는 *Identity response* 메시지까지 다양한 메시지들이 암호화되지 않은 경우가 있음을 확인할 수 있었다. 또한, 통신사 별로 암호화되지 않은 메시지들이 다르게 나타났으며, 적게는 6개의 NAS 메시지, 많게는 36개의 메시지가 암호화되지 않은 채 전송되고 있었다.

3.2 통신사 간 비교를 통한 필터링

전 과정에서 식별한 암호화 되지 않은 메시지들과 노출되는 정보들을 모두 분석하는 것은 많은 노력을 요구할 뿐만 아니라, 불필요한 작업이다. 예를 들어, 표준에서 정의한 바에 따라 암호화가 불필요하거나 암호화를 위한 암호키가 생성되지 않았을 때는 메시지가 암호화되지 않는 것이 당연하며 분석과정에서 배제하는 것이 효율적이다. 하지만 방대한 표준을 분석하여 모든 메시지, 메시지의 내용에 대하여 분석 여부를 판단하는 작업은 여전히 많은 노력을 요구한다. 본 연구에서는, 이처럼 분석이 불필요한 메시지를 검출 및 배제하는 과정에 있어, 표준 분석 대신 비교분석 방법을 적용하였다.

비교 분석과정에서, 모든 통신사에서 공통으로 암호화하거나 암호화하지 않는 메시지들은 모두 알맞게 구현해 놓았다고 가정하였다. 따라서 5개 통신사를 비교하여, 모든 통신사에서 암호화하지 않고 있는 5개의 NAS 메시지들은 분석대상에서 제외하였다. 제외된 메시지에는 앞

+ Mobility Management Entity
++ Evolved Packet System

서 언급한 *Attach request*를 비롯하여 *Tracking area update request* 및 *Security mode command* 등의 메시지들이 포함된다. 해당 메시지들은 표준에서 암호화하지 않도록 명시되어 있다. 또한, 각 메시지의 내용에 대해서도 똑같은 방법을 적용하여, 분석 범위를 크게 줄일 수 있었다.

IV. 암호화되지 않은 메시지 분석

어떠한 정보들이 공격자에게 노출될 수 있는지 알아보기 위해서 암호화 되지 않은 NAS 메시지의 내용을 확인해 보았다. 노출되는 메시지의 내용에는 다양한 정보들이 있으며, 그 중 환경설정이나 타이머의 값, 옵션의 적용여부 등 노출되어도 큰 문제가 되지 않을 내용들 또한 포함한다. 본 연구에서는 그 중 사용자의 고유 정보나 데이터에 관한 내용을 위주로 분석하였다. 그 결과, 노출되는 메시지의 내용 중 사용자 단말의 고유 정보인 IMEI+와 SMS 내용이 존재함을 찾을 수 있었다.

4.1 IMEI

IMEI는 이동통신 사용자의 단말 고유 번호이다. LTE 보안을 정의하는 표준에서는 IMEI는 기밀성이 보장되어야 하는 정보로 정의하고 있다[1]. 하지만 분석대상 중 두 통신사에서는 IMEI가 포함된 NAS 메시지인 *Identity response* 메시지를 암호화하지 않은 채 통신망에 전송하고 있다. 그림 2를 보면, 단말이 망에게 전송하는 메시지인 *ULInformationTransfer* 메시지 안에 포함된 NAS 메시지의 내용이 평문 NAS 메시지의 내용과 같은 것을 볼 수 있다. 이 메시지의 내용은 IMEISV이며, IMEISV의 앞 14개의 숫자는 단말의 IMEI 값을 나타낸다. 이처럼 *Identity response* 메시지가 암호화되지 않을 경우, MitM 공격을 하는 공격자에게 사용자의 IMEI가 노출되게 된다. 이러한 IMEI 노출은 사용자를 특정가능하게 하여 이후에 위치 추적[4, 5]과 같은 또 다른 공격들의 기회를 열어준다.

+ International Mobile Equipment Identity

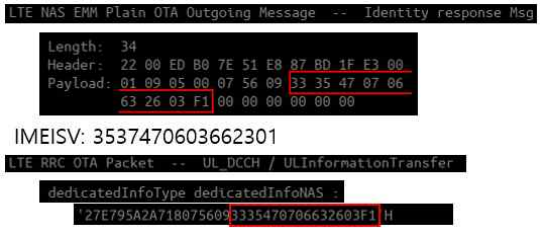


그림 2 통신사 C의 Identity request 메시지

4.2 SMS 데이터

NAS를 통해 SMS 문자 서비스를 사용하는 경우, 사용자끼리 주고받는 문자의 내용은 *Uplink / Downlink NAS transport* 메시지 안에 담겨 전달된다++. 이러한 데이터는 직접적인 사용자의 대화 내용이기 때문에 공격자에게 노출되어서는 안 된다. 하지만 분석대상 중 하나의 통신사에서는 SMS 내용이 포함된 NAS 메시지를 암호화하지 않은 채 통신하고 있음을 발견할 수 있었다. 이 경우, MitM 공격자는 사용자와 이동통신망이 주고받는 SMS 메시지들을 볼 수 있다는 것을 의미한다.

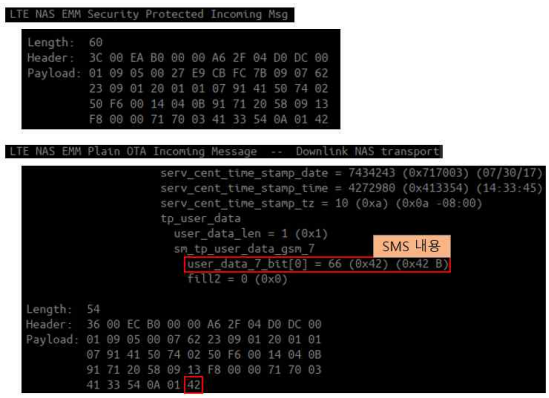


그림 3 통신사 E의 Downlink NAS transport

V. 취약점 원인 분석 및 대응방안

5.1 취약점 원인 분석

LTE NAS 메시지의 암호화를 위해서 사용자 단말은 자신이 사용 가능한 암호화 알고리즘을 통신망에 알려준다. 이 정보를 받은 통신망은 사용자와의 통신에 사용할 알고리즘을 선택한 후 사용자에게 알려준다. 그 뒤로 암호키를 재생성하는 과정이 실행되기 전까지 사용자

++ 다른 방법으로, IMS를 통하여 전송되는 SMS over IMS가 있으며, 이 경우 NAS를 통하여 전송되지 않는다.

와 통신망은 위 과정에서 결정된 암호화 알고리즘을 사용하여 서로 통신한다.

이러한 암호화 과정과 관련된 메시지를 살펴본 결과, 암호화되지 않는 메시지를 사용하는 통신사에서는 암호화 알고리즘으로 EEA0 를 사용하는 것을 찾을 수 있었다. EEA0 (NULL ciphering algorithm)는 암호화 알고리즘을 사용하지 않는 설정값으로, NAS 메시지를 암호화하지 않고 전송한다. 이동통신 표준에서는 NAS 메시지에 대한 암호화는 이동통신망 운영자의 옵션이라고 명시하고 있다[2]. 하지만 사용자가 가능한 암호화 알고리즘이 EEA0만 있는 것이 아니라 EEA1과 EEA2도 있음에도 통신망에서 EEA0를 선택해서 사용하는 것은 사용자의 정보가 유출될 수 있는 상황을 만들게 된다.

해당 취약점의 가능성은 기존 연구[6, 7]에서도 언급이 되었으나, 본 연구에서는 1) 실제 운영 중인 통신사에서 문제가 일어나고 있음을 확인 하였으며, 2) 단말의 설정문제가 아닌, 망에서의 잘못된 설정으로 일어나는 것을 확인할 수 있었다.

5.2 대응방안

위와 같은 취약점들을 방지하기 위해서는 통신망에서의 올바른 설정 및 표준에서의 명확환 지시를 필요로 한다. 첫째로, 통신망이 암호화 알고리즘을 선택할 시 EEA1이나 EEA2를 우선하여 선택하며, EEA0의 사용을 지양하고 불가피한 상황에서는 사용자 개인정보 유출이 되지 않도록 주의해야 한다. 또 한편으로, 이동통신 표준에서는 암호화를 하지 않는 EEA0 알고리즘도 제공하고 있으며, 암호화의 사용은 이동통신망 운영자의 선택이라고 말하고 있다. 하지만 본 논문의 분석 결과를 보면, EEA0 알고리즘을 사용할 시 이동통신 사용자에 대한 데이터가 공격자에게 노출될 수 있다. 따라서 표준에서도 EEA0를 사용하였을 때, 주고받을 수 있는 메시지를 명확하게 명시할 필요가 있다.

VI. 결론 및 향후 계획

본 논문에서는 암호화 되지 않은 메시지 및 노출에 민감한 정보들을 LTE 제어평면 메시지들의 비교 분석을 통하여 효율적으로 찾아낼 수 있는 방법을 고안하고, 이를 통하여 한국 및 미국의 5개 통신사의 제어평면 메시지 분석을 수행하였다. 그 결과, MITM 공격자에게 노출되었을 때, 악용 가능한 두 가지 중요 정보인 IMEI와 SMS 데이터가 암호화없이 주고받는 것을 확인하였다. 또한, 이러한 메시지가 암호화되지 않는 근본적 원인이 망에서의 잘못된 설정으로 암호화 알고리즘의 잘못된 설정이라는 것을 확인하였다.

향후연구로는, 본 연구의 주요 방법인 수집된 데이터를 기반으로 하는 수동적 분석을 확장하여 소프트웨어 정의 라디오 장비와 공개코드기반 LTE 스택[3]을 이용한 능동적 분석 연구를 수행할 계획이다. 본 연구에서 논의한 취약점의 근본 원인이었던 암호화 알고리즘의 잘못된 선택과정에 대해, 이동통신망이 의도적으로 암호화 과정을 무시하도록 하는 단말의 요청에 대한 응답을 조사함으로써 통신사별 보안성 분석을 수행하고자 한다. 또한 이와 함께, 공격으로 이어질 수 있는 정보들의 노출여부 분석을 추가적으로 진행하고자 한다.

[참고문헌]

[1] 3GPP. TS 33.401. 3GPP System Architecture Evolution (SAE); Security architecture, 2018.

[2] 3GPP. TS 24.301. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2017.

[3] srsLTE. Open source 3GPP LTE library. https://github.com/srsLTE/srsLTE

[4] Shaik, A., Borgaonkar, R., Asokan, N., Niemi, V. and Seifert, J.P., Practical attacks against privacy and availability in 4G/LTE mobile communication systems. NDSS '16

[5] Hong B, Bae S, Kim Y. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. NDSS '18.

[6] S. R. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. NDSS '18.

[7] Rupprecht, D., Jansen, K. and Pöpper, C., August. Putting LTE Security Functions to the Test: A Framework to Evaluate Implementation Correctness. In WOOT '16.

LTE 무선 접속 과정 취약점에 관한 연구

이은규*, 김용대*

*KAIST 전기 및 전자공학부

A Study on Insecurity in LTE Radio Access Procedure

Eunkyu Lee*, Yongdae Kim*

*Department of Electrical and Engineering, KAIST.

요 약

LTE에서 무선 접속 네트워크는 사용자 단말인 UE와 기지국인 eNB, 그리고 두 구성요소 사이의 무선 인터페이스를 포함하는 네트워크를 의미한다. 무선 접속 네트워크는 단말의 무선 접속 및 이동성을 지원하며 무선 자원의 효율적인 관리를 위한 다양한 기능을 제공하여 최종적으로 이전 세대의 무선 접속 네트워크보다 더 높은 데이터 전송률과 낮은 지연율을 보장한다. 특히 이러한 무선 접속 네트워크에 문제가 발생하여 정상적인 무선 연결을 보장할 수 없을 경우, 사용자는 LTE의 정상적인 서비스를 받지 못하게 된다. 본 논문에서는, 무선 접속 네트워크에서 발생할 수 있는 무선 접속 프로토콜의 근본적인 취약점과 기지국의 잘못된 구현으로 발생할 수 있는 취약점을 밝히고, 소프트웨어 정의 라디오 (Software Defined Radio, SDR) 장비를 활용하여 LTE 단말에 대한 서비스 거부 공격 (Denial of Service, DoS) 및 고유 정보 유출이 가능함을 테스트베드 환경에서 검증하였다.[1)]

I. 서론

LTE에서 무선 접속 네트워크 (Radio Access Network)는 Evolved UMTS Terrestrial Radio Access Network (E-UTRAN)라고도 불리며, 사용자 단말인 User Equipment (UE)와 기지국인 evolved NodeB (eNB), 그리고 두 구성요소 사이의 무선 인터페이스를 모두 포함하는 네트워크를 의미한다. 무선 접속 네트워크는 무선 인터페이스에서 사용되는 프로토콜 스택 (Layer1 ~ Layer3)을 기반으로 단말의 무선 접속 및 이동성을 지원하며 스케쥴링 및 랜덤 액세스 과정 (Random Access Procedure) 등을 수행하여 효율적으로 무선 자원을 관리한다. 따라서 LTE에서 무선 접속 과정은 실질적인 LTE 서비스 사용 데이터인 사용자 평면 (User Plane)보다 앞서게 되며 이 과정에서 문제가 발생하여 정상적으로 프로토콜이 진행되지 않을 경우, 사용자 평면을 포함한 LTE 전체적인 서비스를 받을 수 없게 된다. 따라서 LTE의 무선 접속 네트워크는 예외 상황 또는 의도적인 공격에 대비할 수 있도록 안정성과 보안을 보장할 수 있어야 한다.

본 논문에서는 무선 접속 네트워크에서 발생할 수 있는 취약점을 분석하고 공격을 검증하였다. 공격자는 무선 접속 프로토콜의 근본적인 문제를 악용하여 같은 지역에 있는 모든 사용자가 대상 기지국에 연결할 수 없도록 하는 서비스 거부 공격 (Denial of Service)이 가능하였고, 만일 기지국이 보안을 고려하여 무선 접속 프로토콜을 제대로 구현하지 않았을 경우 공격자가 임의의 사용자 정보를 얻어내는 공격이 가능함을 보였다. 공격의 검증은 테스트베드 환경을 구축하여 검증하였으며, 근본적인 대책과

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2015-0-00403)

대응방안에 대해 논의하도록 하였다.

II. 관련 연구

2.1 srsLTE[1]

srsLTE는 LTE의 구성 요소간의 프로토콜 스택을 일부 구현한 오픈 소스 프로젝트로써 PC와 소프트웨어 정의 라디오 (Software Defined Radio, SDR) 장비를 통해 UE, eNB, Mobility Management Entity (MME)등이 연동된 LTE 환경을 일부 구축할 수 있다. 무선 접속 시 진행되는 프로토콜 또한 구현되어 있으며 필요에 따라서 사용자가 동작을 수정하여 목적에 알맞도록 라디오 장비를 이용하는 것이 가능하다.

2.2 무선 네트워크 고유 정보 노출 연구

공격자가 같은 지역의 사용자의 개인 정보를 얻어내는 연구는 여러 논문에서 발표된 바가 있다. 대부분은 수동적 공격으로 기지국으로부터 방송 (Broadcast)되는 페이징 (Paging) 메시지를 감청하여 메시지 속에 담긴 사용자의 식별자 (Identifier)를 수집하는 공격이다. 다만, 해당 공격은 방송 채널의 메시지를 수집하기 위한 스니퍼 (Sniffer)가 필요하며, 임의의 식별자를 알아내는 정도의 공격만이 가능하였다.

기존의 방송 채널에 노출되는 사용자의 정보를 스니퍼를 통해 수집하는 연구 외에도 [2]에서는 LTE 무선 접속 프로토콜 중 데이터 링크 계층인 레이어 2 프로토콜에서 발생하는 취약점으로 인한 고유 정보 노출을 서술하였다. 그 중 하나의 공격에 따르면 공격자는 무선 접속 프로토콜 (그 중에서도 랜덤 액세스 과정, Random Access procedure)에서 발생하는 취약점을 통해 스니퍼 없이도 공격 대상의 고유한 정보인 식별자 (e.g., IMSI, TMSI)를 공기 중으로부터 얻어내어 라디오 연결의 식별자 (RNTI)와 매핑 (mapping)할 수 있음을 보였다. 더 나아가, 저자는 해당 문제점이 향후 공격자가 같은 지역의 사용자가 어떠한 웹사이트에 접근하는지 알아내는 공격에 사용될 수 있음을 제시하였다.

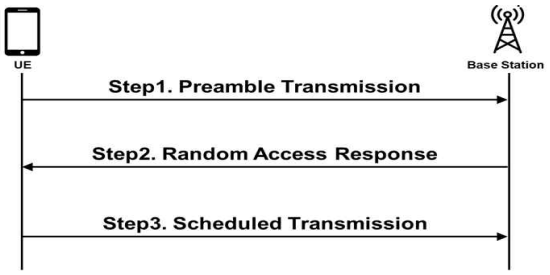


그림 1 랜덤 액세스 (Random Access) 과정

III. LTE 무선 접속 프로토콜

단말이 기지국과 초기 무선 연결을 맺기 위해서는 우선 기지국과 주파수 및 시간 동기화 과정이 필요하며, 이를 위해서 LTE에서는 단말과 기지국이 경쟁 랜덤 액세스 과정 (Contention Based Random Access Procedure)을 거친다. 단말과 기지국의 랜덤 액세스 과정은 그림 1과 같다.

단말은 먼저 기지국에게 자신이 있음을 알리고 동기화를 시작하기 위해 기지국 정보가 담긴 방송 메시지 (Master Information Block, System Information Block)를 읽은 후 (1) 랜덤 액세스 프리앰블 (Random Access Preamble)을 기지국에게 전송한다. LTE에서 프리앰블의 종류에는 총 64가지가 있으며 UE는 이 중 랜덤하게 하나를 선택하여 기지국에 전송한다. (2) 프리앰블을 받은 기지국은 자신에게 접속하고자 하는 UE가 있음을 알게 되고 이후 UE에게 할당한 라디오 자원 정보를 담고 있는 랜덤 액세스 응답 (Random Access Response)을 UE에게 전송한다. 응답을 받은 UE는 (3) 배정받은 라디오 슬롯에 자신이 보내고자 하는 메시지인 RRC Connection Request를 보내면서 초기 무선 접속 과정을 마치게 된다.

IV. 랜덤 액세스 과정에서의 취약점

4.1 의도된 경쟁으로 인한 서비스 거부 공격

4.1.1 랜덤 액세스 과정에서의 경쟁

다만 실제로는 그림 2와 같이 (1) 두 UE가 동시에 같은 기지국에 접속하려고 하는 경우가 발생한다. 만약 두 UE가 보내는 프리앰블이 다른 종류일 경우 기지국은 두 UE를 구분해낼 수

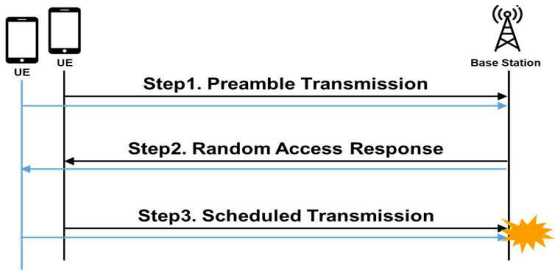


그림 2 랜덤 액세스 과정에서의 경쟁

그림 4 공격 대상 기지국 및 공격자 실험 환경

있지만 두 UE가 같은 프리앰블을 사용했다면 기지국은 두 UE가 보내는 프리앰블을 구분하지 못하게 된다. 이렇게 될 경우 (2) 단말은 기지국으로부터 동일한 랜덤 액세스 응답을 받게 되고 (3) 동일하게 할당받은 라디오 슬롯에 각각의 서로 다른 메시지를 전송하게 된다. 기지국은 서로 다른 메시지를 동시에 받은 후에야 둘 이상의 UE가 동시에 자신에게 접속하려 했음을 확인하게 되며 그중 단 하나의 UE를 선택, 선택받지 못한 UE는 다시 처음부터 랜덤 액세스 과정을 수행하게 된다.

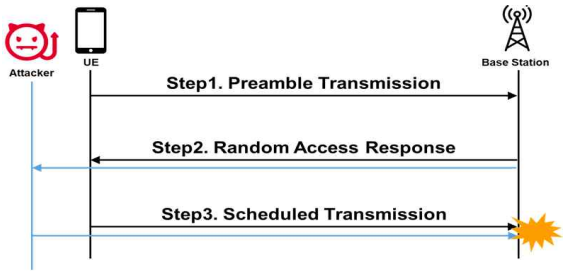


그림 3 서비스 거부 공격 시나리오

4.1.2 의도한 경쟁 시나리오

같은 지역의 공격자는 이러한 경쟁을 악용할 경우 새로운 서비스 거부 공격 (Denial of Service)이 가능하다. 공격의 절차는 그림 3에 간략히 나타내었다. 공격자는 기지국으로부터 전송되는 랜덤 액세스 응답을 계속 주시하고 있다가 랜덤 액세스 응답이 확인되었을 경우 메시지 속에 담긴 할당된 라디오 슬롯 정보를 통해 동시에 메시지를 보내 의도적으로 접속을 시도하는 단말과 경쟁 (Contention)을 유발할 수 있다. 그 경우 기지국은 사용자의 단말과 공격자의 단말 중 하나를 선택해야 하며 만일 공격자를 선택할 경우 사용자는 다시 처음부터 랜덤 액세스 과정을 수행하게 된다. 공격자는 이후 다시 수행되는 랜덤 액세스 과정에 대해서도 똑같이 경쟁을 의도하여 단말이 해당 기지국에 연결할 수 없도록 하는 서비스 거부 공격이 가능하다.

4.1.3 실험 및 결과 분석

실험 환경은 그림 4와 같이 LTE 오픈 소스 프로젝트 (srslte)를 통해 구현한 코어 네트워크에 상용으로 판매되는 기지국 (ipaccess)를 연결하여 LTE 전체 네트워크를 구성하고 실제 상용 실험 단말에 대해 공격을 진행하도록 하였다. 공격자는 마찬가지로 LTE 오픈 소스 프로젝트의 LTE 라이브러리를 참고하여 시나리오처럼 동작하도록 구현하여 단말이 해당 기지국에 접속하려고 시도할 때에 계속 경쟁을 의도하도록 하였다.

실험 결과 공격자가 경쟁을 유발하였을 때 기지국은 무선 신호의 감도가 더 높은 공격자를 우선적으로 선택하였고 따라서 공격 대상이 접속을 시도하려고 할 때 그림 5와 같이 랜덤 액세스 응답 (Random Access Response)만 계속 받고 연결이 되지 않는 것을 확인할 수 있었다. 정상적인 경우 해당 메시지는 단 한 번만 받고 연결이 완료되어야 하지만 공격자가 의도적으로 경쟁을 형성함으로써 랜덤 액세스 과정만을 계속 수행하게 된다.

4.2 랜덤하지 않은 프리앰블

4.1 에서 나타난 공격자는 랜덤 액세스 채널 (RACH)을 계속 주시하며 랜덤 액세스 응답 (Random Access Response) 메시지가 발생하였는지 확인하여야 한다. 이 경우 공격자는 동일한 프리앰블을 보내지 않고서 랜덤 액세스 응답만을 보고 경쟁을 유도할 수 있었다. 공격자가 이렇듯 경쟁을 유도하는 까닭은 사용자가

27	0.26661…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=6[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=102)
29	0.26685…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=8[GroupA]: TA=12, UL-Grant=786688, Temp C-RNTI=107)
31	0.26693…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=20[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=109)
33	0.49084…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=7[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=108)
35	0.49096…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=4[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=110)
37	0.49104…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=13[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=51)
39	0.49120…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=26[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=54)
41	0.71585…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=21[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=52)
43	0.71596…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=27[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=59)
45	0.71603…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=16[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=62)
47	0.71610…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=30[GroupA]: TA=12, UL-Grant=786688, Temp C-RNTI=63)
49	0.71622…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=6[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=70)
51	0.94091…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=32[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=71)
53	0.94104…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=35[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=74)
55	0.94112…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=20[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=76)
57	0.94120…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=28[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=80)
59	1.20494…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=13[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=79)
61	1.20503…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=21[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=84)
63	1.20511…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=36[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=86)
65	1.20517…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=21[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=85)
67	1.20522…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=43[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=87)
69	1.43512…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=28[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=89)
71	1.43521…	MAC-LTE	89	RAR (RA-RNTI=2, SFN=1023, SF=8) (RAPID=33[GroupA]: TA=13, UL-Grant=786688, Temp C-RNTI=94)

그림 5 서비스 거부 공격 상태의 단말이 받은 랜덤 액세스 응답

프리앰블을 전송할 때에는 64가지의 프리앰블 중 랜덤으로 하나를 선택하면서 보내고 공격자는 어느 프리앰블을 사용할지 확인할 수 없기 때문이다.

다만, 실험을 하던 중 발견한 취약점은 단말 중 프리앰블을 전송할 때에 64가지 중 랜덤하게 프리앰블을 선택하여 보내는 것이 아닌 특정 범위의 프리앰블만을 사용하거나 한정된 개수의 프리앰블의 종류만을 사용하는 취약점이 발견되었다. 이 경우 공격자는 사용자가 보낼 프리앰블을 유추하여 프리앰블을 보내는 단계부터 경쟁을 의도할 수 있으며 이 경우에는 랜덤 액세스 응답을 계속 주시하는 것이 아닌 계속해서 프리앰블을 보내는 방법으로도 경쟁을 유도할 수 있다.

4.3 랜덤하지 않은 무선 식별자

모든 사용자는 각각의 무선 연결마다 무선 식별자 (RNTI) 값을 가지며 이 식별자의 값은 무선 연결이 유지되는 동안 계속 같은 값을 갖는다. [2]에 따르면 무선 식별자 별로 할당받은 라디오 슬롯의 스케쥴링 패턴에 따라 사용자가 어느 웹사이트에 방문했는지 파악할 수 있는 부채널 (Side channel)공격이 가능함을 밝혔다. 따라서 LTE에서는 무선 식별자의 값을 16바이트 내에서 랜덤하게 할당하도록 하였다 (특히, 주된 무선 식별자인 C-RNTI의 경우). 무선 식별자는 랜덤 액세스 과정 중 기지국이 랜덤 액세스 응답 (Random Access Response)을 단말에게 보냄으로써 할당된다.

다만, 실험 결과 상용으로 판매되는 펨토셀임에도 불구하고 그림 5와 같이 순차적으로 무선 식별자 값을 증가하여 할당하는 것을 확인할 수 있었다. 이 경우 무선 식별자를 더 쉽게 파악할 수 있으므로 [2]에서 제시한 웹사이트 핑거프린팅 (Fingerprinting) 공격이 더욱 수월해지는 취약점이 된다.

V. 결론

본 논문에서는 단말이 LTE 네트워크에 무선 접속할 때에 발생할 수 있는 취약점을 통해 한 지역의 기지국에 모든 단말을 연결할 수 없도록 하는 서비스 거부 공격 및 고유 식별자 노출 공격이 가능함을 보였다. 무선 접속 단계에서 발견된 취약점은 전체 LTE 서비스를 불가능하게 하므로 해당 문제점의 보완은 필수적이다. 경쟁으로 발생할 수 있는 서비스 거부 공격 취약점은 향후 네트워크에서 보완되어야 할 문제이며 사용자의 고유 정보를 유추할 수 있는 공격의 경우 기지국을 규격에 맞게 구현함으로써 대응할 수 있다.

[참고문헌]

[1] srsLTE. Open source 3GPP LTE library. [Online]. Available: https://github.com/srsLTE/srsLTE

[2] David Rupprecht, Katharina Kohls, Thorsten Holz, Christina Popper. Breaking LTE on Layer Two. IEEE Symposium on Security and Privacy 2019.

[3] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. NDSS 2016.

[4] Roger Piqueras Jover. LTE security and protocol exploits. ShmooCon 2016.

위치기반 키 관리 기법을 적용한 무선센서 네트워크 연구 동향 분석*

김종신*, 권태경*

* 연세대학교 정보대학원 정보보호 연구실

A Survey : Wireless Sensor Networks with Location based Key Management

Jongshin Kim*, Taekyoung Kwon*

*Information Security Lab, GSI, Yonsei University.

요 약

WSN은 물리적 변화를 감지하는 센서가 내장된 노드로 이루어진 무선 네트워크이다. 노드들은 무선으로 통신하며 안전하고 신뢰할 수 있는 암호화 통신이 필요하다. WSN에서는 암호화 통신을 위한 키 관리가 매우 중요한 문제이다. 일반적으로 키 관리 스킴은 사용하는 키에 따라 대칭키/비대칭키 스킴으로 구분하거나 네트워크 토폴로지에 따라 계층적 구조/분산 구조 스킴으로 나뉘기도 한다. WSN에서는 노드의 자원이 한정적이기 때문에 계산이 복자한 비대칭키는 잘 쓰이지 않는다. 이 논문에서는 기존의 키 관리 스킴에 노드들의 위치 정보를 함께 활용한 위치기반 키 관리 기법을 소개하고 그 동향을 함께 조사했다. 또한 동적 키 관리 기법을 소개하고 향후 WSN 키관리 분야가 나아가야 할 방향을 함께 제시하고 있다.

I. 서론

무선 센서 네트워크(WSN, Wireless Sensor Networks)는 각 노드에 빛, 열, 압력 등과 같은 물리적 변화를 감지하는 센서가 내장된 센서 노드의 그룹이다. 수백~수천 개의 센서 노드가 네트워크를 형성하며 무선을 통해 서로 통신한다. WSN은 전 세계에서 주목받고 있는 중요한 연구 분야이다. 무선 센서 네트워크는 군사, 의료 분야에서 표적 식별 및 감시, 가정 자동화, 생활 보조 등 다양한 분야에서 널리 쓰이고 있다. WSN의 다양한 센서 노드들에 의해 수집된 데이터는 무선 채널을 사용하여 기지국(Base Station)으로 전달된다. 센서 노드들은 Ad-hoc 네트워크를 형성할 수 있으며, 무선 통신에서 안전하고 신뢰할 수 있는 암호화 통신이 필요하다. 암호화 통신을 위한 키 관리는 WSN 기술의 중요한 쟁점 중 하나이다.

WSN에서 키 관리 프로토콜은 개별 센서 노드 간(유니 캐스트), 센서 노드 그룹 간 (멀티캐스트), 노드와 기지국 간 (브로드 캐스트)에 중요한 통신을 수행한다. 키 관리 기법의 주요 목적은 데이터의 무결성과 정확성을 유지하여 변조되지 않도록 데이터를 보호하는 것이다. 이를 달성하기 위해 키 관리 프로토콜은 주로 키 공유, 키 탐색, 경로 키 설정, 키 폐기, 키 재설정 등의 프로세스가 필요하다. 생성된 키는 일정한 수명을 가지며 이를 키 관리 프로세스의 사이클이라고도 한다. 사이클의 주기는 키 생성, 하위 프로세스 키의 사용, 저장, 백업, 복구에서 시작하여 키 폐기에서 끝나게 된다.[6]

일반적으로 WSN의 키 관리 문제는 노드의 자원 제약 조건 때문에 상대적으로 자원 제약이 적은 저렴한 대칭키를 사용한다. 모든 센서 노드에 동일한 키를 배치하는 확정적인 스킴부터 각 센서 노드가 배치되기 전에 무작위로 키 풀 (Key Pool)을 부여받고 그중 일부를 사용하여 키를 공유하는 확률적인 스킴까지 다양한 방법이 있다. 확정적 스킴의 경우 센서 노드는 손상에 민감하며 단일 공통키의 누출로 인해 전체 네트워크의 통신이 위험해질 수 있다. 반면, 확률적 스킴은 인접한 두 노드가 특정 수 이상의 공통키를 공유하는 경우에만 통신할 수 있다. 각 센서 노드에 저장할 키의 수를 늘린다면 안전한 링크를 형성할 가능성이 증가한다. 따라서, 확률적 스킴은 확정적 스킴보다 저장해야 할 키의 수가 많아서 보다 큰 스토리지가 필요하지만 보안적 관점에서는 훨씬 안전하다. 하지만 이러한 기법들은 노드 배치 후 센서 노드의 위치 정보를 고려하고 있지 않다.

WSN에서 노드의 위치정보가 주는 의미는 다양하며 활용 가능성이 매우 높다. 배치된 노드의 위치에 따라 지리적으로 의미 있는 데이터를 수집할 수 있고, 구성된 토폴러지에 따라 통신 효율과 에너지 효율에 영향을 주어 네트워크 수명을 좌우하기 때문이다. WSN상의 위치기반 키 관리 기법에서는 네트워크 지역을 그리드로 나누어 노드가 배치되는 좌표를

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

활용하거나 노드 통신 시 주고받는 신호의 강도(RSSI), 신호가 오는 방향의 각도 및 상대적 위치 정보 등을 통해 키를 설정하게 된다. 일반적으로 노드의 위치는 파악되어있다고 가정하며, 이러한 정보를 활용할 경우 기존의 방식보다 제한된 자원에서 효율적인 프로세스가 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 WSN상의 다양한 위치기반 키 관리 기법 및 연구 동향을 설명하고, 3장에서는 WSN에서의 안전한 통신 프로토콜을 위한 향후 연구 방향에 대해 살펴본다. 마지막으로 4장에서 결론을 맺는다.

II. 위치기반 키 관리 기법

Zhu 의[1] 연구는 하이브리드 방식을 사용하며, 키 배포와 노드 인증을 동시에 제공할 수 있는 LEAP (Localized Encryption and Authentication Protocol)을 제시한다. 이 프로토콜은 네트워크 전체, 클러스터, 그룹 및 페어-와이즈 키 설정 기능을 제공한다. 이를 위해 LEAP은 개인키, 그룹키, 클러스터 키, 페어-와이즈 키로 공유된 네 가지 유형의 키를 사용한다. 개인키는 각 센서 노드가 싱크 노드와 통신하기 위해 사용하는 고유한 키다. 그룹 키는 싱크 노드에서 모든 센서 노드로의 통신을 위한 네트워크 차원의 키다. 싱크 노드의 브로드 캐스트 인증을 위해 μTESLA라는 브로드 캐스트 인증 메커니즘이 사용되며, 이는 그룹키와 함께 전송된 패킷이 싱크 노드에서만 전송되도록 보장한다. 클러스터 키는 클러스터 내의 로컬 브로드 캐스트에 사용되는 키이며, 마지막으로 페어-와이즈 키는 이웃 노드들 간의 안전한 통신을 위해 사용된다.

LEAP은 사전 분배 키를 사용하여 네 가지 유형의 키를 설정한다. 개별 키는 먼저 시드의 함수와 노드의 ID를 사용하여 설정된다. 그 다음 페어-와이즈 키 단계에서 이웃을 탐색하고 노드는 자신의 ID를 브로드 캐스팅 한다. 수신 노드는 초기 키가 있는 함수를 사용하여 이 노드와 모든 이웃 노드간 공유 키를 계산하게 된다. 그 후 생성된 초기 키와 모든 중간 키가 지워진다. 세 번재로, 클러스터 키는 페어-와이즈 키로 보호받는 양방향 통신으로 클러스터 헤드에 의해 분배된다. 마지막으로 네트워크 전체의 그룹 키를 배포하기 위해 싱크 노드는 멀티 홉 방식으로 가장 가까운 클러스터부터 브로드 캐스트 한다.

LEAP에는 μTESLA 및 단방향 키 체인 인증, 키 폐기 및 키 재설정 기능이 있다. LEAP의 계산 비용은 노드의 수에 반비례하며 네트워크 밀도가 높을수록 클러스터 당 더 많은 연결이 형성되기 때문에 이웃 노드 수에 비례한다. 스토리지 비용은 페어-와이즈 키 설정이 1 홉 이웃 노드에만 적용되므로 합리적이라 할 수 있다. LEAP은 보안 및 운영적인 요구사항을 대부분 만족시키는 편이다. 하지만 싱크 노드가 절대적으로 손상되지 않아야 하고, 센서 배치 직후에 보안 부트스트랩 시간이 있으며, 이 시간 내에는 공격자들이 노드를 조작할 수 없다는 가정이 필요하다.

일반적인 키 관리 시나리오에서는 센서 노드가 배포되기 전에 하나의 공통키를 부여받고, 배치된 후 이 공통키를 활용하여 이웃들과 다른 키를 도출한다. 이후 각 노드는 공통키를 삭제하여 단일 노드가 손상되더라도 공격자가 접근할 수 없게 한다. 하지만 이는 공격자가 네트워크 전개 후 일정 시간동안 침임할 수 없다는 가정 하에 실행되는 것이며, 모든 노드가 동시에 배포되어야 한다. 최초에 배치된 노드는 공통키를 파괴하여 나중에 배포되는 노드와 통신할 수 없기 때문이다.

Anjum이[2, 3] 제안하는 위치의존 키 관리 기법 LDK (Location Dependent Key Management)는 WSN 지역에 배포된 센서 중 앵커라고 하는 특수 노드가 있다고 가정한다. 앵커 노드는 기능적으로 다른 센서 노드와 동일하지만 서로 다른 전력 레벨로 메시지를 전송하며 변조 방지 기능까지 제공한다. LDK는 노드의 정확한 위치나 배치 정보가 필요 없으며 사전 배치, 초기화, 통신의 3단계로 구성되어있다.

사전 배치단계는 노드가 배치되기 전에 일어나는 작업이다. 우선 최초의 신뢰할 수 있는 기관은 키 풀을 선정하고 센서 노드는 키 풀에서 R개의 키로 구성된 키 링, 해시 함수 H, 네트워크 공통키 K가 로드 된후 네트워크 지역에 무작위로 배치된다. 앵커 노드는 해시 함수 H와 공통키 K만 로드되어 센서 노드와 같이 무작위로 배치된다.

초기화 단계에서 앵커 노드는 거리에 따라 다양한 전력 레벨로 넌스 메시지를 전송하며 이때 메시지는 K로 암호화된다. 서로 가까이 있는 노드들은 비슷한 넌스 메시지를 수신할 것이다. 노드는 이를 다시 K로 복호화한 뒤 보유한 키 링 R과 해시 함수 H를 이용해 새로운 키 링 R`을 도출한다. 마지막으로 통신 단계에서는 임의의 수 α를 선택 후 R`의 모든 키로 암호화 하여 주변의 다른 노드로 전송한다. 암호화된 α를 수신한 다른 노드는 자신의 R`로 복호화한 뒤 암호화 된 α와 몇 개나 일치하는 지 확인한다. 이 때 일치하는 암호문의 개수가 이후에 양 노드 간에 통신에 사용될 도출된 키의 개수를 의미한다. 두 노드 간 동일하게 도출된 키의 개수가 설정된 임계치보다 클 경우 보안링크가 설정된다. 보안 링크는 동일하게 도출된 키들을 함수 H의 결과이다.

LDK는 노드 간의 상대적 위치에 의존적이기 때문에 네트워크 지역상 노드의 정확한 좌표 정보를 요구하지 않는다. 하지만 통신 간섭이 발생할 경우 센서 노드가 새로운 통신키를 도출하는데 필요한 논스를 정상적으로 수신하지 못해 크게 성능이 저하될 수 있다.[9]

Choi 등[4]의 연구에서는 LDK를 기반으로 하여 LDK+를 제안했다. LDK+에서는 계층적 WSN 지역의 그리드 정보를 활용하여 손실되는 논스를 보정하는 단계가 추가하여 새로운 키 생성 과정을 제안한다. 프로세스는 사전 분배, 초기화, 키 생성, 키 동의의 4단계로 구성되어 있다.

LDK+는 통신 간섭으로 인해 논스가 손실되어도

표 1. 위치기반 및 동적 키 관리 기법 연구 서베이 요약

문 제	저 자	접근법	네트워크 구조	메커니즘
Location-based Key Management Scheme	Zhu et al. [1]	Deterministic	Distributed	Pair-wise : BS oriented Group-wise : Symmetric Network-wise : TESLA base
	Anjum [2, 3]	Probabilistic	Hierarchical	Anchor Node RSSI, Pair-wise
	Choi et al. [4]	Probabilistic	Hierarchical	Anchor Node RSSI, Pair-wise, Grid Information.
	Kwon et al. [5]	Hybrid	Hierarchical	Grid Information, Pair-wise Inner Area : Deterministic Border, Outer Area : Probabilistic
Dynamic Key Management Scheme	Nelson [6]	Probabilistic	Hierarchical	EBS Structure (Exclusion Basis System)
	Sahingoz et al. [7]	Hybrid	Hierarchical (support p2p)	UAV - Node : Asymmetric Key Node - Node : Symmetric Key
Restore WSN connection using Drone [8]	T. Truong et al. [8]	Drone	Hierarchical	Works on Star Topology. Replace Node by Drone to Recover network.

배치된 노드간에 서로의 논스를 보정해주어 연결성이 향상되었다. 하지만 정확한 그리드상의 노드 좌표 정보가 필요하며 키 업데이트 단계에서는 K_c를 안전하게 다루어야한다는 한계가 있다.

Kwon 등[5]은 기존 위치기반 키 관리의 한계점으로 거론되는 통신 간섭과 배치 오류 문제를 보완하고 개선된 연결성, 복원력을 제공하는 FRP (Full and Random Pairwise key predistribution)를 제안했다.

FRP는 네트워크 지역을 그리드로 나누고 각 그리드는 내부 (Inner), 경계 (Border), 외부 (Outer) 영역으로 구분한다. 정상적으로 배치가 된 노드들은 내부 영역에 배치되며 확정적으로 키를 교환한다. 배치 오류 문제로 인해 경계 영역과 외부 영역에 배치된 노드들 중 통신키를 설정하지 못한 노드들은 경로키 설정 과정을 통해 연결성을 갖게 된다. 프로세스는 크게 배치 정보를 활용한 사전 분배, 공유키 발견, 경로키 제안을 통한 경로키 설정으로 총 3단계로 구성되어 있다.

우선 네트워크 규모 N은 총 노드의 개수를 의미하며 네트워크 지역은 $t \times s$의 그리드로 분할된다. 각 그리드에는 $n = \frac{N}{t \times s}$개의 노드가 배치된다. 배치에 앞서 사전 분배 단계에서 목표로 하는 네트워크 지역 연결성 p'_c와 인접 영역에 속한 이웃의 비율 r_a를 설정해야 한다. 인접 영역에 있는 두 노드가 키를 공유할 최소 확률 $p' = \frac{p'_c - (1 - r_a)}{r_a}$를 계산하면 각 노드에 최초에 로드되어야 할 키의 개수를 계산할 수 있다. 그리드의 위치에 따라 3개, 5개, 8개의 인접 영역을 가질 수 있으며 인접한 영역의 수에 따라 $m = (n-1) + \{3, 5, 8\} np'$개의 키가 노드에 로드되어야한다. m은 각 그리드의 내부 영역에 배치된 노드 n개중 자신을 제외한 $n-1$개의 노드와 공유할 키 그리고 인접 영역의 수와 앞서 계산된 p'를 통해 필요한 추가적인 키를 포함한다.

공유키 발견 단계에는 노드 u_{ij}^k(그리드 [i][j]의 k 번째 노드, $1 \leq k \leq n$)가 배치된 후 자신의 신원인 <i, j, k>를 전송범위 내로 브로드캐스트 한다. 이를 수신한 노드가 만약 동일한 영역의 노드라면 로드된 m개의 키에서 무조건 공유키를 찾을 수 있다. 인접 영역의 노드라면 p'의 확률로 공유키를 발견한다. 서로 관계없는 노드라면 메시지를 파기한다. 이러한 방식으로 노드들은 완전/부분적 one-hop 서브 그룹을 형성한다.

경로키 설정 단계는 공유키 발견 단계에서 노드의 신원을 정확히 수신했지만 공유키를 생성하지 못한 신원 불명의 노드를 위한 키 설정 단계이다. 이 단계가 필요한 노드들은 영역의 경계/바깥 영역에 있을 수 있다. 이웃하지만 공유키를 생성하지 못한 두 노드는 서로의 one-hop 서브 그룹이 겹치거나 몇 홉 이내에 도달해야한다.

기존의 확률적 스킴에서는 경로키 생성을 위해 암호화 되어 신뢰할 수 있고 제일 짧은 경로를 찾는 과정에서 많은 노드들이 재귀적으로 쿼리해야해서 비용이 많이 들고 길이가 길수록 경로키가 결정 될 가능성이 줄어든다. 하지만 다행히도 FRP에서는 이웃 노드의 절반 이상이 평균적으로 키를 발견하고 2개 이상의 hop을 갖는 노드가 적기 때문에 기존의 방법을 사용하더라도 큰 성능저하가 없을 수 있다. 2-hop 경로의 수에 변동이 없다면 FRP 경로키 생성 단계가 더 효율적이다.

FRP는 배치 오류 문제에 대해 탄성을 가진다. 또한 연결성, 복원력, 통신 오버헤드 등에 대해 우수한 성능을 보여주며 노드에 저장되어야 할 키의 수를 계산하여 스토리지 또한 효율적으로 사용한다. 추가적으로, 정확한 배포 지식까지 갖추게 된다면 보다 뛰어난 확장성을 기대할 수 있다.

III.향후 연구 방향

동적 키 관리 기술은 주기적 또는 키 설정 요청이 발생했을 때 키 재설정을 수행하는 일련의 작업으로 구성된다. 통신 오버헤드가 조금 증가할 수도 있지만, 동적 키를 사용하면 스토리지를 최적화 할 수 있으며 잦은 키 업데이트를 통해 공격자를 줄여 네트워크 생존 가능성과 유연성을 향상시킬 수 있다.

Nelson 등[6]은 WSN에 사용되는 에너지 인식 및 그리드 취기 기반의 계층적 동적 키 관리 프로토콜을 제안한다. 고유한 ID와 주 암호화 키는 기지국에서 노드 그룹으로 형성된 클러스터로 제공된다. 클러스터 헤드는 물리적 위치에 기반하여 선택된다. 클러스터는 가상의 그리드 단위로 나뉘며 각 클러스터 헤드는 여러 그리드를 제어한다. 그리고 이 가상의 그리드에서 에너지 소비를 줄이기 위해 효율적인 동적 키가 생성된다.

Sahingoz가[7] 제안한 다단계 동적 키 관리 시스템은 UAV(Unmanned Aerial Vehicle)를 활용하는 시스템이다. UAV는 이동식 인증기관(MCA, Mobile Certification Authority)으로, WSN의 노드에 공개키를 배포하며, 노드들은 이 공개키를 활용하여 이웃 노드들과 공유 쌍키를 설정하게 된다.

Sahingoz의 스킴은 스토리지 문제 이 외에도 뛰어난 시스템 확장성, 키 분배/업데이트 시 적은 통신량, 노드의 계산량 감소, 네트워크 수명 증가 등 다양한 장점이 있다. 하지만 이 모든 장점에서 UAV가 핵심인 만큼, 만약 네트워크 지역이 전투 지역 및 적대적인 환경이라면 공격자의 UAV 탈취에 특별히 주의해야 한다.

이와 유사하게 Truong 등[8]은 WSN지역에 드론을 배치하여 연결을 복원하는 시나리오는 발표했다. 시나리오에는 Iris+드론과 명령처리와 통신을 위한 라즈베리 파이(Rasberry Pi, IEEE 804.15.4), 노드 탈부착을 위해 사용되는 자석 EPM(eletro magnet permanent device) 등이 사용된다. 정상적인 스타 토폴로지의 네트워크 환경에서 특정 노드의 데이터가 모니터링 되지 않고 통신되지 않는다면 PC(Pan Coordinator)는 복구 에이전트(드론)에 복구 요청을 한다. 요청을 받은 드론은 새로운 노드를 손상 노드 지역으로 운반하고 새 노드와 PC의 연결 설정을 지원한 뒤 복귀하는 시나리오이다.

IV. 결론

본 논문에서는 WSN에서의 위치기반 키 관리 기술에 대해서 살펴봤다. 표 1은 WSN상의 위치기반 키 관리 기법 및 향후 연구 방향을 요약한 것이다.

WSN은 4차 산업 혁명의 핵심 기술 중 하나인 IoT의 기반이 되는 기술이다. 다양한 실제 데이터를 측정하여 활용하기 때문에 WSN의 보안은 매우 중요하다. 하지만 노드들의 배터리, 계산 능력 등의 자원은 매우 한정적이며, 자원 개발만으로 이 문제를 해결하기에는 한계가 있다. 현제 이에 적합한 보안 매커니즘을 위해 많은 연구가 진행 중이며, 앞으로 경량 키 관리 프로토콜을 위한 다양한 노력과 시도가 필요하다.

[참고문헌]

[1] S. Zhu, S. Setia, S. Jajodia, "LEAP+: Efficient security mechanisms for large-scale distributed sensor networks," ACM Transactions on Sensor Networks (TOSN), vol. 2, Issue 4, Nov. 2006

[2] F. Anjum, "Location dependent key management using random key-predistribution in sensor networks," WiSe '06 Proceedings of the 5th ACM workshop on Wireless security pp. 21-30, Sep. 29. 2006

[3] F. Anjum, "Location dependent key management in sensor networks without using deployment knowledge," Wireless Networks, vol. 16 Issue 6, Aug. 2010

[4] J. Choi, J. Bang, LH. Kim, M. Ahn, T. Kwon, "Location-based key management strong against insider threats in wireless sensor networks," IEEE System Journal, 2017

[5] T. Kwon, JH. Lee, JS. Song, "Location-Based Pairwise Key Predistribution for Wireless Sensor Networks," IEEE Transactions on Wireless Commuications, Vol. 8, Issue. 11, Nov. 2009

[6] SC. Nelson, JD. Rose, "Location and Energy Based Hierarchical Dynamic Key Management Protocol for Wireless Sensor Networks," International Conference on Next Generation Computing Technologies, pp. 198-211, June. 2018

[7] OK Sahingoz, "Large scale wireless sensor networks with multi-level dynamic key management scheme," Journal of Systems Architecture, vol. 59, no. 9, pp. 801 - 807, 2013

[8] TT. Truong, KN. Brown, CJ. Sreenan, "Deploying a Drone to Restore Connectivity in a WSN," International Conference on Embedded Wireless Systems and Networks, pp. 239-240, 2016

[9] Y. Wu, J. Stankovic, T. He, S. Lin, "Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks," IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, pp. 13-18, April. 2008

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

구두발표 트랙 D

Mixnet 프로토콜을 통한 자율주행차 운영환경 프라이버시 데이터 수집 방안 연구

강 기 완, 서 정 택

순천향대학교 정보보호학과

Research on collecting privacy data of self-driving vehicle operating environment through Mixnet protocol

Ki Wan Kang, Jung Taek Seo

Department of Information Security Engineering, Soonchunhyang University.

요 약

현재의 자율주행환경에 적용중인 통신 보안 표준은 대표적으로 IEEE(Institute of Electrical and Electronics Engineers) 1609.2 표준, SCMS(Security Credential Management System)규격 및 CAMP VSC3(Crash Avoidance Metrics Partnership Vehicle Safety Communications 3)가 있다. 해당 표준 및 규격은 신뢰할 수 있는 제 3자에 기반한 인증서를 통해 프라이버시 데이터에 대한 보안성을 제공한다. 다수의 인증서를 차량 안에 저장하고, 저장된 인증서들 중 임의로 하나의 인증서를 선택하여 데이터 통신에 이용한다. 인증서를 사용하기 위해서는 정책기관, 최상위 인증기관, 차량인증기관, 등록대행기관의 구성이 필요하다. 까다로운 구성이외에도 최근 인증서와 관련된 취약점이 많이 발견되고 있으며, Active-X 등과 같이 특정 운영체제 및 환경에서만 작동하여 우리나라는 인증서를 폐지하려 하는 움직임을 보이고 있다. 본 논문에서는 Mixnet 프로토콜을 자율주행환경에 적용함으로써 신뢰할 수 있는 제 3자 없이 프라이버시 데이터에 대한 보안성을 제공한다.

I. 서론

현재 지속적으로 자율주행차 주행에 관련된 다양한 기술들이 발표되고 있다. 정책들은 구체화 되고, 주행에 필요한 기술들은 지속적으로 발표됨으로써 자율주행차의 완성도를 높이고 있지만 보안적인 측면에서 많이 부족한 실정이다. 자율주행차에 많은 기술들을 접목시킴으로써 과거의 자동차와 다르게 복잡화, 지능화 되어가고 있다. 특히 시간이 지날수록 도로/교통 인프라와 자율주행차 간의 연결성이 증가함에 따라 공격자의 선택지 또한 같이 증가하고 있다. 2016년에 펜테스트파트너스社는 미쓰비시 아웃랜더 차량의 와이파이 접속구간을 해킹하여 원격에서 차량의 잠금을 해제하고 헤드라이트, 공조장치 등을 임의로 제어한 사례도 있다. 같은 해인 2016년에 중국 텐센트의 보안자회사 킨시큐리티 랩은 테슬라의 웹브라우저의 무선 접속과정에서 해킹에 성공하여 14마일 떨어진 곳에서 주행 중인 차량의 브레이크를 조작하여 급정거시킨 사례도 있다. 테슬라는 이러한 공격에 대해 바로 보안패치를 하였으나, 다음 해인 2017년 킨시큐리티 랩은 패치된 시스템의 보안망을 뚫고 다시 해킹에 성공하여 Escar Europe 2017 등에서 발표하기도 하였다.[1] 위와 같이 시스템에 악의적으로 접근하여 자율주행차의 비정상적인 동작 외에도 도난, 인명피해 등이 발생가능하다.

자율주행차는 주행 중 다양한 데이터 통신을 진행하게 된다. 위치 정보, 목적지 정보, 주행거리 등 다양한 정보들을 주변의 자율주행차와 인프라에 전송함으로써 운행에 필요한 정보들을 얻게 된다. 얻은 정보를 바탕으로 운전자를 안전하고 빠르게 목적지에 데려다 줄 수 있는 경로를 판단하게 되는데 해당 데이터에서 위변조가 발생하였을 시 도난, 인명 피해 등이 발생가능하다. 현재 IEEE 1609.2 표준, SCMS(Security

Credential Management System) 규격을 적용하여 자율주행차 운영 환경을 구축하고자 한다. IEEE 1609.2 표준, SCMS 규격은 인증서를 활용하여 신뢰할 수 있는 통신을 구현하게 되는데 최근 인증서를 폐지하려는 국내의 동향과는 적합하지 않은 점을 바탕으로 Mixnet 프로토콜을 활용하여 신뢰할 수 있는 통신을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 자율주행차 개념 및 자율주행환경에 현재 적용 중인 보안 표준 및 규격에 대해 분석하고, 3장에서는 Mixnet 프로토콜을 활용하여 신뢰할 수 있는 통신 구현에 대해 제안한다. 4장에서는 현재 적용하려는 표준 및 규격과 본 논문에서 제안하는 기술과의 차이점을 다룬다. 마지막으로 5장에서 결론을 맺는다.

II. 자율주행차

2.1 자율주행차 개념

국내 「자동차관리법」 제2조 제1의 3호에서는 자율주행차를 '운전자 또는 승객의 조작 없이 자동차 스스로 운행이 가능한 자동차'로 정의 되어있다.

자율주행차의 주행과정은 사람이 운전하는 것과 동일하게 "인지(센서를 통해 주변 상황에 대한 정보를 수집하고 인지)-판단(인지된 정보를 바탕으로 주행상황을 판단하고 경로를 따라 운행)-제어(결정된 판단을 실제 주행에 반영하기 위해 운전 시스템을 제어)" 3단계 과정을 거쳐 상황을 판단하고 결과에 따른 주행을 하게 된다.

자율주행차의 개발에 있어 최대 촉매제의 역할을 하는 것은 전산 처리 속도 발전에 있음을 알 수 있다. 특히 1980 년대 이전에 자율주행차 하드웨어 구성요소들인 라이다, 레이더, 초음파 센서 및 카메라 개발이 완료, 상용화되어 있었음이 사실을 뒷받침한다.

2.2. 자율주행환경 관련 보안 표준 및 규격

2.2.1 IEEE 1609.2 표준

IEEE 1609.2 표준은 자동차 환경에서의 무선통신 표준인 WAVE(Wireless Access in Vehicular Environments) 관련 표준으로써, 차량이 다른 차량이나 외부 시스템과의 무선 통신 시 준수해야 할 보안 표준이다. 통신량의 급증에 따라 도청, 스푸핑, 패킷 재사용 등과 같은 공격들의 발생 횟수도 함께 증가하였다. IEEE 1609.2 표준은 통신상에서 발생할 수 있는 공격들로부터 메시지를 보호하기 위한 표준이다. IEEE 1609.2 표준에서 사용되는 기술은 다음과 같다[3].

- 공개키 기반 전자 서명
- AES-CCM(Advanced Encryption Standard -in Counter with CBC-MAC)알고리즘

위와 같은 기술을 통해 자율주행차 간 전송되는 메시지의 위/변조 방지 여부 및 메시지 송신자가 적합한 송신자인지를 식별을 한다. 데이터의 기밀성을 유지할 뿐만 아니라 MAC 값을 추가적으로 생성하기 때문에 데이터의 무결성도 보장한다.

2.2.2 SCMS

SCMS는 자율주행차 간의 통신(V2V) 및 자율주행차와 인프라의 통신(V2I) 통신을 위한 POC(Proof Of Concept) 메시지 보안 솔루션이다. SCMS는 암호화 및 인증서를 사용하여 신뢰할 수 있는 통신을 할 수 있게 하는 PKI(Public Key Infrastructure) 기반 접근 방식을 사용한다. 인증된 시스템 사용자는 SCMS POC가 발급한 디지털 인증서를 사용하여 통신이 허가된 자율주행차에서 발생하는 메시지를 인증하고 검증한다. 차량 소유자의 프라이버시를 보호하기 위해 인증서에는 개인 또는 장비 식별 정보가 포함되어 있지 않지만 시스템의 다른 사용자가 각 메시지의 출처를 신뢰할 수 있도록 시스템 자격증명으로 사용된다[4].

V2V 및 V2I 통신의 신뢰 기반을 형성하는 보안 인증서를 발급 및 관리하기위한 보안 인프라를 제공한다. 연결된 차량 장치는 SCMS에 등록하고 인증기관에서 보안 인증서를 얻은 다음 해당 인증서를 디지털 서명의 일부로 해당 메시지에 첨부한다. 인증서는 장치가 개인 정보를 보호하면서 시스템의 신뢰할 수 있는 행위자를 증명한다. 오작동 탐지 및 보고 기능을 통해 시스템은 불량한 액터를 식별하고 필요한 경우 메시지에 대한 접근 권한을 제거할 수 있다.

2.2.3 CAMP VSC3

CAMP VSC3는 자동차 회사들과 관련 기관들이 참여해 구성한 컨소시엄 CAMP에서 만든 보안규격으로 포드, GM, 혼다, 도요타, 현대기아 등 세계 여러 자동차 회사들이 참여하고 있다. 본 규격에서는 KMS(Key Manager System)과 PKI 기술에 대해 정의 되어 있다[5].

- KMS, 암복호화 키 관리 시스템: 인증서를 포함한 암복호화 키의 생성과 폐기 등 키의 생애 주기별 관리 및 안전한 보관을 위한 시스템이다. 외부 통신뿐만 아니라 차량 내부 ECU (Electronic Control Unit) 통신을 위한 키 관리 및 안전한 저장, 접근 제어 및 권한 관리를 통해 키 오남용 및 도용 방지 등의 기능을 맡아 자율주행차 통신 체계를 안전성을 유지한다.
- PKI, 공개키 인프라: PKI는 차량용 인증서를 생성하고 운영 및 관리하며, 교통관리 시스템은 PKI를 통해 각 자율주행차의 존재를 인식한다.

2.2.4 인증서의 문제점

현재 자율주행차는 Fig 2과 같이 인증서를 통해 신뢰할 수 있는 통신을 구현하고 있다. 하지만 현재까지 발견된 국내의 공인인증서의 문제점을 피해갈 수 없을 것이라고 예측된다.

- 단순한 copy & paste를 통해 이용자의 인증서 개인키가 쉽게 복제 유출
- 다양한 운영체제와 디바이스, N-스크린 환경에 대응하기 어려움
- 인증서를 위하여 추가적으로 설치하는 프로그램으로 인한 Attack Surface 증가

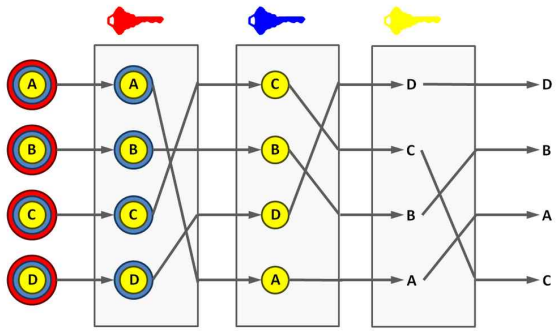
미국 국립표준기술 연구소(NIST, National Institute of Standards and Technology)가 2006년에 발간한 전자인증 가이드라인(electronic authentication guideline)에 의하면, 전자파일 형태로 배포된 인증서(soft crypto token)의 보안강도는 높게 평가되지 않았다. 단순암호(password)보다는 안정성이 높지만, OTP(One Time Password)보다 안전성이 낮게 평가되었다.

III. 자율주행차 운영환경 프라이버시 데이터 수집 방안

3.1. Mixnet 프로토콜 개념

Mixnet 프로토콜은 여러 발신자로부터 메시지를 가져와서 임의의 순서로 다음 대상으로 다시 보내는 믹스라는 프록시 서버 체인을 사용하여 추적하기 어려운 통신을 만드는 라우팅 프로토콜이다. 다수의 프록시 서버를 사이에 두고, 각각의 노드가 암호화된 메시지를 주고받는 과정을 여러번 수행한다. 또한 각 Mix 내부의 순서를 짐작하기 어렵기 때문에 중간에 도청을 당하더라도 공격자는 데이터 송신자와 데이터 간의 연결관계를 추측하기 어렵다[6].

[그림 3]은 간단한 Mixnet 프로토콜의 복호화 과정이다. 메시지는 일련의 공개키로 암호화되고 각 믹스 노드는 자체적인 비공개키를 사용하여 암호화 계층을 제거합니다. 각 노드는 메시지 순서를 섞어 결과를 다음 노드로 전송합니다.



[그림 1] Simple decryption protocol mixnet

3.2 Mixnet 프로토콜이 제공하는 보안 기능

- 익명성: Mixnet의 기본적인 속성으로 발신자와 수신자 간의 추적이 불가능하게 설계되었다. 아무리 뛰어난 공격자도 Minxet을 통해 데이터를 추적할 수 없어야 한다.
- 무결성: 보낸 사람의 메시지는 데이터 손상 없이 익명으로 전송되어야 한다. 무결성 검사는 메커니즘과 체크섬 기술을 통해 대부분의 Mixnet에서 제공한다.
- 검증 가능성: Mixnet은 혼합 출력 배치에서 메시지의 정확성을 검증할 수 있는 수단을 제공한다.

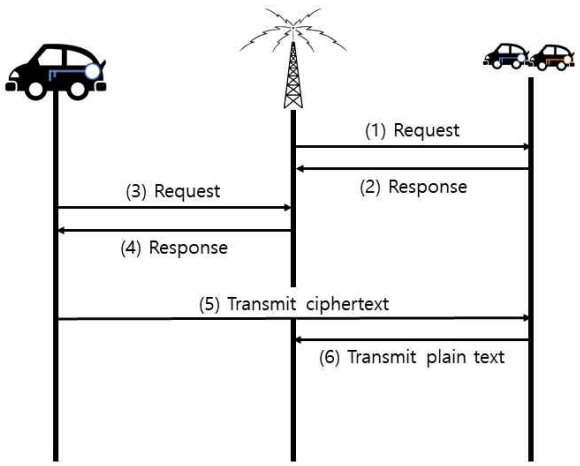
- 공격에 대한 강성:Mixnet은 공격자의 수동적인 공격뿐만 아니라 능동적인 공격에도 내성이 존재한다.
- 내결함성: Mixnet이 작동되는 동안 참여 단계에서 특정 결함이 있는 단계를 허용한다.

3.2 Mixnet 프로토콜을 적용한 자율주행차 운영환경

현재 자율주행환경에 적용중인 표준들은 인증서를 기반으로 익명성, 무결성 등 다양한 보안 요소들을 보장한다. 본 논문에서는 신뢰할 수 있는 제 3자가 필요한 인증서 기반의 프라이버시 데이터 보호가 아닌 Mixnet 프로토콜을 적용함으로써 신뢰할 수 있는 제 3자 없이 다양한 보안 요소들을 제공하는 것을 목적으로 한다. 특히 위치 정보, 목적지 정보 등 사용자의 사생활이 담긴 프라이버시 데이터에 대해 익명성이 필수적으로 필요하다.

Mixnet 프로토콜에서 가장 중요한 것은 다수의 프록시 서버를 가지는 것이다. 이 개념을 자율주행차에 적용하여 각각의 자율주행차가 각각의 프록시 서버가 되어 프록시 서버 체인을 구성한다. 이를 통해 사용자와 사용자 메시지 간의 연결성을 없애 역추적이 불가능하게 한다.

Mixnet 프로토콜을 적용한 자율주행 운영환경에서의 간략한 프로토콜은 [그림 4]와 같다.



*The infrastructure has the public key of every vehicle that accesses the network.

[그림 2] Mixnet protocol apply to self-driving environment

*가정: 2번의 믹스노드를 거침

Step 1. 인프라에 접속한 모든 자율주행차의 공개키를 인프라가 요청

Step 2. 접근한 자율주행차는 인프라에 공개키 전송

Step 3. 데이터를 전송하려는 자율주행차는 인프라에 같은 네트워크에 있는 임의의 자율주행차 공개키 요구

Step 4. 인프라는 요청한 자율주행차에게 임의의 자율주행차 공개키 2개 전송

Step 5. 공개키로 암호화 시킨 메시지를 해당 공개키 소유 자율주행차에게 전송

Step 6. 암호문을 복호화 하여 인프라에 전송

위와 같은 과정을 통해 자율주행환경에 있는 데이터 센터에서는 사용자와 사용자간의 메시지를 추적할 수 없다. 따라서 중간에 공격자가 존재한다 하더라도 도청 및 변조 공격에 대해서 보안성을 제공한다. 또한 공격자가 네트워크 인프라에 접근하여 동시에 모든 차량을 공격하면 데이터에 대해 접근 가능 하지만 도로라는 한정된 공간 안에서 자율주행차는 빠른 네트워크 토폴로지 변화를 가지기 때문에 동시에 모든 차량이 침해되는 확률은 매우 적다.

IV.결론 및 향후 연구방향

현재 자율주행차 운영환경은 IEEE 1609.2, SCMS, CAMP VSC3를 적용 중이다. 인증서를 기반하여 프라이버시 데이터를 보호하고 있으며, 재전송 공격, 도청 공격 등 네트워크에서 발생할 수 있는 공격에 대한 내성을 가지고 있다. 하지만 최근 인증서를 폐지하려고 하는 국내의 동향과는 적합하지 않으며 신뢰할 수 있는 제 3자를 통한 금융거래에서도 지속적으로 취약점과 피해가 발견됨으로써 제 3자에 대한 소비자의 불안감은 계속적으로 존재한다. 본 논문에서 제안하는 Mixnet 프로토콜 기반 통신은 신뢰할 수 있는 제3자가 존재하지 않아도 자율주행차 운영환경

에서 신뢰할 수 있는 통신을 제공한다. 데이터 송신자와 데이터 간의 연결성이 없을 뿐만 아니라 자율주행차가 각각의 노드가 되어 데이터를 임의로 배열함으로써 익명성을 보장하고, 중간에 도청을 당한다 하더라도 사용자와 사용자 메시지 간의 연결성이 존재하지 않아 역추적이 불가능하다.

향후 연구방향으로는 본 논문에서 제안한 Mixnet 프로토콜을 자율주행차에 적용시켜 구체적인 프로토콜을 설계하고 그에 대해 효율성 검증 및 인증서를 사용하는 기존의 방식과의 차이점을 분석하는 연구가 필요하다.

[참고문헌]

[1] Kwon Hyuk Chan, Security Trends for Autonomous Driving Vehicle, Electronic Telecommunications Trend Analysis, Feb.2018

[2] Park Sang Won, Autonomous vehicles, past and present And the future, Heungduk Securities, Feb. 2017

[3] IEEE 1609.2-2013: IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages, Apr, 2013

[4] Security Credential Management System Proof - of - Concept Implementation EE Requirements and Specifications Supporting SCMS Software Release 1.2.2, Made Available to the United States Department of Transportation National Highway Traffic Safety Administration (NHTSA), Nov, 2016

[5] Car security, six key technologies you should know, pentasecurity, Oct.2016D.S.Milojicic, V.Kalogeraki, R.Lukose,K.Nagaraja, J.Pruyne, B.Richard, S.Rollins and Z.Xu, Peer to Peer Computing, HP Laboratories Palo Alto HPL-2002-57, March, 2002.

2차 Rényi 엔트로피 기반 차량용 침입 탐지 시스템

김광식*, 김영식*1)
*조선대학교 정보통신공학부

Vehicular Intrusion Detection System Based on Rényi Entropy

Kwang-Sik Kim* Young-Sik Kim*
*Depart. Information & Communication Engineering, Chosun University.

요 약

본 논문에서는 CAN BUS로 전송되는 데이터에 대한 Rényi 엔트로피 추정을 기반으로 침입을 탐지하는 방법을 제안한다. Rényi 엔트로피는 Shannon 엔트로피를 포함하는 일반화된 개념으로 볼 수 있으며, Rényi 엔트로피를 정의하는 차수 α에 따라 매우 낮은 복잡도로 구현 및 계산이 가능하다. 이 논문에서는 충돌 엔트로피로도 불리는 $\alpha=2$인 2차 Rényi 엔트로피를 직접 사용하여 차량을 위한 침입 탐지 알고리즘을 제안하였다.

I. 서론

오늘날 자동차는 순수 기계 시스템이 아닌 수많은 전자 제어기에 의해서 동작하는 복잡한 시스템으로 구성되어 있다. 내부 전자제어기 간에는 CAN (control area network) BUS 기반의 네트워크를 통해서 데이터를 주고받으며, 전기자동차와 같은 특수한 경우에는 1백 개 이상의 전자제어기가 CAN BUS의 전체 대역폭의 대부분을 사용하고 있는 상황이다.

CAN BUS를 기반의 자동차 시스템은 이미 보편적으로 사용되고 있지만 CAN 프로토콜을 설계할 당시부터 보안에 대한 고려가 전혀 이루어 않았다. 따라서 차량 내부 진단을 위한 OBD(on-board diagnostics)-II 포트 및 인포테인먼트 시스템을 통해 외부에서 차량 내부 시스템으로 쉽게 진입이 가능하기 때문에, 2010년 이후 지금까지 자동차 해킹에 대한 많은 보고가 매년 지속되어 왔다 [1]. 이로 인해 차량 내부 네트워크에 보안 시스템을 설계하고자 하는 연구가 활발하게 이루어졌다. 대표적으로 내부 데이터에 대한 암호화와 인증을 위한 장치 및 내부 디바이스 인증 침입 탐지를 위한 장치에 대한 연구가 있다 [2],[3].

본 논문에서는 CAN BUS로 전송되는 데이터에 대한 Rényi 엔트로피 추정을 기반으로 침입을 탐지하는 방법을 제안한다. Rényi 엔트로피는 Shannon 엔트로피를 포함하는 일반화된 개념으로 Rényi를 정의하는 차수 α에 따라 매우 낮은 복잡도로 구현 및 계산이 가능하다. 이 논문에서는 충돌 엔트로피로도 불리는 2차 Rényi 엔트로피를 직접 사용하여 차량을 위한 침입 탐지 알고리즘을 제안하고 있다.

II. 기존의 차량용 IDS

차량에 대한 침입 탐지 시스템의 기본 원리

1) 교신저자 (iamyskim@chosun.ac.kr)

는 일반 네트워크에서의 침입 탐지 시스템과 동일하다. 첫 번째 방식은 외부의 공격의 시그니처를 저장하고 전송되는 패킷을 검사하는 방식과, 차량에서 생성되는 데이터의 정상적인 범위의 통계적 특성을 사용해서 이상 현상을 감지하는 방법이 적용될 수 있다.

기존 네트워크와 차량을 위한 네트워크의 가장 큰 차이점 중 하나는 차량에서 생성되고 전송되는 메시지는 인간이 만든 메시지와 달리 균일하고 규칙적인 특징을 지닌다는 점이다. 따라서 이상 현상을 규정하는 것이 정규적엔 패턴에서 벗어나는지 여부를 판정하는 것으로 이루어지기 때문에, 오류 확률을 줄일 수 있다. 반면에 차량에서 사용되는 전자 제어기의 연산 능력이 일반 PC 환경보다 더욱 제한되어 있지만, IDS의 특성상 실시간으로 데이터를 처리해야하기 때문에 적용되는 알고리즘의 연산 복잡도에 한도가 있다는 어려움이 동시에 존재한다.

통계적인 특성을 종합적으로 반영할 수 있는 검사 방식 중에 엔트로피 기반의 방식이 있다[3],[4]. 엔트로피를 측정해서 침입을 탐지하는 방식은 이미 기존 네트워크를 위한 침입 탐지 시스템에도 적용하였으나, 차량에 대해서도 이러한 가능성들이 다방면으로 고려가 되었다. 예를 들어 [4]에서는 상대거리(Relative Distance)와 조건부 자기 정보(Conditional Self Information, CSI)을 사용한 침입탐지 방식을 제안하였다. 상대거리는 두 사건 집합의 확률 분포 p와 q가 있을 때 사건 x에 대해서 $p(x)$와 $q(x)$의 상대적인 거리 $RD_{p|q(x)}$로서 다음과 같이 정의된다.

$$RD_{p|q}(x) = p(x)\log_2 \frac{p(x)}{q(x)}$$

이름에서 알 수 있는 것처럼 RD는 두 확률 분포의 상대적인 거리 값에 대한 메트릭으로 사용이 가능하다. 만일 $q(x)$가 정상적인 동작 상태에서의 차량의 내부 데이터의 분포 값이고 $p(x)$가 현재 상태에서의 차량 데이터의 분포 값이라 하면 $RD_{p|q}(x)$가 크다는 것은 $p(x)$와 $q(x)$사이의 거리가 멀고, 이는 $p(x)$의 패턴이 정상과 거리가 있음을 간접적으로 나타낸다. [4]에서는 이와 같은 $RD_{p|q}(x)$를 사용해서 매우 정확하게 이상 값을 찾아낼 수 있음을 보였다. 엔트로피는 차량 내부 데이터의 이상 여부를 나타내는 지표로 사용이 가능하지만, 실시간으로 생성되는 데이터셋의 엔트로피를 추정하기 위해서는 일정 단위로 데이터셋을 수집한 후에 통계 분포를 구한 후에 계산이 가능하다. 이는 실시간으로 일어나는 공격에 전혀 대응하지 못할 뿐만 아니라 내부에서 일어나는 연산 역시 로그의 누적값을 계속해서 관리해야한다는 점에서 복잡도가 작지 않다.

III. Rényi 엔트로피 기반 차량용 IDS

1961년에 Rényi 엔트로피는 다음과 같이 정의가 되었다[5].

$$R_\alpha(x) = \frac{1}{1-\alpha} log_2 \sum_{b \in F_2^L} \Pr(b)^\alpha$$

이 때 $\alpha \to 1$로 수렴할 때 Rényi 엔트로피는 Shannon 엔트로피가 되며, $\alpha \to \infty$일 때는 min-entropy가 된다. 또한 $\alpha = 2$인 경우를 충돌 엔트로피라 하며 다음과 같이 나타낼 수 있다.

$$R_2(X) = -\log_2 \sum_{b \in F_2^L} \Pr(b)^2$$

$\alpha > 1$일 때 Rényi 엔트로피는 Shannon 엔트로피보다 항상 작거나 같은 성질을 지닌다.

[6]에서는 Rényi 엔트로피를 동일 값의 발생 빈도를 사용해서 구하는 Nearest Neighbour Distance 기반의 엔트로피 추정 방법이 소개되었다. 본 논문에서는 [6]에서 소개된 방법 중에서 $\alpha = 2$를 사용할 것이다.

본 논문에서 제안하는 차량 CAN ID에 대한 윈도우 기반의 반복적인 Rényi 엔트로피 추정 기법은 Algorithm 1과 같이 나타낼 수 있다.

Algorithm 1의 i번째 Time Slot에서 발생한 CAN ID가 이전에 발생한 CAN ID와 동일한지를 비교해서 동일한 경우에만 (즉, "충돌"이 일어난 경우) 누적값을 증가시키도록 하고 있다. N_S의 샘플에 대해서 값을 누적 시킨 후 전체 누적값에 가중치가 반영된 상태로 더해진다.

두 가지 지점에 주목할 수 있다. 먼저 제안하는 방식에서는 엔트로피를 추정하기 위해서 먼

Algorithm 1
Input: CAN ID r_i
Output: Rényi 엔트로피 값

```
begin
  S = 0
  while 검사 시작 후 (무한) 반복 do
    A = 0; L = 0;
    for i = 1 to N_S do
      t = r_i;
      if r_i == L then A++;
      L = r_i;
    end for
    S = wS + A;
    R = -log_2[S(1-w)/N_S)];
  end while
    return R;
end
```

저 확률 분포를 계산할 필요가 없다. 연속해서 같은 ID가 생기는 이벤트만 단위 시간 동안 카운트한다. 이런 경우에 정확하게 2차의 Rényi 엔트로피로 수렴함이 [6]에서 제시되고 있다. 따라서 기존 엔트로피 기반 IDS에 비해서 매우 단순한 연산을 통해서 엔트로피 추정이 가능하다. 그 다음으로 마지막 단계에서 로그 연산을 사용하고는 있지만 이는 필수적이지 않은 과정으로 역으로 이상 여부를 판정하는 Threshold를 2의 Exponent로 사용하는 것으로 대체가 가능하다.

즉 로그 함수는 단조 증가함수이기 때문에 $2^R = S(1-w)/N_S$를 통해서 2^R이 이상 여부의 기준이 되는 Threshold보다 큰지 작은지를 평가하는 것으로 충분하다. 위의 알고리즘을 사용해서 CAN ID에 대한 IDS 스킴을 설계하면 다음 Fig. 1과 같은 순서도로 나타낼 수 있다.

IV. 주행 컨텍스트 기반의 IDS

위에서 설명한 알고리즘을 통해서 엔트로피를 추정하는 것이 가능하다. 이 때 어떤 엔트로피값이 수용 가능하고 또는 그렇지 않은지에 대한 기준을 선정해야만 한다. 엔트로피는 장기

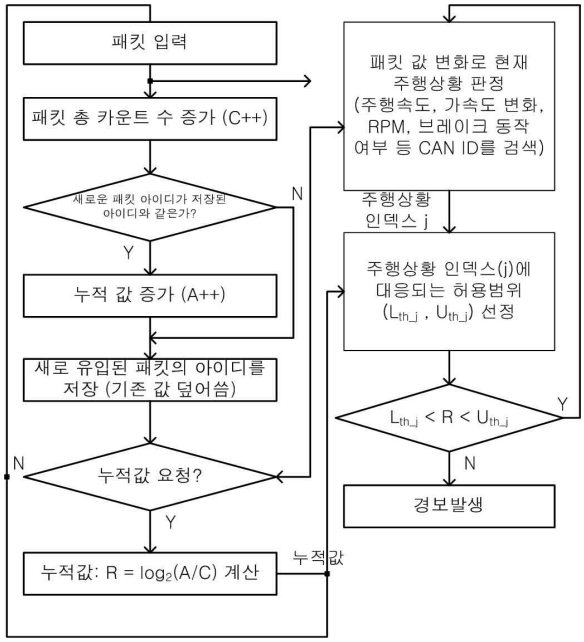


Fig. 1. Rényi 엔트로피 기반 IDS 순서도

간에 걸친 전체적인 확률 분포의 하나의 통계적 요약값이라고 볼 수 있다. 만일 모든 ID가 균일하게 발생한다면 엔트로피 값은 큰 값으로 나타날 것이다. 반대로 ID별로 고유의 확률 무게를 지닌다면 상대적으로 낮은 엔트로피 값이 나오게 된다. 만일 ID의 발생 분포가 어떤 원인에서든 변화한다면 이는 기존 엔트로피 값과 큰 차이를 일으키는 원인으로 작용할 것이다.

문제는 차량이 주행하면서 ID의 발생 빈도도 함께 변할 수 있다는데 있다. 전체 ID를 대상으로 하는 경우 주행 데이터에 맞춰서 발생 빈도가 변화하는 ID들이 있다. 이에 대응하기 위해 엔트로피의 값을 판정하는 기준값도 차량의 주행 컨텍스트에 맞추어서 함께 변해야만 한다. 이를 통해 침입 탐지의 정확도를 더욱 높이는 것이 가능하다.

차량의 주행 환경에 의해서도 정상적인 CAN 패킷의 발생 패턴이 달라지기 때문에 차량용 IDS에서도 차량의 속도 범위, 가속도 변화를 중심으로 차량의 운행 상황을 여러 단계로 나누어서 차량 내부 ID의 Rényi 엔트로피 허용 범위를 정하도록 하였다. 실제 허용 범위는 실제 차량에서 사용하는 CAN 패킷 규격에 따라서 달라지고, 차량의 CAN 데이터에 대한 지정은 차량 제조사마다 다르고, 차종마다 달라지기 때

문에, 특정 차량이 정해지고 CAN 포맷이 완성되면 차량에 맞추어서 허용 범위를 특정해야하는 문제가 있다.

이는 차종에 따라서 제안하는 IDS를 설치한 후에 일정 기간 동안 훈련 기간을 거쳐서 허용 가능한 범위를 결정해야만 한다. 그 이후에는 허용 가능한 범위를 벗어나는 경우에는 위험 신호를 발생하고 그렇지 않은 경우에는 계속해서 모니터링을 실시하게 된다.

차량 운행 환경 변화를 감지하기 위해서 속도 구간별 (예, 0~20 km/h, 20~40km/h, 등) 가속도 변화량 ($\pm 1m/s^2$, $\pm 2m/s^2$ 등)을 사용해서 (속도 구간, 가속도 변화량) 별 운행 환경이 구분하고 각 운행 환경 별로 고유의 인덱스가 사전에 할당한다. 기본 훈련 단계에서는 사전에 정의된 운행 환경 별로 차량의 운행을 실시하여 Rényi 엔트로피를 추정한다. 일정 수 번 이상의 훈련을 통해서 정상적인 상황에서의 허용 가능한 상한값과 하한값이 결정되어 저장한다.

V. 결론

본 논문에서는 2차 Rényi 엔트로피 기반의 차량용 IDS에 대한 기본적인 설계 및 동작 원리를 설명하였다. 실제 특정 차량에 맞춘 실험 결과는 후속 연구 결과로서 발표될 예정이다.

VI. 감사의 글

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 정보보호핵심 원천기술개발사업("자율 주행차를 위한 침입 감내 핵심 요소 기술 개발", IITP-2017-0-00441)의 연구결과로 수행되었습니다.

[참고문헌]

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. of USENIX*, 2011.

[2] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame," in *Proc. 15th Annual Conference on Privacy, Security and Trust (PST)*, 2017.

[3] P. Berezinski, B. Jasiul, and M. Szpyrka, "An Entropy-Based Network Anomaly Detection Method," *Entropy*, vol. 17, pp. 2367-2408, 2015.

[4] M. Müter, Naim Asaj, "Entropy-Based Anomaly Detection for In-Vehicle Networks," in *Proc. 2011 IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, June 5-9, 2011.

[5] A. Rényi, "On measures of entropy and information," in *Proc. the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 20 June - 30 July 1960, pp. 547 - 561.

[6] Young-Sik Kim, "Low Complexity Estimation Method of Rényi Entropy for Ergodic Sources," *Entropy*, vol. 20, no. 9, 657(1-8), Sep. 2018.

차량 사이버 보안 가이드 동향 분석1)

이정호*
*한국국정보인증

Case study of Vehicle Cyber Security Guide

Jung-Ho Lee*
*Korea Information Certificate Authority Inc.

요 약

최근 출시된 차량은 다수의 ECU(Electronic Control Unite)라 불리는 전자제어 장치로 운전자에게 다양한 편의와 주행 안전성을 제공하고 있다. 그리고 차량기술 발전은 기계식 장치의 발전을 통한 주행의 효율성과 안전성에서 점차 ECU를 이용한 전자제어로 운전자에게 다양한 편의성을 제공하는 형태로 진화하고 있다. 그리고 외부 네트워크와 통신 가능한 차량은 원격으로 차량을 제어하여 운전자에게 편의를 제공하고 있다. 이처럼 차량의 전자화와 외부 네트워크와 연결은 차량을 하나의 IoT(Internet of Thing)기기로 인식하게 되었다.

하지만 이러한 차량의 전자화는 기존 IT환경의 해킹위협에 노출되며, 해커에 의한 공격대상이 될 수 있다. 실제 2015년 Charlie Miller와 Charlie Miller는 Jeep Cherokee의 통신 ECU를 원격으로 해킹하여 차량의 제어권을 장악하고 원거리에서 강제제어에 성공하였다. 이처럼 차량 대상의 사이버 위험은 현실이 되었다.

이를 위해 최근 차량 제조사를 포함하여 국가 주도로 차량을 위한 사이버 보안 가이드에 대한 연구가 진행 중이다. 본 논문에서는 최근까지 진행 중인 차량 사이버 보안 가이드에 대한 동향에 대해 알아본다.

I. 서론

최근 IT기술의 발전은 차량기술 발전에 많은 변화를 가져왔다. 과거 차량은 단순 이동수단으로 주행에 대한 안전성과 연비 개선 등 차량의 기계적 장치에 대한 기능개선을 위주로 연구되었다. 그리고 차량기술의 발전과 IT기술과의 융합으로 차량은 운전자에게 ADAS (Advanced Driver Assistance System)와 같은 첨단 기능을 제공함으로써 편의성이 향상되었다. 그리고 가까운 미래에는 차량 스스로 주행하는 자율주행 기능으로 운전자의 조작없이 목적지까지 이동함으로 이동수단의 새로운 패러다임을 가져오게 될 것이다.

하지만 차량은 전자화와 외부 통신으로 해킹위협에 노출되게 되었다. 실제 2010년부터 차량을 대상으로 해킹위협 연구가 시작되었으며, 사이버 보안 전문가들은 차량의 해킹가능성을 경고했다. 그리고 2015년 BlackHat에서 Charlie Miller와 Charlie Miller는 실제 상용 차량인 Jeep Cherokee의 통신 ECU와 차량 내부 네트워크의 취약점으로 원격에서 차량의 제어권을 획득하는데 성공했다. 그리고 After Market의 정비기기를 이용한 차량 해킹 위협도 존재한다. 이렇게 차량 대상의 보안위협은 꾸준히 증가하고 있으며, 차량설계 시 사이버 보안에 대한 고려가 필요함을 나타내고 있다. 이러한 사이버보안 위협에 대응하기 위해 차량의 사이버 보안 가이드에 대한 연구는 진행되고 있다.

본 논문에서는 차량에서 보호해야 하는 사이버 정보의 자산(Asset)중 내부 통신 프로토콜과 외부통신 방법을 알아보고 차량을 사이버 위협에 보호하기 위한 사이버 보안 가이드 동향에 대해 확인하고자 한다.

1) 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. B0717-16-0097.자율주행차량을 위한 V2X 서비스 통합 보안 기술 개발)

II. 배경지식

사이버 보안 위협은 해당 시스템 취약점과 네트워크의 취약점을 통해 이루어진다. 따라서 보호해야할 사이버 보안 가이드를 작성하기 위해서는 사이버환경에서 지켜야할 자산을 분류하여야 한다. 본 장에서는 차량의 ECU를 위한 내부 통신 프로토콜과 외부 네트워크 기술인 V2X(Vehicle to Everything Communication)에 대해 알아보고자 한다.

2.1 차량 내부 통신 프로토콜

차량은 100개 이상의 다양한 ECU를 이용해 운전자를 위한 편의 기능을 제공한다. 이 ECU를 효율적으로 통신하기 위한 통신 프로토콜은 필요하다. 차량 내부 네트워크는 CAN(Controller Area Network), CAN FD, FlexRay, LIN, MOST 등 다양한 내부 네트워크 프로토콜이 존재한다. 그 중 CAN(Controller Area Network)은 ECU를 제어하기 위해 가장 많이 사용하는 통신 프로토콜 업계표준이다. CAN은 송신자 ID기반의 브로드케스팅 통신 방식으로 차량 내부 ECU들 사이의 통신 회선 복잡성과 길이를 감소하였다. 이러한 장점과 통신에 대한 고신뢰성 때문에 차량 업계에서는 차량 내부 네트워크의 대부분을 CAN으로 사용하기 시작했다.

과거 CAN이 업계에서 사용되기 시작하고 지금까지 차량에는 과거보다 더 많은 ECU가 사용되고 있다. 이러한 ECU의 증가는 통신회선의 높은 데이터 트래픽 처리 환경을 요구하게 된다. 현재 CAN의 통신 전송속도는 이러한 요구사항에 부족하게 되었다. 이러한 문제를 해결하기 위해 CAN보다 높은 데이터 대역폭과 통신 속도를 제공하는 CAN FD가 등장하게 되었다. CAN FD는 64byte의 통신 대역폭과 10Mbit/s의 통신 속도를 제공하고 있다. CAN FD는 CAN과 유사한 데이터 프레임 구조를 가져 오류검출, 충돌 중재, 메시지 우선순위 같은 특징을 유지하고 있다.

그리고 FlexRay는 오류의 허용성 및 시간 결정성 기능 조건 제공과 고속을 지원하는 버스 통신 프로토콜이다. FlexRay는 빠르고 신뢰성 있게 데이터를 송수신하게 설계되었으며, 최대 254byte의 대역폭과 이중화된 10Mbit/s의 통신속도를 제공한다.CAN, CAN FD, FlexRay의 데이터 프레임 구조는 [그림 1]과 같다..

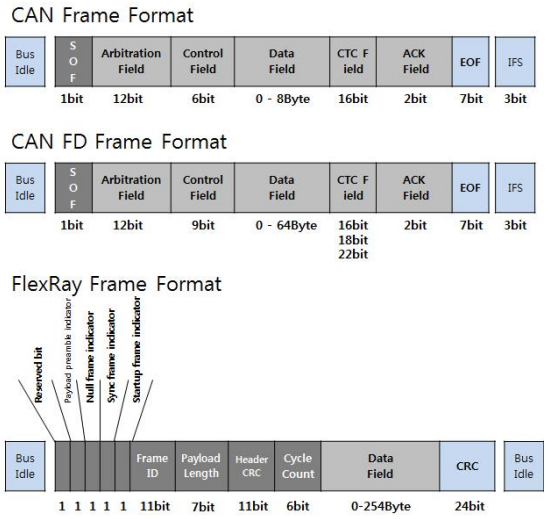


그림 1. CAN, CAN FD, FlexRay frame format 비교

2.2 차량 외부 통신(V2X)

차량은 IT와 융합되면서 외부 네트워크와 연결이 가능해졌다. 그리고 이러한 외부 네트워크를 통한 상용서비스는 GM의 Onstar, BMW의 Connected Drive, Merceds-Benz의 mbrace2 등이 있다. 이러한 통신은 셀룰러 방식으로 통신하면서 사용자에게 운행에 필요한 정보를 제공하게 된다. 그리고 최근 자율주행 기술이 단계적으로 상용화 되고 자율주행 기능을 보조하기 위한 차량 근거리 통신인 DSRC가 보급되고 있다.

자율주행 기능을 보조하기 위해서는 차량 주변의 다양한 정보 수집이 중요하다. 이를 위해 DSRC를 이용한 주변정보를 수집하게 될 것이며, 차량 외부 통신은 차량에 필수적인 기능으로 자리잡게 될 것이다. 현재 V2X를 위한 DSRC는 1초당 10건 이상의 메시지를 생성해 주변의 차량에 브로드케스팅하면서 정보를 전달/수집을 한다. 하지만 V2X의 통신 신뢰성이 보장되지 않으면 해킹에 대한 위협에 쉽게 노출되게 될 것이다. 따라서 이에 대한 사이버 보

안 가이드에 관한 연구가 진행 되어야 한다.

III. 차량 사이버 보안 가이드

사이버 보안 위협에 대한 분석을 위해서는 해당 도메인에서 지키기 위한 정보의 자산(Asset)을 정의하고 해당 환경에서 정보의 흐름 파악이 선행 되어야 한다. 그리고 위협분석이 선행되고 위협에 대한 대응책이 마련될 수 있다.

본 장에서는 해외에서 연구중인 차량 사이버 보안 가이드에 대해 알아보고자 한다.

3.1 IPA 사이버 가이드

IPA(Information Technology Promotion Agency)에서 2013년 9월에 "Approaches for Information Security"를 발표했다. 해당 가이드에서는 차량 시스템의 Security Design을 향상시키기 위해 차량의 사이버 보안위협과 그에 대한 조치를 제시하였다. 그리고 Life Cycle(설계, 개발, 운영, 폐기)에서 사이버 보안을 위해 개발자가 가져야할 주의사항에 대해 설명되어 있다.

본 가이드에서는 차량에서 보안을 위해서는 차량뿐만 아니라 차량에 장착가능한 통신장비와 통신장비에서 제공되는 서비스에 대한 이해도가 필수라 설명했다. [그림 2]는 차량시스템에서 사용되는 장비와 서비스의 전반을 나타내고 있다.

차량의 경우 제조사별 기능과 내부 구조가 서로 다르기에 공통적인 차량 모델을 정의하기 어렵다. [그림 2]는 본 가이드에서 제시한 공통적인 차량 모델을 설정하여 기능을 분류하고 각 기능의 보안에 대한 분석이 진행된다.

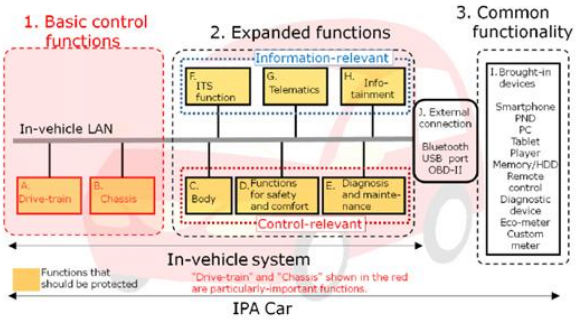


그림 2. IPA Vehicle System Model

기능에 대해 분류가 이루어지고 보안위협에 대한 분류를 "공격자에 의한 위협"과 사용자의 부주의에 의한 위협"으로 나누었다.

3.2 ENSIA 사이버 가이드

ENISA에서는 2016년에 차량을 위한 사이버 보안 가이드인 "Cyber Security and Resilience of smart Car"를 발표했다. 해당 가이드는 스마트카를 대상으로 자산분석, 위협/위험 분석 등을 수행했다. 이러한 위협/위협으로부터 차량을 보호하기 위한 보안기법에 대해 설명하고 있다. 본 가이드에서를 작성되는 과정은 6단계이며 그 단계는 아래와 같다.

- 보안과 관련된 기술 검토가 가능한 전문가를 선별하여 모집하는 단계
- 관련 자료(기술문서, 논문 등) 수집과 엔지니어와의 인터뷰를 통한 자산 및 사례 분석 단계
- 업계 종사자의 의견 수렴 단계
- 차량의 사이버 보안에 대한 기술 분석 수행 단계
- 분석된 내용을 바탕으로 한 권고 사항 작성 단계
- 위의 단계에 참여한 모든 관계자에게 모든 단계에서 도출된 정보 공유 및 검토 수행 단계

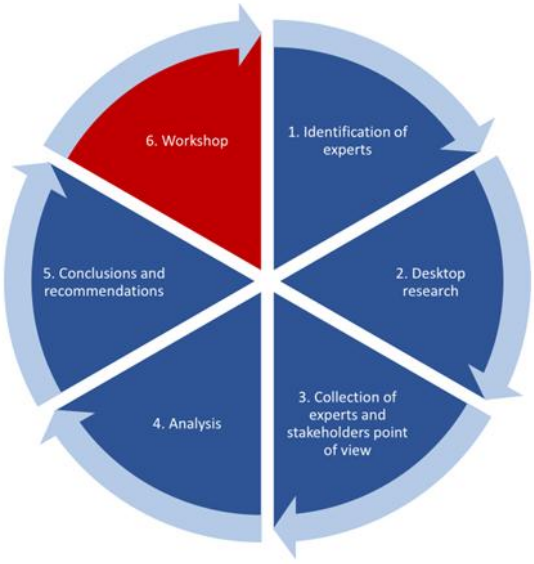


그림 3. 분석 프로세스

IV. 결론

차량의 기술 발전으로 차량은 더 이상 이동수단이 아닌 새로운 IoT기기로 자리 잡게 되었다. 차량 대상의 사이버 침해사고 발생 시 재산상의 피해로 끝나는 것이 아닌 인명피해로 발전할 수 있기 때문에 차량 개발에 있어 사이버 보안 가이드의 필요성이 높아지고 있다. 새롭게 등장하는 기술인 만큼 설계와 구현 단계부터 사이버 보안을 위한 가이드가 정확하게 제시되어야 한다. 하지만 아직 차량을 위한 사이버 보안 가이드는 일반적인 IT환경만 고려되어 작성되었기에 실제 차량에 적용이 한계가 있다.

사이버 보안 가이드는 IT전문가들에 의해서 작성되기 때문에 차량산업과 생태계에 대한 이해가 부족하며 차량산업 종사자의 경우 IT, 정보보안에 대한 이해가 부족하기 때문에 모든 이해관계자가 모여 협의를 계속 진행해나가야 한다. 향후 차량의 모든 정보자산을 분류하고 발생가능한 취약점과 이를 위한 보안대책을 제시한 차량 사이버 보안 가이드에 대한 연구가 진행 되어야 할 것이다.

[참고문헌]

[1] Karl Koscher, Experimental Security Analysis of a Modern Automobile, 2010 IEEE Symposium on Security and Privacy

[2] BOSCH CAN, www.can.bosch.com, 2004

[3] C. Miller, and C. Valasek, "Demo: Adventures in Automotive Networks and Control Units," in DEFCON 2013, 2013

[4] Approaches for Vehicle Information Security(Information Security for "Networked" Vehicles), IPA(Technology Promotion Agency), 2013

[5] Approaches for Vehicle Information Security(Information Security for "Networked" Vehicles), IPA(Technology Promotion Agency), 2013

[6] Cyber Security and Resilience of smart cars (Good practices and recommendations), ENISA, 2016

CAN 트레이스 정적분석을 통한 데이터 영역의 카운터 필드 식별 방법 연구

김병남*, 지청민**, 홍만표*

*아주대학교 사이버보안학과, **아주대 컴퓨터공학과

Method for Identifying Counter Field from Static CAN Trace

Byungnam Kim*, Cheongmin Ji**, Manpyo Hong*

*Dept. of Cyber Security, Ajou University.

**Dept. of Computer Engineering, Ajou University

요 약

수 년간 차량에 대한 공격 사례 및 연구가 활발히 진행되어 왔다. CAN 기반 내부 통신의 보안 취약성이 드러나면서 이에 따른 보안 기술에 대한 연구도 진행되고 있다. 그러나 자동차 내부 통신의 상위 계층 프로토콜은 표준이 없어 모델마다 다르다. 이는 차량 내부 통신 분석 및 차량을 대상으로 한 블랙박스 테스트를 위한 연구에 큰 어려움을 남긴다. 이러한 어려움을 극복하기 위해 데이터 영역의 필드 추론에 관한 연구가 진행되어 왔다. 하지만 데이터 영역의 명백하게 존재하는 카운터를 고려하지 않아 충분한 해결책이 되지 않는다. 본 논문에서는 이러한 문제를 완화하기 위하여 CAN 트레이스의 정적 분석을 통하여 데이터 필드내의 체크썸 기법의 하나인 카운터 필드를 식별 및 추론하는 방법을 제안한다. CAN 버스는 기존의 프로토콜 역공학 기술에 알맞지 않는 구조를 갖는다. 본 논문에서는 CAN 버스의 특성을 이용한 방법을 제안하며 실제 차량의 CAN 트레이스를 대상으로 검증된다.

Ⅰ. 서론

차량 내부 통신을 위한 프로토콜로는 CAN, CAN-FD, Flexlay 등 여러 프로토콜이 채용되고 있다. 현재 대부분의 차량에서는 1986년 Bosch사가 개발한 CAN(Controller Area Network)이 주를 이룬다. CAN 프로토콜 특성상 보안 기능이 충분하지 않아 관련하여 공격을 성공시킨 사례 및 연구가 이어지고 있다.[1-4]. 이에 대한 CAN 기반 차량내부통신에 대한 연구가 진행되어야 할 필요가 있다.

CAN 기반 차량내부통신에서 데이터 링크 계층 프로토콜인 CAN은 명세가 모두 알려져 있다. 하지만 상위 계층인 데이터 영역의 명세는 표준이 따로 존재하지 않고 차량마다 상이하며 모델 별 명세는 공개되지 않는게 일반적이다. 데이터 영역에는 체크썸을 두어 패킷에 대한 무결성 및 인증을 한다. ECU에서 수신한 패킷내의 체크썸이 틀리다면 ECU는 해당 패킷을 에러로 처리한다. 실제 공격시에 체크썸을 파악하여 우회하는 기술은 중요하게 여겨진다[4]. 기존에 진행된 데이터 영역 필드 추론 연구[5,6]에서는 체크썸을 고려하지 않아 CAN 기반의 보안 기술 연구에 어려움이 따른다. 블랙박스 퍼징 테스트를 통해 차량의 내부통신의 보안수준을 객관적으로 평가하고 취약점을 탐지하거나 트레이스를 통한 차량 이상을 탐지하는 등의 보안 연구는 한계를 가지게 된다.

본 논문에서는 이러한 문제를 완화하기 위하여 CAN 트레이스의 정적 분석을 통해 데이터 영역내의 체크썸의 하나인 카운터를 식별하고 추론하는 방법을 제안한다. 제안하는 방법은 크게 두 단계로 구성된다. 첫 번째 단계에서는 CAN 트레이스를 메시지 ID별로 클러스터를 구분하고 비트 당 하나의 필드로 구분하고 각각의 필드의 카운터를 식별한다. 두 번째 단계에서는 식별된 비트별 필드의 카운터를 합쳐 최종 카운터 필드를 도출해낸다.

Ⅱ. 카운터 필드 식별 방법

본 장에서는 비트별 필드의 카운터 식별 및 각각의 비트별 필드를 합쳐 최종 카운터 필드를 도출하는 방법론에 대해 서술한다.

2.1 카운터

본 논문에서는 변수 중 데이터 내용과는 독립적으로 일정한 규칙을 가지는 체크썸을 카운터라 정의한다. 단순히 1씩 증가하거나 감소하는 것뿐만 아니라 패킷 순서에 따른 모든 규칙을 갖는 필드로 정의한다. 그림 1은 CAN 데이터 영역의 존재하는 카운터의 몇 가지 예시이다. 일정한 크기로 지속적으로 감소하거나 증가하는 규칙을 증감 규칙이라 정의한다. 하나의 값이 나오고 그 다음값이 정해져 있는 규칙

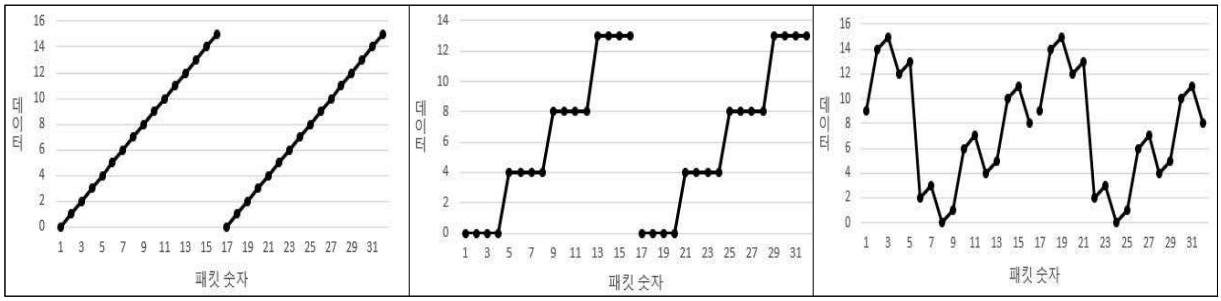
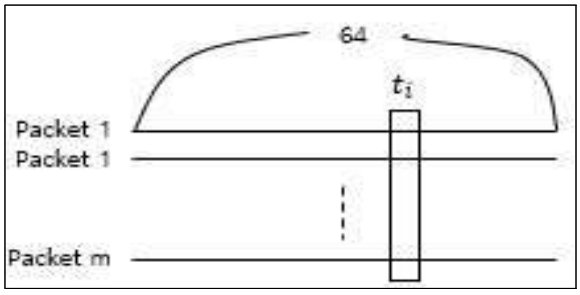


그림 1 증감 규칙, 계단 규칙, 연속 규칙(왼쪽부터)

을 연속 규칙이라 정의한다. 하나의 값이 일정한 횟수를 반복한 후 다음 값이 나오는 규칙을 계단 규칙이라 정의한다.

2.2.1 비트별 카운터 식별

본 절에서는 각각의 메시지 ID별 클러스터에서 비트별 정적분석을 통하여 카운터를 식별하는 방법을 서술한다. 본 절에서 제안하는 방법은 크게 3가지 과정으로 구성된다. 첫 번째 단계에서는 클러스터내에서 각각의 데이터를 비트별로 쪼개어 하나의 필드로 간주한다. 그림 2는 i번째 위치한 비트 필드t_i를 트레이스내에서 도식화한 것이다. CAN 프로토콜의 데이터 영역의 최대 크기는 8바이트이며 최대 64개의 비트별 필드가 생성될 수 있다. 두 번째 단계에서는 트레이스에 연속되는 필드내의 값들을 순차적으로 이어붙여 하나의 문자열을 생성한다. 생서된 문자열은 0과 1로만으로 이뤄진 하나의 비트열이 된다. 세 번째 단계에서는 각 필드의 비트열로부터 카운터 검증 과정을 거친다. 본 논문에서는 2가지 방식을 제안한다. 2.2.2절에서는 무차별 대입 방식에 대해 서술하고 2.2.3절에서는 전이빈도수 비교 방식을 서술한다.



[그림 2] i번째 비트 필드t_i

```
procedure Brute_Force
  for i in [2, bit_stream_length/2] do
    pattern_length <- pattern_length + 1
    pattern <- bit_stream[:pattern_length]
    for split_bit_stream in bit_stream[size:pattern_length]
      if compare pattern, split_bit_stream is false:
        break
      if end_of_stream: all mathced
        return pattern_length
end procedure
```

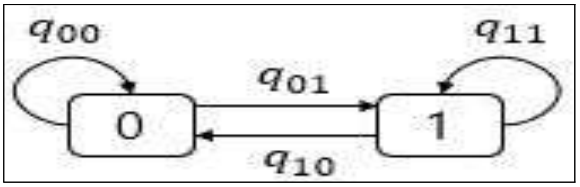
[그림 3] 무차별 대입 방식 pseudo 코드

2.2.2 무차별 대입 방식

본 절에서 서술하는 무차별 대입 방식은 반복되는 규칙 길이를 2부터 하나씩 증가시키며 규칙을 찾는 방식이다. 그림 3은 무차별 대입 방식을 이용하여 카운터를 검증하는 pseudo 코드이다. 비트열을 규칙 길이만큼 앞에서 잘라내어 비교할 새로운 비트열을 만든다. 전체 비트열에서 새로운 비트열을 규칙 길이만큼 순차적으로 비교를 한다. 전체 비트열이 일치한다면 카운터 검증에 통과한 것으로 해당하는 규칙 길이와 생성된 비트열을 규칙으로 도출한다.

2.2.3 전이빈도수 비교 방식

본 논문에서는 비트열내에서 하나의 값 이후 다음 값이 나오는 경우의 수를 전이라 한다. 비트별 필드에서 생성된 비트열의 전이의 모든 경우는 그림 4와 같이 4가지이다. 비트열 내에서 전이가 등장하는 횟수를 전이빈도수라 한다. 만약 비트열이 '001101'이라면 전이빈도수는 $q_{00}=1$, $q_{01}=2$, $q_{10}=1$, $q_{11}=1$이다. 비트별 필드의 전이빈도수를 $Q_i(i=00,01,10,11)$이라 하고 해당 필드가 카운터 필드이며 규칙이 0으로 시작하여 1로 끝난다고 가정하면 그림 5와 같은 상관관계를 갖는다. k는 규치의 길이를 의미하며 m은 패킷의 개수이다. $q_i(i=00,01,10,11)$는 규칙이 갖는 전이빈도수를 의미하며 $q'_i(i=00,01,10,11)$은 필드의 전체 비트



[그림 4] 전이빈도수

$$Q_{00} = \left\lceil \frac{m}{k} \right\rceil q_{00} + q'_{00}$$
$$Q_{01} = \left\lceil \frac{m}{k} \right\rceil q_{01} + q'_{01}$$
$$Q_{11} = \left\lceil \frac{m}{k} \right\rceil q_{11} + q'_{11}$$
$$Q_{10} = \left\lceil \frac{m}{k} \right\rceil q_{10} + q'_{10} + \left\lceil \frac{m}{k} \right\rceil$$

[그림 5] 전이빈도수 상관관계

```
procedure Transition_Frequency
 bit_stream_TF = # of '00','01','10','11' in bit_stream
 for i in [2,bit_stream_length/2] do
   pattern_length <- pattern_length +1
   pattern <- bit_stream[:pattern_length]
   pattern_TF = # of '00','01','10','11' in pattern
   if compare bit_stream_TF, pattern_TF is true:
     if compare pattern, bit_stream[-pattern_length:]:
       return pattern_length
```

[그림 6] 전이빈도수 비교 방식 pseudo 코드

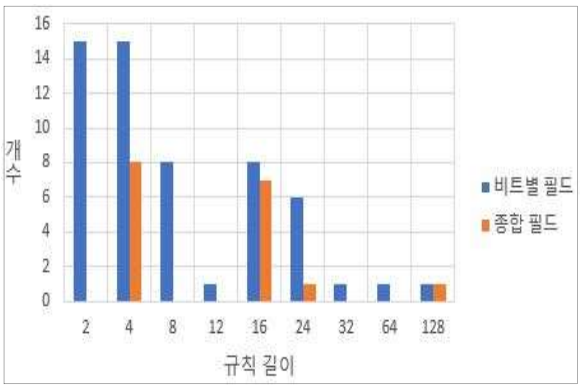
열이 규칙 길이의 배수가 아닐 경우 규칙 길이로 분리하였을 때 남는 마지막 비트열이 갖는 전이빈도수이다. Q_{10}의 상관관계만 다른 형태의 관계가 형성된 이유는 규칙간의 전이빈도수를 채워주기 위함이다. 전이빈도수 비교 방식은 규칙 길이를 2부터 시작하여 늘려가면서 새로운 비트열을 생성하고 생성된 비트열내의 전이빈도수를 계산 및 전체 비트열의 전이빈도수와의 상관관계를 갖는지 확인하여 카운터 필드임을 식별하며 해당 비트열 및 비트열의 길이를 규칙과 규칙 길이로 도출한다. 전이빈도수 비교 방식의 psudo 코드는 그림 6과 같다.

2.3 최종 카운터 필드 도출

본 절에서는 비트필드에서 도출된 카운터 필드들을 합쳐서 최종 카운터 필드를 도출해내는 방법론에 대하여 서술한다. 통상적으로 대부분의 차량에서 비트단위로 필드를 구성하지 않는다. 이러한 점을 고려하여 연관성 있는 비트필드들을 종합하여 하나의 큰 필드로 구성한다. 서로 인접한 카운터 필드는 두 규칙 길이의 최소공배수를 규칙길이로 한 하나의 카운터 필드로 합쳐진다. 그리고 카운터 필드로 둘러싸인 상수필드도 카운터 필드로 합쳐진다.

Ⅲ. 실험결과

본 장에서는 2장에서 서술한 카운터 필드 식별 및 추론 방법을 실제 CAN 트레이스를 대상으로 적용한 결과를 제시한다. Python 언어로 구현하여 실험을 진행한다. Intel(R) Xeon(R) CPU E3-1220 v5 3.00GHz 프로세서, 16GB 메모리 사양의 Ubuntu 16.04.2 운영체제 환경에서 수행된다. 그림 7, 8, 9는 현대 아반떼(Hyundai Avante, 2012년식) 차량의 실제 CAN 버스 트레이스(6.32MB)를 대상으로 본 논문의 제안 방법을 적용하여 도출한 실험 결과를 그래프로 나타낸 것이다. 카운터의 규칙길이의 최대가 128이며 4, 2가 제일 많이 있는 것을 확인할 수 있다. 증감유형이 압도적으로 많이 있지만 나머지 두 형식의 카운터도 적지 않게 존재하는 것을 확인 할 수 있다. 카운터 필드의 사이즈가 1 byte를 넘지 않으며 4 bit 이하의 사이즈가 압도적으로 많은 것을 볼 수 있다. 비트별 필드로 존재하는 카운터 필드는 없이 최종 카운터 필드 도출 과정에서 2-bit 이상의 사이즈의 카운터 필드만 있는 것을 확인할 수 있다.



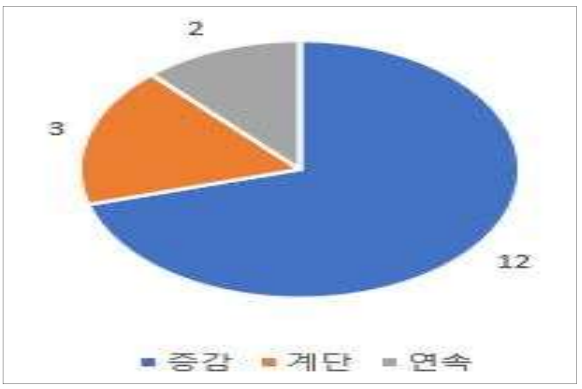
[그림 7] 규칙길이별 개수

Ⅳ. 결론

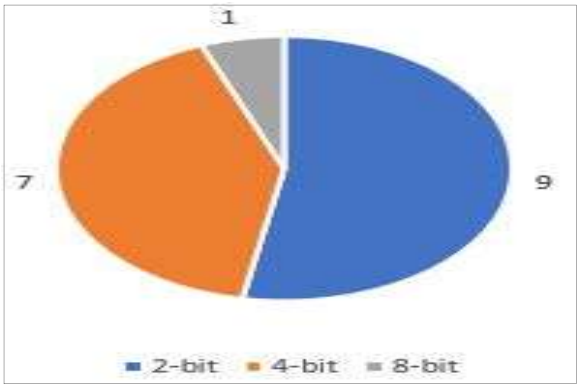
본 논문에서는 CAN 기반 차량내부통신에서 CAN 상위 계층 프로토콜인 데이터 영역내의 카운터 필드를 식별 및 도출하기 위한 방법을 제안하였다. 제안한 방법은 트레이스만을 가지고 정적 분석을 토대로 카운터 필드를 추론한다. 기존 논문[5,6]에서는 변수 및 센서와 카운터 필드를 구분짓지 못한 점을 보완하여 실험시 17개의 카운터 필드를 실제 차량 트레이스에서 추론했다.

CAN 프로토콜의 상위 계층인 데이터 영역의 추론 결과는 다양한 기술에 활용될 수 있다. CAN 트레이스 분석을 통해 차량의 내부통신 안전성을 평가하기 위한 방법으로 블랙박스 퍼징를 활용할 경우 카운터 필드 식별을 통해 더 깊숙한 상태의 퍼징 테스트가 가능할 것으로 기대된다. 추가적으로, 펌웨어 리버싱 기술 없이 카운터 필드를 통과할 수 있는 방법을 찾는 것은 기존의 차량대상 침투 테스트에 어려움을 낮춰 줄 것이라 기대된다. CAN 트레이스 정적분석을 통해 이상 패킷을 탐지하는 기술[6]의 경우 카운터/센서 필드를 세분화할 수 있기 때문에 더 정확한 필드 추론 결과를 도출해내어 오탐율을 낮출 수 있을 것으로 기대된다.

본 논문에서 제안하는 방식은 기존 논문에서 정의하는 필드 종류에 카운터 필드를 추가한다. 실제 CAN 버스의 데이터 영역에는 데이터의 값에 의존적인 체크썸이 존재한다. 차후에는 이런 체크썸까지 포함하며 더 다양한 데이터 모델이 필요할 것으로 생각된다.



[그림 8] 카운터 유형별 개수



[그림 9] 필드크기별 개수

[Acknowledgement]

This research was supported by the Korea Ministry of Land, Infrastructure and Transport. It was also supported by the Korea Agency for Infrastructure Technology Advancement(Project No.: 18TLRP-B117133-03)

본 연구는 국토교통부 및 국토교통과학기술진흥원의 연구비지원(18TLRP-B117133-03)으로 수행된 연구임.

[참고문헌]

[1] Experimental Security Assessment of BMW Cars: A Summary Report, KEEN security lab, May 22, 2018
[2] Experimental Security Analysis of a Modern Automobile, Karl Koscher, Alexei Czeskis, et al, Department of Computer Science and Engineering University of Washington, IEEE Symposium on Security and Privacy, May 16, 2010
[3] FREE-FALL: TESLA HACKING 2016, KEEN security lab, Sep 19, 2016
[4] CAN Message Injection, Charlie Miller, Chris Valasek, June 28, 2016
[5] 지청민, 김지민, 홍만표, CAN 트레이스 분석을 통한 데이터 필드 형식 추론 방법 연구, 아주대학교 컴퓨터공학과, 아주대학교 사이버보안학과 , Feb, 2017.
[6] M. Markovitz and A. Wool, Field classification, modeling and anomaly detection in unknown CAN bus networks, Vehicular Communications, vol, Jul. 2017

스마트 밴드와 스마트 워치의 디지털 포렌식 사례연구※

류지은*, 김수빈*, 김은주*, 장혜지*, 최지원* 김종성*,**

*국민대학교 정보보안암호수학과 **국민대학교 금융정보보안학과

The Need for Digital Forensics about Smart Band and Smart Watch

Jieun Ryu*, Subin Kim*, Eunju Kim*, Hyeji Jang*, Jiwon Choi*, Jongsung Kim*,**

*Dept. of Information Security, Cryptology and Mathematics, Kookmin University

**Dept. Financial Information Security, Kookmin University

요 약

IoT(Internet of Things) 시대가 가속화되면서 소형 웨어러블 기기의 사용률이 증가하고 있다. 웨어러블 기기가 기록하는 사용자 데이터를 다양한 사건·사고의 조사 및 해결에 활용하면서, 이에 대한 디지털 포렌식의 필요성이 대두되고 있다. 본 논문에서는 웨어러블 기기 중 사용자 이용량이 높은 스마트 밴드와 스마트 워치를 대상으로 데이터 추출 방법과 포렌식 관점에서 중요하게 사용될 데이터를 정리한다. 또한, 스마트 밴드와 스마트 워치를 통해 해결한 사건·사고 사례에 기반을 둬 웨어러블 기기에 대한 디지털 포렌식의 중요성에 대해 언급한다.

I. 서론

IoT 시대가 가속화되면서 많은 사람이 스마트 밴드나 스마트 워치와 같은 소형 웨어러블 기기를 사용하게 되었다.

[1]에 의하면 웨어러블 등 착용형 스마트기기 시장은 점차 증가하고 있고 앞으로도 계속 증가할 것임을 알 수 있다. 이 중 스마트 밴드와 스마트 워치는 웨어러블 기기의 대표격으로 2017년 기준 IoT 사용자의 90% 정도가 스마트 밴드 혹은 스마트 워치를 사용함을 알 수 있다.

스마트 밴드의 국내 시장 점유율은 샤오미가 1위를 차지했다, 이어서 삼성전자, 핏빗, 코아, 소니, 기타 순이다. 해외에서는 마니아층이 있는 애플을 제외하면 저가 스마트밴드가 두각을 보인다. 특히 '샤오미'는 2017년 기준으로 점유율 68%를 차지했다[2].

스마트 워치의 국내 시장 점유율은 삼성전자가 1위, 그 뒤를 이어서 애플과 LG전자, 소니 순이며, 연도별 순위변동은 없다. 국내와 달리 해외 스마트 워치 시장 점유율의 경우 2015년 1분기부터 18년 1분기까지 종합하면, 애플과 삼성전자가 높은 비율을 차지한다[3].

스마트 밴드와 스마트 워치는 사용자의 흔적을 여러 방면에서 기록하며, 기록된 자료는 다양한 사건·사고의 조사 및 해결에 많은 도움이 되고 있다. 이에 따라 웨어러블 기기에 대한 디지털 포렌식의 필요성이 대화 주제로 부상하고 있다. 본 논문에서는 스마트 밴드와 스마트 워치에 대한 디지털 포렌식의 필요성에 대해 다룬다. 2장과 3장에서는 스마트 워치에 대한 데이터 추출 방안 및 추출 가능한 데이터의 정보를 다룬다. 마지막으로, 4장에서는 사례를 바탕으로 스마트 밴드 및 스마트 워치에 대한 디지털 포렌식의 필요성에 대해 언급한다.

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 작성되었습니다. (No.2017-0-00344, 최신 모바일 기기에 대한 암호해독 및 포렌식 분석)

II. 스마트 밴드

본 장에서는 피트니스 트래커로 알려진 스마트 밴드에 대한 데이터 추출 방안 및 추출 가능한 데이터의 정보를 다룬다.

스마트 밴드에는 데이터 추출 과정은 핏빗 블레이즈 (Fitbit Blaze)로 설명한다. 핏빗 블레이즈는 트래커가 수집하는 데이터를 Fitbit 대시보드와 동기화시켜 사용할 수 있도록 지원한다. 대시보드에서 데이터 분석, 생체량 기록 등을 통해 이용자의 건강 생활에 도움을 준다. 데이터 추출 과정은 다음과 같다[3][4].

1) 앱 설정 단계

2) 기한 설정 및 데이터 타입 선택 단계

3) 다운로드 단계

앱 설정 단계에서는 Windows 10의 환경에서 작동하는 Fitbit Windows Application을 이용한다. 기어 아이콘을 누르고 설정에서 Data Export를 선택한다.

기한 설정 및 데이터 타입 선택 단계에서는 데이터를 추출하려는 기간과 형식을 선택한다.

다운로드 단계는 다운로드 버튼을 통해 데이터를 추출한다. 추출된 데이터는 CVS 파일로 저장된다.

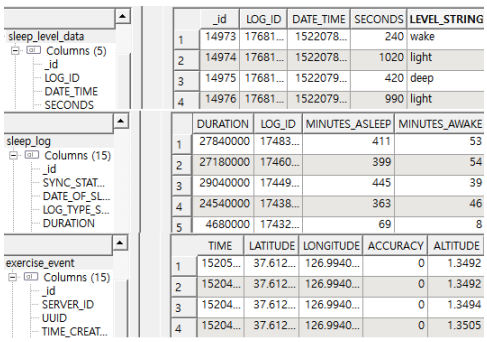
스마트 밴드의 데이터 추출 과정을 통해 추출되는 데이터는 다음과 같다[표 2-1][5][6].

[표 2-1] 스마트 밴드에서 추출 가능한 데이터

종류		내용
디바이스		위치정보, 이동거리, 걸음 수, 칼로리 소모량, 활동 시간, 수면 시간
네트워크	Blue tooth	SpecificationID, VendorID, ProductID, VersionPrimaryRecord, VendorRecord, VendorIDSource
플랫폼/서비스		사용자 개인 정보, 개인 SNS의 정보, 방문 페이지 기록, IP주소

[5]에 의하면, [표 2-1]의 관리영역은 수집할 수 있는 정보의 유형을 데이터를 추출하는 환경에 따라 분류한다. 이는 센서가 부착된 디바이스에서 수집할 수 있는 정보, 센서의 정보를 전달하는 네트워크에서 수집될 수 있는 정보, 플랫폼/서비스 단계에서 수집될 수 있는 정보이다.

다음은 추출 과정에 의해 추출된 실제 Fitbit의 데이터 중 일부이다(그림 1).



(그림 1) 추출된 Fitbit 데이터

III. 스마트 워치

본 장에서는 스마트 워치에 대한 데이터 추출 방안 및 추출 가능한 데이터의 정보를 다룬다.

스마트 워치의 데이터 추출 과정은 안드로이드 기반의 웨어러블 커스텀 OS의 운영체제를 가진 삼성 갤럭시 기어2로 설명한다. 갤럭시 기어2는 기존의 시계 역할을 하면서 스마트폰과 연동되어 이용자의 데이터를 수집 및 분석을 하는 등 다양한 기능을 한다. 데이터 추출 과정은 다음과 같다[6][7].

1) 기기의 초기화 단계

2) 장치의 부팅과 연결 단계

3) 이미징(Imaging) 단계

기기의 초기화 단계는 클라이언트와 서버의 데이터를 일치시키기 위해 진행된다. 이를 위해서는 '기어 매니저' 애플리케이션 설치가 요구되며 NFC 또는 Bluetooth를 이용한 동기화 과정이 필요하다.

장치의 부팅과 연결 단계는 갤럭시 기어를 루팅 할 수 있게 하는 단계이다. 이를 위해서는 '삼성 USB 드라이버', '시디아 임팩터[1]', '비지박

스[2]' 애플리케이션의 설치가 요구된다. 루팅은 일반유저의 권한을 최상위 권한인 루트 권한으로 상승시키기 위해 필요하다. 루팅을 함으로써 사용자는 기기에 대한 모든 접근이 가능해지고 이를 통해 데이터 추출이 가능해진다. 이러한 루팅을 하기 위해서는 USB로 갤럭시 기어와 PC를 연결해야하며, 이 과정에서 '시디아 임팩터'와 '비지박스'가 이용된다. 도중 에러 메시지 출력이 없다면 루팅에 성공한 것으로 취급한다.

이미징 단계에는 두 가지 정도의 제한사항이 있다. 우선 Wi-Fi나 3G/4G와 같은 네트워크 인터페이스에서는 이미징을 할 수 없다. 그리고 저장 매체인 갤럭시 기어의 내장 메모리는 4GB로 그 이상의 데이터를 저장하지 못한다. 이러한 제한사항이 허용하는 선에서 데이터를 이미징 할 수 있고 디렉토리에 추출된 이미지를 마운트하고 데이터베이스를 추출할 수 있다. 데이터를 추출하기 위해서는 FTK imager를 이용한다.

스마트 워치의 데이터 추출 과정을 통해 추출되는 데이터들은 다음과 같다[표2-2][4][5].

[표 2-2] 스마트 워치에서 추출 가능한 데이터

종류		내용
디바이스		심박수, 걸음 수, 디바이스 인증정보, 심전도
네트워크	Wi-Fi	SSID(Service Set Identifier), 단말기 정보(모델, 일련번호, USIM번호 등)
	Blue tooth	SpecificationID, VendorID, ProductID, VersionPrimaryRecord, VendorRecord, VendorIDSource
플랫폼/서비스		사용자 정보, 캘린더, 연락처, 환자의 진료기록, 미디어 및 설정

1) 모바일 장치를 사용하기 위한 GUI 도구이며, iOS에 IPA 파일을 설치하고 Android에 APK 파일을 설치할 수 있음.

2) 하나의 실행 파일 안에 스트립 다운된 일부 유닉스 도구들을 제공하는 소프트웨어임.

IV. 사례를 통한 스마트 밴드 및 스마트 워치에 대한 디지털 포렌식의 필요성 및 시장현황

언제 어디에서 무엇을 하는가에 대해 다른 디지털기기보다 빠르고 쉽게 얻어낼 수 있는 스마트 밴드와 스마트 워치에 대한 디지털 포렌식을 통해 증거를 추출하여 형사사건을 해결한 사례들이 있다.

첫 번째 사례는 미국에서 일어난 살인사건이다. 목격자는 사망자가 괴한에게 9시경에 공격당해 살해당했다고 진술했지만, 사망자가 착용했던 스마트 밴드 핏빗에는 10시에 그녀의 움직임이 기록되어 있었다. 이 기록을 근거로 스마트 밴드를 분석하여 증거를 찾아낸 뒤 거짓 증언을 파악해 범죄를 해결하였다[5].

두 번째 사례는 호주에서 일어난 살인사건이다. 용의자는 현장에 있던 피해자이자 목격자였다. 그는 사망자가 운전 중 시비가 붙은 무리에게 20분간 공격을 당해 사망했다고 진술했다. 하지만 사망자가 지니고 있던 애플워치에 기록된 심박 수와 활동 데이터를 분석한 결과, 사망자는 용의자의 진술과 다르게 기습 공격을 당하여 7분 안에 사망한 것으로 밝혀졌다. 결국, 거짓 진술을 한 용의자는 살해 혐의로 기소되었다[7]. 첫 번째 사례와 비슷하게, 스마트 워치를 분석하여 증언이 사실인지 거짓인지를 판단할 수 있다. 또한, 스마트 워치에 기록된 활동량을 조사하여 어떤 공격을 당했는지까지도 알 수 있다.

세 번째 사례는 미국에서 일어난 또 다른 살인사건이다. 사망자의 집 안에서 일어난 사고는 목격자가 없어 난항을 겪었지만, 사고 당시 착용하고 있던 핏빗의 기록을 통해 사건 시각을 파악했다. 사망자의 심박 수는 3시 20분에 급격히 느려졌고, 기기는 8분 뒤 데이터 전송을 중지했다. 이에 따라 해당 시간에 주차되어있던 차량을 조사하여 범인을 체포할 수 있었다[8]. 이 사례를 통해, 스마트 밴드 포렌식 기술이 사건의 목격자가 없는 사건의 해결을 위한 실마

리를 제공할 수 있다는 것을 알 수 있다.

위와 같은 사례들을 토대로 스마트 밴드와 스마트 워치에 대한 디지털 포렌식은 수사에 적극적으로 활용 가능함을 알 수 있다. 스마트 밴드와 스마트 워치는 다른 전자기기보다 장시간 사용자와 가장 가까이에 위치하기 때문에, 보다 연속적인 데이터를 기록한다. 따라서 위와 같이 사건 해결에 도움이 되곤 한다. 또한, 강력사건에서 그 정황을 입증하기 위해서 필수적으로 활용되는 등 중요성이 커지고 있다.

하지만 아직까지 우리나라에서는 스마트폰이나 컴퓨터에 대한 포렌식을 지원하는 기업만 있을 뿐, 아직 스마트 밴드나 스마트 워치에 대해 전문적으로 포렌식을 지원하는 기업이 존재하지 않는다. 해외의 경우 미국을 중심으로 스마트 밴드와 스마트 워치를 연구하고 있으며, BlackBag TECHNOLOGIES와 같은 포렌식 업체는 안드로이드나 iOS 기반 디바이스를 전문으로 포렌식을 지원하고 있다. 따라서 스마트 밴드와 스마트 워치에 대한 전문적인 포렌식 기술을 연구해야 한다.

V. 결론

본 논문에서는 예시를 통해 스마트 밴드와 스마트 워치에 대한 대표적인 데이터 추출 방법 및 추출 데이터들에 대하여 알아보았다. 또한, 실제 사례를 통해 스마트 밴드와 스마트 워치에 대한 디지털 포렌식 기술 가치도 확인했다. 이후로도 스마트 밴드와 스마트 워치의 점유율은 증가할 것으로 보인다. 따라서 이에 대한 디지털 포렌식 연구 역시 발전되어야 한다.

[참고문헌]

[1] 김한, "성장동력, 스마트기기 분야의 정책 및 산업 현황에 대한 고찰", 한국통신학회 학술대회논문집, 1432-1433쪽, 한국통신학회, 2018.

[2] 고현실, "스마트밴드가 달군 웨어러블 시장, 차기 주도권은?", 연합뉴스, 2016. (2018. 10. 29)

[3] '韓 스마트워치 시장점유율 "삼성이 애플 갑절"', 전자신문, 2018. (2018. 08. 06)

[4] "How do I export Fitbit data?", Fitbit Help, 2018. (2018. 10. 22)

[5] Courtney Grimes, "Application Analysis: Fitbit", Champlain college LCDI, 2017.

[6] 김성배, "빅데이터 및 사물인터넷(IoT) 환경에서의 디지털 포렌식 연구", 2018.

[7] Jae-ki Kim, "Digital Evidence from Android-based Smartwatch", F-INSIGHT, 2014.

[8] Sara Alhajaili, "Samsung Gear 2 Smartwatch Forensica", UCD(Dublin), 2014.

[9] AUSTRALIAN ASSOCIATED PRESS, "Apple Watch murder breakthrough: Heart rate recorded by smart device worn by woman, 57, 'bashed to death by her daughter-in-law reveals she was the victim of an ambush attack", MailOnline, 2018. (2018. 10. 25)

[10] Emma Betuel, "Fitbit Data Point to 90-Year-Old Culprit in Grisly Family Murder", KUERSE, 2018 (2018. 10. 22)

메모리 포렌식을 이용한 오토핫키로 컴파일된 바이너리 판별 방안

장지나*, 김동현, 염흥열*
*순천향대학교 정보보호학과

Identification of Compiled AutoHotkey binary codes using Memory Forensics

Ji-na Jang*, Dong-hyun Kim, Heung Youl Youm*
*Division of information security, Soonchunhyang University.

요 약

최근 오토핫키로 제작된 악성코드가 유포되고 있으며, 종류도 다양하다. 오토핫키 악성코드 제작자는 오토핫키 언어로 작성된 스크립트를 컴파일하여 실행 파일 형태로 배포한다. 또한, 제작 과정 중에 패커를 이용해 바이너리 난독화를 적용하는 경우도 있어 바이너리 분석을 더 어렵게 한다. 컴파일된 스크립트 언어의 특성상 디컴파일하여 원본 소스 코드를 획득할 수 있다. 즉, 분석할 바이너리가 스크립트 언어가 컴파일된 것을 알 수 있다면 디컴파일하여 쉽게 분석할 수 있다. 바이너리가 난독화되어도 물리 메모리에서 API가 노출된다는 점을 이용하여 본 논문에서는 메모리 포렌식을 활용하여 난독화된 바이너리가 오토핫키로 제작되었는지 판단한다.

I. 서론

오토핫키(AutoHotkey)는 사용자가 반복 작업을 자동화하는 매크로 프로그램 작성에 특화된 스크립트 언어다. 다양한 Windows API를 직관적인 문법으로 간단하게 사용할 수 있다는 것이 특징이다. 이러한 특징으로 프로그래밍을 전문으로 하지 않는 사람도 쉽고 간결하게 사용할 수 있다. 또한, 오토핫키는 컴파일하여 EXE, DLL과 같은 실행 파일 형태로 범용적으로 배포할 수 있으며, 컴파일된 실행 파일은 C++ 기반으로 작성된다.

오토핫키의 간편하지만 강력한 기능으로 악성코드를 제작하여 악의적인 목적으로 사용한 사례들이 있다. 이러한 형태는 과거부터 존재했으며, 최근에는 안티 바이러스 제품 위장, 키로거, 암호화폐 채굴기 등 다양한 종류와 형태로 유포되고 있다.

실제로 2018년 3월에 카스퍼스키 안티바이러스 제품을 위장한 오토핫키 악성코드가 유포된 사례가 있다. USB를 통해 전파되며 사용자의 입력을 키로깅하여 정보를 탈취한다, 또한, 2018년 4월 레타덥(RETADUP)이라는 암호화폐를 채굴하는 웜이 발견된 사례도 있다.

기본적으로 포함되어 있는 오토핫키 컴파일러에는 변환된 바이너리의 크기를 축소하고 원본 코드를 보호하기 위해 UPX 또는 MPRESS 패커를 적용할 수 있는 옵션이 존재한다. 이러한 방법으로 악성코드 제작자가 패킹을 한다면, 분석가들은 오토핫키로 컴파일된 바이너리 분석에 어려움을 겪게 된다. 그러나, 컴파일된 스크립트 언어 특성상 디컴파일하여 원본 소스 코드로 되돌릴 수 있다. 이러한 이유로 분석가가 해당 바이너리가 오토핫키로 컴파일되었다는 사실을 알 수 있다면, 쉽게 분석할 수 있다.

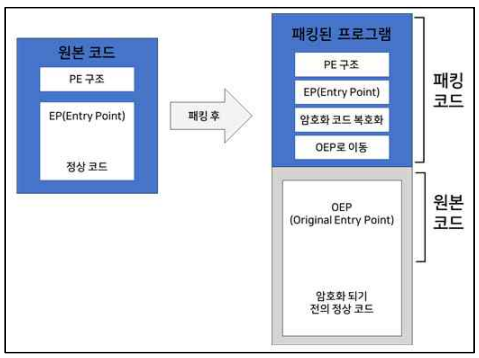
본 논문은 메모리 포렌식을 활용해 난독화된 바이너리가 오토핫키로 제작되었는지 판단한다.

II. 본론

2.1 바이너리 난독화

바이너리 난독화는 안티 디버깅, 코드 가상화, 패킹 등을 사용하여 분석가들이 역공학하는 것을 어렵게 만드는 코드 보호 기법이다. 오토핫키에서 제공하는 UPX와 MPRESS 또한 패킹의 종류 중 하나이다.

패킹은 실행 파일을 압축하는 기술이다. 프로그램에 대한 코드를 보호하기 위한 목적인 Protector와 파일의 크기를 줄여 배포를 쉽게 하기 위한 목적인 Compressor가 있다. 패킹은 주요 코드를 인코딩하여 다른 섹션에 저장해두고, 실행 파일 내에 언패킹하는 루틴을 포함한다. 언패킹 루틴을 통해 코드 영역과 데이터 영역을 디코딩하여 압축 해제한다. 위의 과정을 통해 인코딩된 코드를 원본 코드로 디코딩한다. 즉, 메모리에서 디코딩된 원본코드를 얻을 수 있게 된다.



[그림 1] 패킹 원리

2.2 메모리 포렌식

운영체제의 종류가 Linux, OS X, Windows 등에 따라 메모리 관리 방식과 프로세스의 구조가 달라지기 때문에 물리 메모리의 수집 방식과 분석 방법이 달라진다. 오토핫키는 Windows 운영체제에서만 사용할 수 있으므로 본 논문에서는 Windows 운영체제에 대한 메모리 포렌식 내용을 담았다. Windows 운영체제에서의 메모리 획득은 [표 1]과 같은 방식들이 있으며, 각 방식마다 장단점이 존재한다. 메모리 분석은 볼라틸리티(Volatility), 레칼(Rekall), 레드라인(Redline)이라는 도구를 가장 많이 사용하며, 같은 운영체제라도 조금씩 구조가 달라지기 때문에 이를 고려하여 분석 도구를 사용해야 한다.

[표 1] 메모리 획득 방법과 장단점

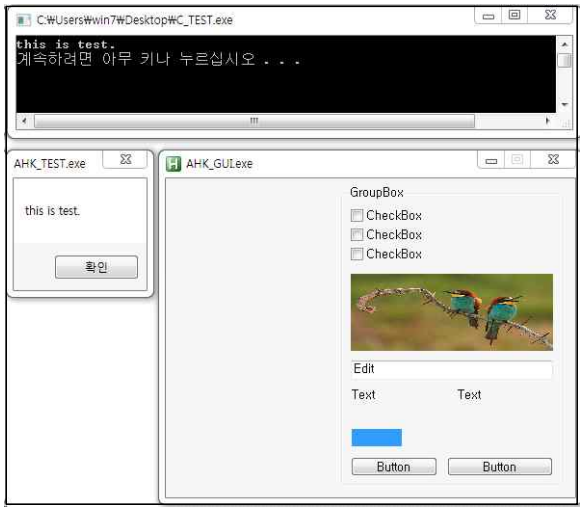
방식	장점	단점
하드웨어	루트킷과 같은 악성코드에 의해서 수집에 방해를 받지 않음	고가의 장비가 필요하며, 블루스크린이 발생할 수도 있음
소프트웨어	FTK Imager, Dumpit 등의 도구를 이용하여 간편하고 안정적으로 메모리 수집 가능	루트킷에 의해 방해받을 수 있으며, 수집된 메모리에 사용한 도구가 기록됨
크래시 덤프	운영체제가 생성하므로, 순도 높은 메모리 수집 가능	재부팅해야 얻을 수 있으며, 메모리의 일부만 수집하도록 기본 설정됨
최대 절전모드	크래시 덤프와 같이 순도 높은 메모리 수집 가능	최대 절전 모드로 진입했을 때만 수집 가능
가상화 도구	스냅샷 기능을 이용하여 간편하게 메모리 수집 가능	VMware와 같은 가상화 도구가 있어야 가능

2.3 분석 결과

분석에 사용된 AutoHotkey는 2018년 8월 22일에 공개된 1.1.30.00 버전이며, 메모리 포렌식에 사용된 도구는 볼라틸리티(Volatility) 2.6 버전이다. 테스트 환경은 Windows 7 Professional K이며, VMware를 이용하여 가상환경을 구성하였다. 물리 메모리는 VMware의 일시중지 기능을 이용하여 수집하였다.

AutoHotkey 1.1.30.00 버전에서 제공하는 "Ahk2Exe.exe"를 이용하여 AutoHotkey 스크립트를 컴파일하였으며, MPRESS 패킹 옵션을 선택하여 컴파일된 바이너리를 보호했다. 분석에 사용될 바이너리는 [그림 2]와 같다. 첫 번째

바이너리 "C_Test.exe"는 "this is test."를 출력하도록 C언어로 작성되어 Visual Studio 2010에서 컴파일하였다. 두 번째 바이너리 "AHK_TEST.exe"는 오토핫키로 작성되어 메시지 박스로 "This is test test."를 출력하며, 세 번째 바이너리 "AHK_GUI.exe"도 오토핫키로 작성되어 체크 박스, 버튼 및 사진 출력 등을 수행하는 비교적 복잡한 코드로 이루어져 있다.



[그림 2] 위 : C_TEST.exe(첫 번째) / 좌 : AHK_Test.exe(두 번째) / 우 : AHK_GUI.exe(세 번째)

각 바이너리들을 실행한 후 메모리를 수집했다. 수집한 메모리를 볼라틸리티의 pslist 명령어를 이용해 실행한 바이너리의 프로세스 정보를 출력했다. 출력된 정보를 정리하면 [표 2]와 같다.

[표 2] 비교할 프로세스의 PID

프로세스 이름	PID
C_TEST.exe	2624
AHK_TEST.exe	2512
AHK_GUI.exe	156

볼라틸리티에서 특정 바이너리에서 호출된 API들의 목록을 출력하는 플러그인을 [그림 3]과 같이 사용하여 출력된 값을 각각 C_TEST_APIs, AHK_TEST_APIs, AHK_GUI_APIs 파일에 저장하였다. 각 파일에 저장된 내용을 정리하여 비교하면 [표 3]과 같다. [표 3]을 통해 Visual Studio 2010에서 C언어로 컴파일된 바이너리에는 MSVCR100.dll을 호출하였지만, 오토핫키로 컴파일된 바이너리는 MSVCR100.dll을 호출하지 않고 많은 DLL을 호출한 것을 알 수 있다. 또한, 오토핫키로 컴파일된 바이너리는 코드의 규모와 상관없이 같은 수의 API와 같은 종류의 DLL을 호출하는 것을 알 수 있다.

```
volatility -f dump --profile=Win7SP1x86 impscan -p 2624 > C_TEST_APIs

volatility -f dump --profile=Win7SP1x86 impscan -p 2512 > AHK_TEST_APIs

volatility -f dump --profile=Win7SP1x86 impscan -p 156 > AHK_GUI_APIs
```

[그림 3] 호출한 API 목록을 출력하는 명령어

[표 3] 각 바이너리 비교

프로세스 이름	호출한 API 개수	호출한 DLL 종류
C_TEST.exe	41	kernel32.dll, ntdll.dll, MSVCR100.dll
AHK_TEST.exe	447	ADVAPI32.dll, COMCTL32.dll, COMDLG32.dll, GDI32.dll, kernel32.dll, ntdll.dll, OLEAUT32.dll, PSAPI.DLL, SHELL32.dll, USER32.dll, VERSION.dll, WINMM.dll, WS2_32.dll, ole32.dll
AHK_GUI.exe		

[그림 3]의 명령어의 결과를 저장한 AHK_TEST_APIs와 AHK_GUI_APIs 파일이 같은지 확인하기 위해 SHA-1 연산을 하였다. 수행한 결과는 [표 4]와 같으며, [표 4]를 통해 오토핫키로 컴파일된 바이너리 모두 같은 API들을 호출하였다는 것을 알 수 있다.

이때까지의 분석 내용을 종합해보면 오토핫키로 컴파일된 바이너리는 코드의 규모에 상관없이 모두 같은 수와 내용의 API를 호출한다는 사실을 알 수 있다. 해당 사실을 통하여 오토핫키로 컴파일된 바이너리 판단할 수 있게 되었으며, 오토핫키로 컴파일된 바이너리를 다시 디컴파일하여 소스코드 단으로 분석할 수 있다.

[표 4] SHA-1 값을 통해 저장된 결과 비교

파일명	SHA-1 해시 값
AHK_TEST_API	92230FF440767B32FC08 38950F4204B108CF9B85
AHK_GUI_APIs	92230FF440767B32FC08 38950F4204B108CF9B85

III. 결론

오토핫키는 간편하지만 강력한 기능을 가진 스크립트 언어이며, 자체적으로 컴파일러를 제공하여 실행 파일 형태로 배포할 수 있도록 한다. 또한, 바이너리의 크기를 줄이고 원본 코드를 보호하기 위해 패커도 제공하고 있다.

이러한 특성으로 악성코드 유포자들은 오토핫키를 이용하여 과거부터 지금까지 키로거, 암호화폐 채굴기 등 다양한 악성코드를 제작하고 배포해왔다.

컴파일된 스크립트 언어는 디컴파일하여 원본 소스 코드를 획득할 수 있는데, 분석가가 오토핫키로 작성된 코드임을 알 수만 있으면 디컴파일하여 쉽게 분석할 수 있다. 하지만, 악성코드 유포자가 기본적으로 제공되는 패커로 바이너리를 보호하고 해당 바이너리에 약간의 값을 수정하면, 분석가는 오토핫키로 작성된 바이너리임을 알기가 어려워져 분석 난이도가 급격하게 증가한다. 이러한 경우에도 오토핫키로 작성된 바이너리임을 쉽게 판단할 수 있는 방법을 본 논문에서 소개하였다. 메모리에서 API들이 노출된다는 점을 이용하여, 해당 바이너리가 호출한 API들의 수와 종류를 특징으로 오토핫키로 컴파일된 바이너리임을 판별하였다.

[참고문헌]

[1] AutoHotkey, "AutoHotkey Documentation", https://www.autohotkey.com/docs/AutoHotkey.htm

[2] 트랜트 마이크로, 2018 중간 위협 보고서

[3] 보안뉴스, "카스퍼스키 백신 제품 흉내 내는 멀웨어, USB 통해 확산" https://www.boannews.com/media/view.asp?idx=68008

[4] 보안뉴스, "변신하는 암호화폐 채굴 레타덥이 주는 교훈" https://www.boannews.com/media/view.asp?idx=68742

[5] 문해은, 성준영, 이현식, 장경익, 곽기용, 우상태, "악성코드 및 패커 탐지를 이용한 공격 그룹 판별", 2018년 정보과학학회 논문지

[6] 양대엽, 정만현, 조재익, 손태식, 문종섭, "메모리 초기화를 이용한 사용자 데이터 유출 방지에 관한 연구", 2012년 7월 전자공학회 논문지 제49권 TC편 제7호

[7] Volatility Foundation, "Volatility wiki", https://github.com/volatilityfoundation/volatility/wiki

[8] 최석우, "난독화 바이너리 분석 방안", 2018년 3월 정보보호학회지 제36권 제3호 26-31 pages

[9] 한승원, 이상진, "악성코드 포렌식을 위한 패킹 파일 탐지에 관한 연구", 2009년 10월 정보처리학회논문지C 제16권 제5호 555-562 pages

안드로이드 애플리케이션 사용 시 저장되는 주요 캐시 데이터 분석에 관한 연구1)

김형찬* 김동현* 손태식**
*아주대학교 사이버보안학과
**아주대학교 컴퓨터공학과

A Study on Analyzing Cache Data Analysis generated by Android Applications

Hyung-Chan Kim*, Dong-Hyun Kim*, Tae-Shik Shon**
*Departments of Cyber Security, Ajou University.
**Departments of Computer Engineering, Ajou University.

요 약

안드로이드에서 애플리케이션을 사용하면 캐시 데이터가 생성된다. 이는 디지털 범죄가 발생했을 때 수사나 분석에 있어 중요한 단서의 역할을 할 수 있다. 하지만 분석가들은 다양한 캐시 아티팩트의 명확한 생성 목적 및 구조 파악 없이 분석을 수행하고 있다. 이로 인해 단순 분석에 의한 부가적인 데이터로 활용되는 경우가 대다수이다. 따라서 본 논문에서는 애플리케이션 사용 시에 저장되는 캐시의 종류, 의미 및 구조를 분석하여 각 캐시에 따른 분석 방안을 제시함으로써 캐시에 대한 활용성이 향상되는 것을 기대한다.

Keywords disk cache, journal, memory cache, Mobile Forensic, Android Forensic

I. 서론

2018년 8월 24일 통신 업계와 정보통신진흥협회 등에 따르면 지난달 말 국내 이동전화 시장의 스마트폰 가입자 수가 약 5천 11만 명으로 집계되었다[1]. 또한 가트너가 매년 분기별로 발표한 세계 모바일 OS 시장 점유율을 확인해 보면 안드로이드 OS의 점유율이 상승하고 있으며 최근에는 80% 이상의 점유율을 나타내고 있다[2].

또한 안드로이드 기기의 경우 애플리케이션을 사용했을 때 캐시가 기기에 저장된다. 생성된 캐시는 데이터나 값을 미리 복사해 놓는 임시 장소를 가리키며 이를 통해 데이터에 더 빠른 속도로 접근할 수 있게 된다.

즉, 안드로이드 기기에 저장되는 캐시 분석을 통해 사용자의 행위를 추적할 수 있다. 하지만 분석가들은 안드로이드 기기에 저장되는 다양한 종류의 캐시 Artifact에 대해 명확한 생성 목적 파악 없이 분석을 수행하고 있다.

따라서 본 논문에서는 애플리케이션에서 생성되는 캐시에 대한 분석 방안을 제시한다. 2장에서는 안드로이드 애플리케이션에서 생성되는 캐시에 대한 디지털 포렌식 관점에서의 연구 사례를 언급하고, 3장에서는 캐시의 종류 및 저장 방식, 캐시의 생성 목적 및 저널과의 상관관계를 분석하고 이를 기반으로 한 체계화된 분석 방안을 제시한다. 4장에서는 향후 수행될 수 있는 연구에 대해 논한다.

II. 관련 연구

안드로이드 운영체제에서 어플리케이션 사용

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2018-2016-0-00304)

시 자동으로 생성되는 캐시 정보는 디지털 포렌식에 주요하게 활용될 수 있는 정보로 여겨진다. 관련 연구로는 생성된 여러 캐시들의 분석 방법, 생성 구조 그리고 활용 방안에 대한 연구 등이 있다[3][4][5].

또한 알렉사를 대상으로 포렌식 모델 연구를 진행하며chromium라이브러리에서 발생하는 캐시에 대한 주요 구조를 분석한 바가 있다[6].

위의 논문들과 같이 안드로이드에서 제공하는 캐시 시스템이나 주요 라이브러리에 대한 캐시들에 대한 연구들이 선행되어 왔다. 또한 플랫폼이 다양화 되며 사용되는 라이브러리의 수나 캐시 시스템이 변경되어 왔지만 이에 대한 연구는 아직 미비한 것이 실정이다.

따라서 해당 연구에서 나아가 새로운 기술들에 적합한 포렌식 모델을 제안할 필요가 있다.

III.애플리케이션 캐시 분석

2.1 캐시의 종류 및 저장 방식

안드로이드 기기에 저장되는 캐시의 종류에는 메모리 캐시와 디스크 캐시가 있다. [표 1]은 안드로이드에서 생성되는 캐시의 종류와 특징을 나타낸 것이다.

[표 1] 안드로이드 캐시의 종류

캐시 종류	특징
메모리	처리 속도가 빠르다 저장 공간이 상대적으로 작다
디스크	처리 속도가 느리다 저장 공간이 상대적으로 크다

두 종류의 캐시 중 디스크 캐시를 저장할 때 사용하는 대표적인 외부 라이브러리의 종류와 저장 위치를 정리하면 [표 2]와 같다.

[표 2] 안드로이드 디스크 캐시 저장 라이브러리 종류

라이브러리	저장 위치
Android Universal Image Loader	uli-imges
Glide	image_manager_disk_cache
Picasso	picasso-cache
Chromium	org.chromium.android_webview

2.2 캐시의 생성 목적 및 구조

본 절에서는 두 가지의 캐시 종류 중 디스크 캐시에 초점을 맞춰 생성 목적 및 구조에 대해 분석한 결과를 서술할 것이다.

외부 라이브러리를 통해 생성되는 디스크 캐시의 경우 웹을 사용해 검색했을 때 사용된 이미지 및 URL등을 저장해 웹 컨텐츠의 처리를 가속화 하는 것을 목적으로 한다[7].

하지만 캐시의 생성 목적은 동일하나 생성되는 캐시의 구조에는 라이브러리별로 차이가 존재한다.

첫 번째로 Glide 라이브러리를 사용했을 때 생성되는 캐시는 image_manager_disk_cache 디렉토리 내에 저장이 되며 파일 명의 마지막에는 .0이 붙는다. 이를 Hxd를 통해 파일의 헤더 시그니처를 분석해보면 FF D8 FF E0(JPEG) 혹은 89 50 4E 47(PNG)인 것을 알 수 있다. 즉, Glide 라이브러리를 사용했을 때 생성되는 캐시는 이미지 파일이라는 것을 의미한다[8][9].

두 번째로 Picasso 라이브러리를 사용했을 때 생성되는 캐시는 picasso-cache 디렉토리 내에 저장이 되며 파일 명의 마지막에 .0 혹은 .1이 붙는다. 이를 Hxd를 통해 분석해보면 파일 명의 마지막에 .0이 붙을 경우 HTTP 요청 및 응답메시지에 대한 파일, 파일명의 마지막에 .1이 붙을 경우 HTTP 요청에 해당하는 Raw 데이터 파일인 것을 알 수 있다. 즉, Picasso 라이브러리를 사용했을 때 생성되는 캐시는 Raw 데이터 파일과 HTTP에 대한 정보를 의미한다[10].

세 번째로 Chromium 라이브러리를 사용했을 때 생성되는 캐시는 org.chromium.android_webview 디렉토리 내에 저장이 되며 파일 명의 마지막에 _0과 _1이 붙는다. 이를 Hxd를 통해 분석해보면 파일 명의 _0이 붙을 경우 요청 URL, Raw 데이터, HTTP 응답 정보의 순으로 이루어져있다. 또한 요청 한 URL이 js파일일 경우 파일 명의 마지막에 _1이 붙는 캐시파일이 생성된다.

2.3 생성된 캐시와 저널과의 상관관계

사용자가 애플리케이션을 사용함에 있어 다수의 캐시가 발생하게 된다. 이를 체계적으로 관리하기 위해서 캐시는 캐시의 상태를 기록하거나 캐시 데이터를 분할하여 관리하게 된다. 이를 캐시에 대한 저널(Journal) 혹은 인덱스(Index)라고 한다.

본 논문에서 언급되는 대표적인 캐시 라이브러리인 Picasso Library와 Glide Library는 Journal에 캐시의 상태를 기록하는 방식을 사용한다. Journal은 Journal을 기록하는 캐시 라이브러리의 정보가 기재된 헤더 영역과 실제 캐시에 대한 상태가 기록된 데이터 영역으로 구분된다. 이에 대한 상세한 구조는 아래와 같다.

[표 3] Journal Header Strucrure

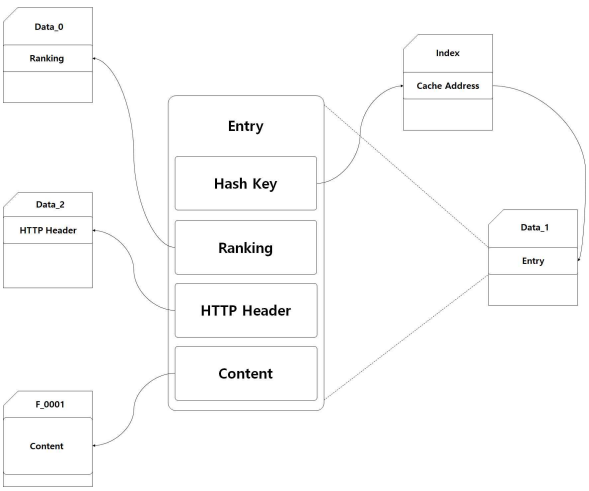
Contant String
Disk Cache Version
Application Version
Value Count

[표 4] Journal Data Structure

State	URL Hash	Meta Data Size	Content Size

State는 크게 CLEAN, DIRTY, READ로 구분된다. DIRTY는 캐시에 대해 쓰기 시작하는 경우, CLEAN은 이에 대한 작업이 완료 되었을 시, Read는 캐시 객체가 외부 요소에 의해 접근되는 경우를 의미한다. URL Hash는 요청한 URL 값을 Hash Function으로 Encoding 한 것을 의미한다. Meta Data Size는 0 파일의 크기를 나타내며 Content Size는 실제 캐시 된 1파일의 크기를 의미한다[11].

반면 org.chromium.android-webvew와 같은 라이브러리는 Index를 사용하여 캐시를 분할하고 이를 관리한다. 이에 대한 구조는 아래와 같다[12].



[그림 1] Chromium 캐시 저장 체계

Chromium 소스 코드의 헤더 파일을 일부 참조한 결과는 다음과 같다. 여러 개의 Entry (캐시 데이터) 들이 모여 하나의 블록을 구성하게 된다. Index 안에 Cache Address (Hash) 값은 특정 블록의 Entry를 지칭하게 된다. Entry 내부에 존재하는 Hash Key를 통해 Index에 접근이 가능하며 Index를 통해 Entry에 접근이 가능한 역함수 같은 존재이다. 이를 통해 Entry의 누락이나 Index의 누락 시에도 파일의 존재 여부를 판별하고 복구를 시도할 수 있게 된다.

2.4 저장되는 캐시에 따른 분석 방안

실생활에서 사용되는 애플리케이션은 다양한 라이브러리를 사용해서 캐시를 저장하게 되며 고성능의 기기라도 반응성을 위해서 캐시는 필수적으로 사용되고 있다. 실제 사용자의 사용으로 인해 데이터가 누적된 애플리케이션의 캐시 양은 방대하며 그 종류도 다양하다.

디지털 포렌식을 수행함에 있어 이러한 캐시들을 수집하고 분석하는 것은 사용자의 접근 패턴과 중요정보를 파악할 수 있는 중요한 가치를 지니고 있다. 하지만 구조의 난해함과 라이브러리 의존성이라는 문제로 캐시에 대한 연구가 미흡한 것이 현재의 상황이다.

이에 본 논문에서는 안드로이드 애플리케이션에 저장되는 주요한 캐시들을 분류하고 구조를 분석함으로써 캐시를 체계적으로 활용할 수 있는 디지털 포렌식 분석 방안를 제시하고자 한다.

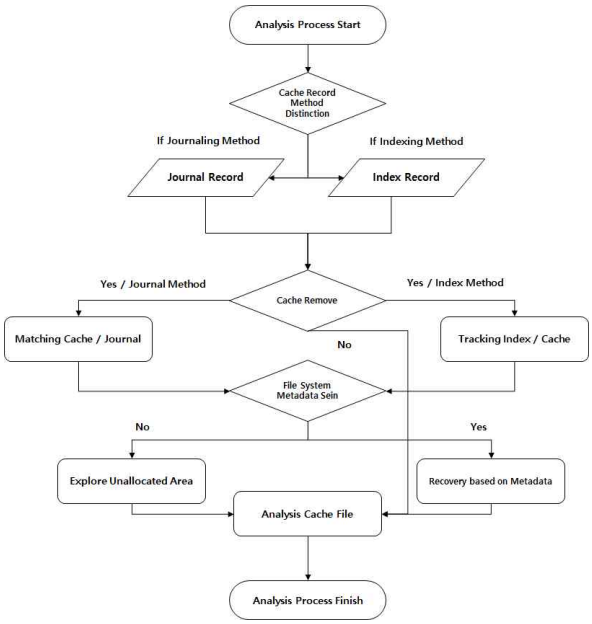
제안된 분석 방안은 캐시의 종류에 대해 먼저 파악함으로써 분석 방안 및 캐시 구조를 먼저 숙지할 수 있도록 한다. 또한 캐시의 수량과 상태를 나타내는 Journal 혹은 Index 방식을 파악하여 이를 기반으로 분석하며 추가적인 데이터 복구를 진행할 수 있도록 하였다.

대부분의 애플리케이션이 사용자의 캐시에 매핑(Mapping)되는 Journal이 존재하거나 캐시가 존재한다. 그러나 사용자가 의도적으로 삭제하거나 애플리케이션의 기능(로그아웃)으로 인해 강제 삭제되는 경우도 존재한다.

이러한 경우를 고려하여 전체 캐시 분석 프로세스에 캐시와 Journal의 존재 여부를 판별해

파일 시스템 메타데이터 혹은 비할당(Unallocated Area)에 대한 분석을 권고하고 있다.

즉, 이러한 프로세스를 도식화 하면 아래와 같으며 이를 통해 증거 수집 능력을 향상시키고 디지털 포렌식 및 침해사고 대응(DFIR)에 도움을 줄 수 있을 것이라 기대한다.



[그림 2] 캐시에 따른 분석 방안 제시 도식화

IV. 결론

본 논문에서는 안드로이드 환경에서 발생하는 캐시의 구조와 내용을 살펴보고 현장에서 주요 캐시를 분석할 수 있는 디지털 포렌식 기법을 제안하였다.

제안된 기법을 통해 안드로이드 애플리케이션에서 주로 활용되는 4개의 라이브러리와 라이브러리 사용에 따라 생성될 수 있는 다수의 캐시 아티팩트들을 분석하였다.

안드로이드 플랫폼의 활용성과 사용자가 점점 늘어나고 있는 만큼 이와 관련된 디지털 포렌식 기술에 대한 연구가 더욱 중요해지고 있다. 향후에는 주요 캐시에 대한 수집 분석 도구 개발 및 타 플랫폼에 대한 캐시 파악이 수행되어야한다.

즉, 본 논문에서 다룬 주요 캐시 분석 방안을 기반으로 타 플랫폼의 캐시 생성 목적 및 구조에 대한 구체적인 분석 및 활용 방안 연구에 활용될 수 있다.

[참고문헌]

[1] 연합뉴스, "'1인 1스마트폰 시대' 개막…가입자 수 5천만명 돌파", 2018.08.

[2] The Statistics Portal, "Global mobile OS market share 2009-2018, by qurter", 2018

[3] Chandrakumar, Felix Jeyareuben, et al. "An evidence-based Android cache forensics model." (2014).

[4] Immanuel, Felix, Ben Martini, and Kim-Kwang Raymond Choo. "Android cache taxonomy and forensic process." Trustcom/BigDataSE/ISPA, 2015 IEEE. Vol. 1. IEEE, 2015.

[5] Hazarika, Pinku, and Seshubabu Tolety. "Recommendations for webview based mobile applications on android." Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on. IEEE, 2014.

[6] Chung, Hyunji, Jungheum Park, and Sangjin Lee. "Digital forensic approaches for Amazon Alexa ecosystem." Digital Investigation 22 (2017): S15-S25.

[7] 조진철, "Android 웹뷰의 캐시 분석", NAVER D2, 2014.12.

[8] 박상권, "안드로이드 유용한 라이브러리 - Glide", 박상권의 삽질 블로그, 2015.07.

[9] nuuneoi, "PIntroduction to Glide, Image Loader Library for Android, recommended by Google", The Cheese Factory Blog, 2015.03.

[10] 정현우, "Picasso 파헤치기", spoqa 기술 블로그, 2014.03.

[11] Norman Peitek, "Retrofit 2 - Analyze Cache File", Futere Studio, 2017.05.

[12] The chromium Project, "Disk Cache Structure"

애플리케이션 디컴파일을 통한 디지털 포렌식 아티팩트 도출에 대한 연구*

유동균**, 신영훈***, 손태식**

**아주대학교 컴퓨터공학과

***아주대학교 사이버보안학과

A Study on Acquiring Digital Forensics Artifact via Application Decompilation

Dongkyun Yoo**, Yeonghun Shin***, Taeshik Shon**

**Department of Computer Engineering, Ajou University
***Department of Cyber Security, Ajou University

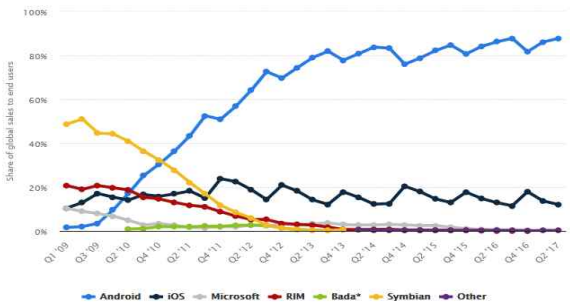
요 약

이 논문은 모바일 기기 중 80% 이상의 점유율을 갖는 안드로이드의 애플리케이션으로부터 필요한 포렌식 아티팩트를 선정하는 방법과 획득하기 위한 정적분석 방법을 서술한다. 정적 분석기법을 기초로 이를 정형화하여, 포렌식 아티팩트 도출을 자동화 하는 방법을 서술한다. 본 논문에서 제안하는 방법론을 통해 개발된 도구로 디지털 포렌식 과정에서 유의미한 아티팩트를 획득할 수 있음을 보인다.

Keywords reverse engineering, apk decompilation, analysis automation

I. 서론

모바일 기기의 사용률은 출시 이후 빠르게 증가해왔으며, 안드로이드 운영체제 기반 모바일 기기의 점유율 또한 아래 [그림 1]과 같이 지속적으로 증가하여 현재 80% 이상 차지하는 것으로 집계된다. 스마트폰은 사용자의 위치 정보, 검색 기록 등 사용자의 프로파일을 갖고 있으므로 범죄 발생 시 강력한 증거로 작용하며[1], 최근 IoT 홈서비스와 AI 비서 디바이스의 등장으로 더 많은 정보가 서버와 모바일 기기에 저장되고 있다. 위와 같은 이유로 안드로이드 운영체제를 대상으로 하는 다양한 디지털 포렌식 수사 기법의 연구가 진행되고 있다. 안드로이드 애플리케이션 내에 존재하는 데이터 분석을 통한 포렌식 기법에 대한 연구[2], 안드로이드를 기반으로 동작하는 모바일 디바이스에서 삭제된 사용 흔적을 복구하는 포렌식 기법에 대한 연구[3] 등 많은 연구가 진행되고 있지만, 안드로이드 애플리케이션 디컴파일을 적용해 아티팩트를 도출하는 포렌식 기법에 대한 연구는 악성코드 분석을 제외하면 거의 진행되지 않고 있다[4][5].



[그림 1] 2018년 1분기 스마트폰 시장 OS 점유율

상용 서비스로부터 수사에 도움이 되는 아티팩트를 얻기 위해 보통 중간자 공격(MITM)을 사용

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2018-2016-0-00304)

한다. 애플리케이션이 어떠한 정보를 서버로 전송하며, 반대로 서버로부터 얻어 올 수 있는 정보를 분석하여 클라우드 아티팩트와 클라이언트 아티팩트로 나누어 수사 관점에 도움이 되는 아티팩트를 수사관에게 제공할 수 있다. 그러나, 중간자 공격의 한계로써 이러한 분석 기법은 보안 연결 수립에 사용되는 인증서를 고정하는 방식에서는 효과를 볼 수 없다는 특성에서 제한적이다. 이러한 한계는 배포된 애플리케이션을 분석하여 개발 단계에서 개발자의 의도 자체를 분석함으로써 해결할 수 있다.

안드로이드 애플리케이션은 일반적으로 Java 언어로 작성되어 있으며, Java bytecode로 compile되고 DEX(Dalvik Executable) bytecode로 번역된다[6]. 개발자가 의도한 flow를 분석하는 것은 Java언어로 디컴파일된 소스코드에 대해서 진행하는 것이 일반적이나, 현재 상용화되어 있는 많은 안드로이드 애플리케이션은 배포 단계에서 소스코드 및 바이너리 난독화를 통해 방지되어 있다. 이러한 경우는 DEX bytecode를 분석하여야 하며, 가독성 등의 문제로 Java코드를 분석하는데 비해 오랜 시간이 소요된다.

본 논문에서는 소스코드 난독화와 같은 안티포렌식 기법이 존재하는 애플리케이션을 대상으로 여러 정적 분석기법을 적용하여 디지털 포렌식 관점에서 의의가 있는 아티팩트를 추출하고, 이를 자동화하는 방법에 대해 서술한다. 2장에서는 안드로이드 대상으로 진행된 디지털 포렌식 연구와 역공학(Reverse-Engineering)과 관련된 연구에 대해 서술하며, 3장에서는 안드로이드 애플리케이션 디컴파일을 통한 아티팩트 도출 시나리오와 시나리오를 기반으로 수행한 실험에 대해 서술한다. 마지막으로 결론에서는 3장에서 얻은 포렌식 아티팩트에 대한 정리와 아티팩트 도출 자동화 기법, 추후 연구 방향에 대해 서술한다.

II. 관련 연구

최근 모바일 디바이스의 사용량이 높아지고, 그 중 안드로이드를 기반으로 하는 디바이스가 많이 사용되면서 안드로이드의 보안에 대한 관심이 높아짐에 따라 연구에 대한 중요성이 증가하고 있다.

안드로이드 모바일 기기의 등장 이후 꾸준히 디지털 포렌식 연구가 수행되었으며, 안드로이드 기기의 보안을 위한 디지털 포렌식 연구로 Malware 탐지를 위한 포렌식 분석 또한 진행되었다.[4] 위 연구 또한 정적 분석(소스코드 분석)에 대한 중요성을 언급한다[5]. 소스코드 정적 분석에 대한 중요성이 언급되면서 안드로이드 애플리케이션을 디컴파을 통해 달빅 바이트 코드를 분석하는 기법에 대한 연구가 진행되었다. 위 연구에서는 달빅 바이트 코드에서 변수, 필드, 메소드 등에 대한 동적 분석에서 달빅 바이트 코드 분석을 통해 기존 동적 분석에서 얻을 수 없었던 유용한 세부 정보를 추출하였다[7].

안드로이드 애플리케이션 분석의 필요성이 증가함에 따라 안드로이드의 기본 애플리케이션인 통화, SMS, MMS, 브라우저 등에서 삭제된 데이터 복구를 통해 아티팩트를 도출하는 연구가 진행되었으며[3], 모바일 클라우드 애플리케이션 7종을 대상으로 애플리케이션 디렉토리 분석을 통해 얻을 수 있는 아티팩트에 대한 연구가 진행되었다[2].

하지만 IoT 서비스의 등장으로 악성 애플리케이션이 아닌 일반적인 애플리케이션 또한 사용자의 위치정보, 사용 흔적 등의 프로파일의 저장이 점점 증가하는 추세이며, 상용 애플리케이션을 대상으로 디컴파일을 통해 아티팩트를 도출하는 자동화된 기법에 대한 연구가 필요하다.

III. 애플리케이션 디컴파일을 통한 아티팩트 도출 시나리오

애플리케이션 디컴파일을 통해 아티팩트를 도출하기 위해 일련의 시나리오를 작성한다. 시나리오에는 아티팩트를 도출하기까지의 과정에 필요한 분석기법과 찾으려고 하는 아티팩트에 대한 정보가 포함된다. 이 시나리오는 여러 애플리케이션들에 대해 적용할 수 있도록 구성하며, 상황에 따라 서술한 시나리오와는 달리 분석 순서를

변경하여 적용할 수 있다.

[표 1] 분석 도구

도구명	버전	설명
Java	1.8.0_231	Java 런타임
APKTool	2.3.3	Smali code 변환
Jadx	0.8.0	Java code 변환

분석 환경은 안드로이드 애플리케이션을 분석하는데 있어 별도 제약사항 없이 일반PC에서 사용이 가능하다. [표 1]에 분석에 필요한 도구와 환경에 대해 요약하여 서술한다.

안드로이드 애플리케이션을 역공학에 이용되는 도구를 사용하기 위해서는 Java 런타임을 설치하여야 한다. 버전은 1.8.0을 사용하여야 하며, 이외 버전에서는 Jadx등의 분석 도구가 정상적으로 실행되지 않는 문제점이 존재한다. DEX파일을 Baksmaling하여 Smali코드로 변경하는 도구인 APKTool은 Smaling 또한 지원하여 DEX Bytecode를 다시 애플리케이션으로 변경할 수 있다. Jadx는 baksmaling과 bytecode translation을 동시에 지원하는 분석 프로그램으로, 분석에 필요한 최종 산출물인 Java code를 출력한다. 단, Java code의 직접적 수정은 제한되며 소스코드를 수정하려면 APKTool을 통해 smali를 추출 한 후 수정을 가해야 한다. ADB는 Android Debug Bridge의 약자이며 개발자 도구로써 활용된다. 개발한 애플리케이션을 안드로이드 디바이스에 설치하거나 기기의 Shell에 접근하는 등 다양한 기능을 제공한다.

시나리오 진행에서 중점적으로 다루어야 할 것은 최종적인 목표로 삼는 아티팩트와 접근하기까지의 단계 별 분석기법이다.

첫 번째로, 아티팩트는 찾으려는 혹은 검증하려는 것에 대해 구체적이고 논리적으로 접근하여 선정하여야 한다. 예시로써, 어떠한 데이터가 MITM을 통한 분석으로부터 내용이 암호화되어 클라우드 혹은 디바이스에 저장되는지 확신할 수 없는 상황이 있다. 이 때, 애플리케이션이 서버로 보내는 루틴이 존재한다면 해당 데이터는 클라우드 아티팩트가 될 수 있다. 이 경우에는 클라우드내 서버 주소가 애플리케이션 내에 저장되어 있거나 서버로부터 가져오는 루틴이 존재할 수 있다. 원격지 주소에 대해 대부분의 애플리케이션은 HTTPS요청으로써 데이터를 전송하게 되는데, 요청에 사용되는 URL을 추적하면 서버로 직렬화하여 전송하는 데이터를 추적할 수 있다. 결론적으로, URL schemetic의 signature를 통해 URL을 찾아낼 수 있게 된다.

두 번째로, 시나리오 진행의 중심인 분석기법에 대해 서술한다. 애플리케이션 디컴파일 과정과 그 산출물에서 중점적으로 다룰 부분은 소스코드 부분이며, 이는 DEX Bytecode와 Java code에 해당한다. Java code의 분석은 가장 가중치가 높은 분석 대상이며 정적 분석에 해당한다.

정적 분석은 4가지의 기법으로 진행된다. 문자열 기반 검색기법, 흐름 분석기법, 호출-계층 분석기법, 그리고 순차적 코드 분석기법이다. 대부분의 아티팩트를 정적 분석기법을 통해 획득 가능하며, 각 분석기법에 대해 아래 서술한다.

```
131        public final String m6657a() {
132            switch (C1706c.$EnumSwitchMapping$0[ordinal()]) {
133                case 1:
134                    return "https://dev-api.sktnugu.com/v1/";
```

[그림 2] 문자열 기반 검색기법을 통한 분석

[그림 2]은 문자열 기반 검색기법을 통해 URL schemetic을 찾아낸 내용이다. 이와 같이, 문자열 기반 검색기법은 디컴파일된 애플리케이션 파일에서 아티팩트를 직접적으로 검색할 수 있으며, 이들의 연관성을 통해 분석 지점의 entry point로 지정할 수 있게 하여 디컴파일된 애플리케이션을 최초 분석 시 유용하게 사용할 수 있다.

```
public final C2375m m6768a() {  Retrofit_builder (가명)
    if (C1750a.f5437d == null || !Intrinsics.areEqual(C1750a.f5436c, C1704a.f5394a.m6664a().
        C1750a.f5436c = C1704a.f5394a.m6664a().m6657a())
        C1489f a = new C1490g().m5981a(new C3240a()).m5980a();
        Intrinsics.checkExpressionValueIsNotNull(a, "GsonBuilder().setExclusi…usionStrategy(
        C1750a.f5437d = new C2374a().m8309a(C1750a.f5436c).m8314a((e$a) C7130a.a(a)).m8313a(
    } Retrofit instance  Retrofit.Builder().baseUrl(apiurl)  .addConverterFactory(~)  .client(~)
    return C1750a.f5437d;
```

[그림 3] 흐름 분석기법을 통한 분석

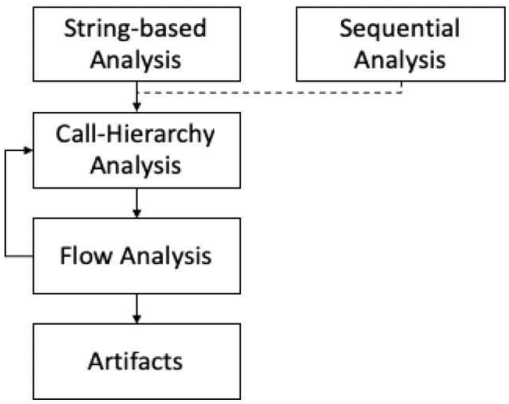
[그림 3]는 흐름 분석기법을 통해 해당 메소드가 어떤 역할을 하는 메소드인지 분석한 내용이다. 흐름 분석기법은 문자열 기반 분석기법의 후행 분석기법으로, 프로시저가 순차적으로 진행되

는 흐름을 따라가 개발자가 의도한 흐름을 이해하고 분석할 수 있게 한다. 대부분의 아티팩트를 흐름 분석기법을 종점으로 획득할 수 있다.

[그림 4] 호출-계층 분석기법을 통한 분석

[그림 4]는 호출-계층 분석기법을 통해 어떠한 메소드들이 대상 메소드를 호출하고 있는지 분석한 내용이다. 호출-계층 분석기법은 앞서 설명한 문자열 기반 분석기법과 흐름 분석기법에서 난독화 된 프로시저들의 작동 원리를 파악하기 위해 필수적인 분석기법으로, 각 함수들의 유기적인 호출관계로부터 필요한 아티팩트까지의 도달 과정을 계층으로써 이해하고 분석할 수 있게 한다.

순차적 코드 분석기법은 entry point를 지정할 수 없을 때 사용할 수 있으며, 이는 모든 소스코드에 대한 순차적 열람 및 분석을 의미한다. 이 기법은 가장 효율이 낮으나, 더 나아가 위 설명한 3가지 분석기법의 기반이 될 수 있다.



[그림 5] 정형화한 정적기법 순서도

위에서 진행한 정적 분석기법들을 [그림 5]으로 정형화할 수 있다. [그림 5]에서 주목해야 할 점은, 호출-계층 분석과 흐름 분석의 반복이 분석의 가장 많은 부분을 차지하게 된다는 점이다. 이 형태를 기반으로 다음과 같은 자동화 스크립트를 작성할 수 있다. 문자열 기반 검색을 진행하고 이를 사용하는 메소드를 추출하고, 해당 메소드의 호출-계층 구조에 따라 해당 프로시저의 종점까지 내려간다. 그 다음, 프로시저의 마지막에 최종적 목표인 아티팩트가 존재하는지 확인한다. 만약 존재한다면, 아티팩트를 출력한다. 만약 재귀적으로 생성된 나머지 프로시저의 분석이 아직 진행중이라면 해당 분석이 완료될 때 까지 대기한다.

interface	URI	Generic	MethodName
DeliveryApiInterface	/v1/integration/comm	AddressSearch	getCommerceAddresses(
AppCommonInfoInterface	/v1/setting/common/a	AppCommonInfo	getAppCommonInfo()
DeviceConnectionService	/v1/integration/btv/de	BtvDeviceInfo	getBtvDeviceInfo(@C236
DeviceConnectionService	/v1/auth/devices	CreateUserDevice	createUserDevice(@C234
DeviceConnectionService	/v1/auth/devices/statu	CreateUserDevice	getDeviceStatus(@C2362

[그림 6] 자동화 도구를 이용한 아티팩트 추출

[그림 6]과 같이, 정형화된 정적 분석 순서를 기반으로 자동화 도구를 개발하여 URL 아티팩트 추출에 성공하였다.

IV. 결론

제안하는 분석 기법을 통해 클라우드와 통신하며 기능을 제공하는 애플리케이션들로부터 포렌식 아티팩트를 추출할 수 있음을 보였으며, 분석기법을 자동화하여 개발한 도구를 통해 충분히 디지털 포렌식 현장에서 활용 가능함을 증명하였다.

추후 연구 방향으로써, 정형화 된 분석 기법 중 상단에 위치한 순차적 코드 분석기법을 어떠한 형태로 일반화할 수 있는지에 대한 연구와 이 연구에서는 다루지 않은 동적 분석 방법인 코드 삽입과 로그 분석의 자동화에 대한 주제로 진행 예정이다. DEX Bytecode 삽입의 경우 두 가지의 시나리오로 진행할 수 있다. 첫 번째로, 대부분의 애플리케이션이 같은 HTTP 라이브러리를 사용하고 있다는 점에 착안하여, 해당 라이브러리의 Intercept기능을 이용할 수 있다. 두 번째로는 애플리케이션 배포 시 삭제되지 않은 Debugging flag를 Bytecode 단계에서 활성화하여 해당 로그를 분석하여 아티팩트를 획득할 수 있다.

[참고문헌]

[1] Kasiaras, Dimitrios, et al. "Android forensics: Correlation analysis." Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for. IEEE, 2014.

[2] Martini, Ben, Quang Do, and Kim-Kwang Raymond Choo. "Mobile cloud forensics: An analysis of seven popular Android apps." arXiv preprint arXiv:1506.05533 (2015).

[3] 오정훈, and 이상진. "안드로이드 스마트폰 포렌식 분석 방법에 관한 연구." 디지털포렌식연구 9 (2012): 47-75.

[4] Schmidt, A-D., et al. "Static analysis of executables for collaborative malware detection on android." Communications, 2009. ICC'09. IEEE International Conference on. IEEE, 2009.

[5] Spreitzenbarth, Michael. "Dissecting the Droid: Forensic analysis of android and its malicious applications." (2013).

[6] Enck, William, et al. "A study of android application security." USENIX security symposium. Vol. 2. 2011.

[7] Desnos, Anthony, and Geoffroy Gueguen. "Android: From reversing to decompilation." Proc. of Black Hat Abu Dhabi (2011): 77-101.

NB-IoT 환경의 제어 영역 프로토콜 분석 툴 개발 및 보안성 분석+

심규찬*, 박신조**, 김홍일***, 김용대***

*한국과학기술원 정보보호대학원

**Technical University of Berlin

***한국과학기술원 전기전자공학부

NB-IoT Control Plane Protocol Analysis Tool and Security Analysis

Kyuchan Shim*, Shinjo Park**, Hongil Kim***, Yongdae Kim***

*KAIST Graduate School of Information Security

**Technical University of Berlin

***KAIST Electrical Engineering

요 약

NB-IoT (Narrow-Band IoT)란 Cellular-IoT (C-IoT)의 한 종류로 IoT 기기를 위한 무선통신 프로토콜로 사용하기 위해 이동통신 표준 단체 (3GPP)가 만든 표준이다. 기존의 LTE보다 작은 대역폭을 사용하여 낮은 전송률을 가지지만 이동통신망 기반을 활용함으로써 기존의 IoT 기술 대비 넓은 통신범위를 가진다. 새롭게 생긴 프로토콜인 만큼 서비스가 활성화 되기 전에 NB-IoT에 대한 보안성 분석이 필요하지만 현재 연구 목적으로 사용할 수 있는 진단 툴이 존재하지 않고 상업용 로그 분석 장비는 가격이 높아 쉽게 NB-IoT 보안 연구를 할 수 없다.

본 논문에서는 오픈 소스 소프트웨어를 이용하여 NB-IoT 제어 영역 분석 소프트웨어를 개발하고 이를 이용하여 찾을 수 있는 NB-IoT 기술의 취약점에 대해 알아보고자 한다.

I. 서론

Internet of Things (IoT)는 다양한 개체가 인위적인 개입 없이 상호 협력적으로 감지, 네트워킹, 정보 처리와 같이 지능적 관계를 형성하는 사물 공간 연결망을 뜻한다. 이를 통해 IoT 시대에는 대부분의 기기에 정보 획득 및 네트워크 연결 기능이 탑재되고 프로그램을 통해 사용자가 편리하게 이용할 수 있게 된다. 예를 들어, 삼성의 Smarthome, 구글 HomeControl과 같이 가정용 기기에서부터 의료, 제조, 수송, 유틸리티, 무기 산업에까지 IoT 기술을 활용한 서비스는 지속적으로 늘어가고 있고 2020년에 이르러서는 약 250억 개의 IoT 기기가 존재할 것이라는 전망이 있다[9].

+ 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2015-0-00403)

IoT 기기 사용이 늘어나면서 IoT의 한계점이라고 할 수 있었던 좁은 통신 영역을 보완하려는 기술들이 등장하기 시작했다. 주파수의 비면허 대역을 사용한 중장거리 IoT 통신 프로토콜 LPWAN (Low-Power Wide-Area Network)이 등장하게 되었고 이에 대응하여 이동통신 기반 Cat-M, NB-IoT 등의 표준이 발표되었다.

본 논문에서는 새롭게 생긴 IoT 통신 프로토콜 중 NB-IoT 프로토콜의 보안성 분석을 하고자 한다. 새롭게 생긴 IoT 통신 프로토콜인 만큼 서비스되기 전 NB-IoT에 대한 보안 연구가 필수적이기 때문이다. 하지만 공개된 NB-IoT 프로토콜 진단 및 분석 도구가 존재하지 않고 상업용 진단 도구 [5]는 값이 비싸 쉽게 구하기 어려운 제한점이 있다. 이를 극복하고자 기존 이동통신망 제어영역 메시지 분석

툴 Signaling Collection and Analysis Tool (SCAT) [8]을 이용하여 NB-IoT 프토로콜을 분석할 수 있는 툴을 개발하고 이를 이용하여 NB-IoT 표준상 발생 가능한 취약점에 대해서 논하고자 한다.

II. 배경지식 및 관련연구

2.1 NB-IoT

3GPP는 LPWAN에 대응하여 이동통신망을 이용하는 IoT용 LTE인 C-IoT 표준을 만들게 된다.

NB-IoT는 IoT 기기 지원을 위하여 저전력 모드인 Power Saving Mode (PSM)과 기존 LTE에 존재하던 Discontinuous Reception (DRX)를 확장한 extended Discontinuous Reception (eDRX)를 지원한다. 이를 통해 IoT 기기는 LTE 기기에 비해 배터리 수명이 획기적으로 늘어나게 된다.

또한 기존 LTE와는 다르게 Non-IP Data Delivery를 지원하며 적은 양의 NB-IoT 데이터에 대해 제어 영역에서 데이터를 전달하는 Control plane CIoT EPS optimization을 도입하였다. 이를 통해 적은 양의 데이터 전송을 위해 추가적으로 연결을 맺는 과부하를 줄이고 궁극적으로 저전력 모드를 지원 할 수 있다.

2.2 이동통신망 제어영역 메시지 분석 기술

실제 단말과 통신사의 이동통신망과의 동작 방식을 이해하기 위해서는 단말의 베이스밴드와 이동통신망이 서로 주고받는 제어영역 메시지 분석이 필수적이다. 하지만 어떤 메시지를 주고받는지는 단말 베이스밴드 제조사마다 형식이 다르며 포맷이 공개되어있지 않기 때문에, 제어영역 메시지를 분석하는 일은 매우 제한되어있다.

따라서 이를 분석하는 도구의 개발이 이루어졌으며, 대표적으로 UCLA에서 만든 MobileInsight [7]와 TU Berlin과 KAIST에서 개발한 SCAT [8]이 있다. 두 분석 도구 모두 단말 베이스밴드의 DIAG 메시지를 분석하여 단말이 이동통신망과 어떤 제어영역 메시지를 주고받는지 모니터할 수 있도록 하였다.

LTE 표준에서 단말과 망 사이에 주고받는 제어 영역 프로토콜은 크게 단말과 코어 네트워크간 주고받는 영역인 Non-Access Stratum (NAS)과 기지국 (eNB)과 무선 통신 구간에서 주고받는 Radio Resource Control (RRC) 두 가지가 있다. 앞에서 언급한 분석 도구([7], [8])들은 NB-IoT 프로토콜에 대한 분석은 지원 하지 않는다.

III. NB-IoT 제어 영역 메시지 분석 기술

3.1 개발 환경

대상 NB-IoT 기기는 퀄컴사의 베이스밴드 칩을 사용하였으며 분석 프로그램은 Ubuntu 16.04 LTS 환경에서 개발되었고, Wireshark 2.6.3 버전을 통해 분석할 수 있도록 개발하였다.

3.2 개발 과정

NB-IoT 메시지를 분석가능하게 하기 위하여 다음과 같은 과정을 거쳐 개발을 수행하였다. 우선, 퀄컴 NB-IoT 베이스밴드에 DIAG 포트를 열어 USB 통신을 할 수 있게 한다. DIAG 포트에 DIAG 마스크를 보내서 원하는 메시지 데이터를 받고, 이렇게 받은 데이터에서 NB-IoT 프로토콜 데이터 정보만을 파싱하는 작업을 거친다.

NB-IoT 프로토콜 Hex 데이터를 얻은 후 NB-IoT 표준 [6]을 참고하여 Hex 데이터를 해석한다. 먼저 Hex 데이터의 길이, Flag 등을 확인하여 분류한 후 각 필드가 의미하는 내용들을 알맞게 나타낼 수 있도록 구현하였다.

분석 툴을 이용해 나온 데이터를 보다 범용적으로 분석할 수 있도록 하기 위해 Wireshark를 활용하였다. Wireshark에서 파싱된 정보를 보기 위해서는 별도의 구현이 필요하며 이를 위하여 Personal Plugin을 만들었다. Personal Plugin은 Lua파일형식으로, 표준분석을 통하여 각 메시지의 형식을 정의하였다.

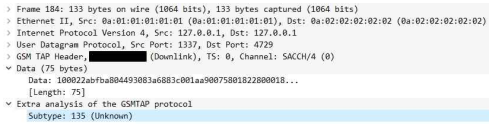


그림 1. 개발 툴 적용 전 데이터 모습

그림 1는 기존 Wireshark에서 NB-IoT 프로토콜 데이터를 본 모습이다. NB-IoT 프로토콜을 인식하지 못하여 데이터를 제대로 해석하지 못하는 모습을 볼 수 있다.

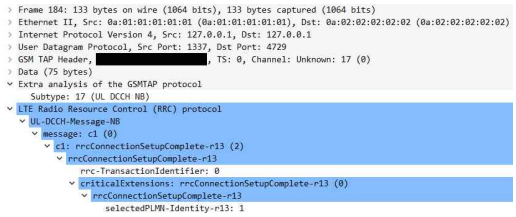


그림 2. 개발 툴에 의해 파싱된 모습

그림 2는 Personal Plugin을 설치한 후 모습이다. 프로토콜에서 RRC 채널 끝에 NB-IoT 프로토콜을 나타내는 NB 문구를 추가하였다. 이를 통해, 사업자, 제조사의 정책, 구현에 따라 다른 단말과 통신망 간에 프로토콜 과정을 분석할 수 있으며 암호화 및 무결성 보호 여부에 대해서도 확인 할 수 있다.

다음 장에서는 본 논문에서 개발한 분석 장비를 활용해 찾을 수 있는 상용 NB-IoT 서비스의 다양한 보안성 분석 사례에 대해 이야기하고자 한다.

IV. NB-IoT 분석 도구을 이용한 취약점 분석 활용

4.1 암호화 및 무결성 보호 유무

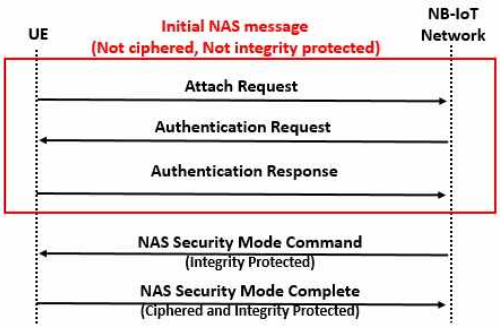


그림 3. NAS Security Setup 과정

기존 LTE에서 UE는 RRC 연결이 맺어지고 난 후 NAS Security Setup 과정을 통해 암호 알고리즘 및 NAS Security Key들을 공유하고, NAS 구간, 메시지 암호화 및 무결성 보호에 적용한다. 이 때 NAS 암호화 키는 그림 3과 같은 과정을 통해 도출된다. 1) UE가 MME로 Attach Request를 전송할 때 사용 가능한 Security 알고리즘 (암호화 및 무결성 보호)를 넣어 전송한다. 2) MME는 Attach Request를 수신하여 UE의 사용가능한 Security 알고리즘을 파악하고 이 중에서 알고리즘을 선택, NAS Security Setup 과정을 통해 UE로 전달한다. 결과적으로 UE와 MME에서는 같은 NAS Security Key들이 도출되어 NAS 메시지 전송 시에 이용한다. 따라서 NAS 메시지 암호화 키가 도출되기 전까지의 NAS 메시지들은 암호화가 되어 있지 않고 평문으로 전송 된다.

기존의 LTE와 다른 점은 NB-IoT 표준([2], [3])에 따르면 Control plane CIoT EPS optimization에 의해 NB-IoT 프로토콜은 User Plane 외에 Control Plane을 통해 IP/Non-IP packet을 보낼 수 있다.

따라서 NB-IoT 프로토콜에서 IP/Non-IP packet을 암호화하지 않고 Control plane CIoT EPS optimization을 이용하여 Initial NAS message를 통해 민감한 데이터를 전송하면 OTA (Over The Air)에서 도청 및 Spoofing 당할 수 있다.

또한 Initial NAS message에서 Authentication Reject이 발생할 경우 이전 NAS message에서 Control plane CIoT EPS optimization을 이용해 보낸 IP/Non-IP packet의 전송 여부도 확인해 볼 필요가 있다. Authentication 확인하기 전에 Packet 전송이 가능하다면 Invalid한 NB-IoT SIM 카드를 이용하여 과금 우회 공격도 가능하다.

본 논문에서 개발한 NB-IoT 분석 툴을 이용하면 NAS Security Setup 되기 전 암호화 및 무결성 보호 알고리즘이 적용되지 않은 NAS 메시지에 IP/Non-IP packet이 있는지 확인할 수 있으며, 이 패킷이 별도의 암호화가 되어 있는지 여부도 확인할 수 있다.

4.2 PSM 진입 방해 가능성

NB-IoT 표준 [3]에 의하면 NB-IoT 기기가 PSM 모드에 들어가기 위해서는 T3324, T3412 라는 PSM 관련 타이머가 필요하다. 이 타이머 값들은 NB-IoT 기기가 이동통신망에 메시지를 보낼 때 함께 보내게 되며 이동통신망에서 확정된 타이머 값들을 받게 된다. 이를 통해 PSM 모드에 진입하게 되고 전력 소모를 줄일 수 있게 된다.

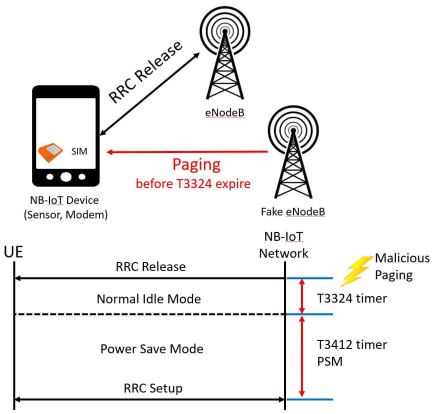


그림 4. PSM 진입 방해 공격 도식도

T3324 타이머는 PSM 모드 진입 전까지의 시간을 나타내고 T3412 타이머는 PSM 모드 유지 시간이다. 정상적으로 T3324, T3412 값을 이동통신망으로부터 받게 되더라도 T3324 시간 사이에 망으로부터 메시지를 받게 되면 PSM 모드에 진입할 수 없게 된다. 따라서 공격자가 그림 4와 같이 가짜 기지국 혹은 신호 주입 공격을 통하여 T3324 시간 안에 NB-IoT 기기에 지속적으로 메시지를 보낸다면 NB-IoT 기기의 배터리가 빨리 소모되어 정상 작동을 할 수 없게 된다.

본 논문에서 개발한 도구로 ATTACH REQUEST, ATTACH ACCEPT, Tracking Area Update (TAU) 등의 메시지들을 분석하여 T3324 타이머 설정에 대한 정보를 도출해 낼 수 있으며, 추가로 배터리 소모 공격을 위한 악의적인 Paging 메시지 검출이 가능하다.

V. 결론

본 논문에서 개발한 소프트웨어는 첫 번째로 만들어진 오픈소스 NB-IoT 제어 영역 프로토콜 진단 도구이다. 이 도구를 사용하면 비싼 상업용 진단 도구 [5] 없이도 시간 및 비용 효율적으로 취약점을 진단할 수 있다. 이 분석 도구는 지속적으로 사용이 증대되는 NB-IoT 서비스에서 개인정보를 보호하고 사용성을 증대시킬 수 있다.

그리고 NB-IoT 표준 상 발생 가능한 취약점에 대해서 정의하고 이를 분석할 수 있는 방안을 논의해 보았다. 본 논문에서 개발한 도구를 활용하여 NB-IoT 프로토콜에서 사용하는 암호화 및 무결성 보호 알고리즘 종류, 적용 여부에 대해서 분석이 가능하다 IoT 서비스 목적으로 개발된 기술이기에 저전력, 간소화에 중점을 두고 만들어진 프로토콜이지만 간편한 활용성을 이유로 개인의 정보를 다루는 어플리케이션이나 서비스가 사용되고 있으므로 보안성에 대한 분석과 지속적인 점검이 필요하다.

본 논문에서 제시한 진단 도구는 퀄컴 제조사만이 아니라 다양한 제조사에 대해서도 동일한 방식으로 적용 확장 가능하다. 후속연구에서는 지속적인 베이스밴드 제조사와 프로토콜 업데이트를 통해서 범용성과 효율성을 가진 도구를 만들 예정이다.

[참고문헌]

[1] https://github.com/fgsect/scat
[2] 3GPP TS 36.300 v15.3.0
[3] 3GPP TS 24.301 v15.4.0
[5] http://www.innowireless.co.kr/index.asp
[6] http://www.3gpp-message-analyser.com/decoder/eurrc.htm
[7] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, Tao Wang, MobileInsight: Extracting and Analyzing Cellular Network Information on Smartphones, MobiCom 2016
[8] Byeongdo Hong, Shinjo Park, Hongil Kim, Dongkwan Kim, Hyunwook Hong, Hyunwoo Choi, Jean-Pierre Seifert, Sung-Ju Lee and Yongdae Kim, Peeking over the Cellular Walled Gardens – A Method for Closed Network Diagnosis, IEEE Transactions on Mobile Computing, Vol. 17, 2018
[9] IoT 현황 및 주요 이슈, 2014, IITP

이종 IoT 플랫폼 간 인터워킹을 위한 장치 식별자 기반 관리체계

구자훈* 김영갑*,†

*세종대학교 정보보호학과

Device Identifier based Management System for Interworking between Heterogeneous IoT Platforms

Jahoon Koo* Young-Gab Kim*,†

*Division of information security, Sejong University.

요 약

최근 사물인터넷(IoT; Internet of Things) 기술의 개발과 함께 다양한 IoT 플랫폼이 개발 중이다. 그러나 이러한 IoT 플랫폼들은 고유한 장치 식별 시스템(ID)을 기반으로 개발 중이며 다른 플랫폼의 장치를 인식하고 서비스 또는 자원을 사용하는 것이 어려운 상황이다. 현재 이종 IoT 플랫폼 간의 자원 상호 운용성에 관한 연구는 진행 중이지만 IoT 플랫폼에서의 장치 식별자와 자원 요청 포맷에 대한 연구는 미흡하다. 본 논문에서는 이러한 이종 IoT 플랫폼 간의 상호운용을 위하여 각 플랫폼에서의 장치 식별 시스템과 자원 요청 형식을 분석하고 이를 해석하여 변환하는 IoT DNS(Device Name System)를 제안한다.

I. 서론

최근 사물인터넷(IoT; Internet of Things), 빅데이터 기술로 인한 모든 사물이 인터넷으로 연결되는 초연결사회(Hyper-connected Society)로의 진행이 급속하게 진행되고 있다. 또한 IoT 기술의 개발과 함께 다양한 IoT 플랫폼(oneM2M, OCF 'IoTivity', Apple 'HomeKit', 삼성 'ARTIK', Google 'Brillo/Weave', AllSeen Alliance 'AllJoyn', IBM 'Watson IoT', GS1 'Oliot' 등)이 개발 중이다. IoT 플랫폼은 다양한 센서, 액세스 포인트 및 데이터 네트워크를 포함한 장치를 연결하고 확장하며 다양화하는 소프트웨어이다. 따라서 다양한 IoT 플랫폼과 관련된 상호운용 가능한 서비스를 제공하는 것이 필수적이다. 그러나 이러한 IoT 플랫폼들은 고유한 장치 식별 시스템(ID)을 기반으로 개발 중이다. 즉, 장치 식별자(Device Identifier)를 포함한 자원 요청 포맷이 서로 다르기 때문에 다른 플랫폼 기반의 장치를 식별하는 것이 어렵다. 현재 이종 IoT 플랫폼 간의 자원 상호 운용성에 관한 연구는 진행 중이지만 IoT 플랫폼에서의 장치 ID 시스템에 대한 관심은 거의 집중되지 않고 있는 상황이다. 이 문제를 극복하기 위해서 이종 IoT 플랫폼의 장치 ID 시스템을 비교 분석하고, 이를 기반으로 IoT Device Name System(DNS) 아키텍처와 시나리오를 제안하여 개발하는 것을 목적으로 한다.

본 논문은 2장에서 이종 IoT 플랫폼 간의 상호운용을 위한 관련 연구를 소개하고 3장에서 4개의 IoT 플랫폼(oneM2M, GS1 'Oliot', IBM 'Watson IoT', OCF 'IoTivity')의 장치 ID 시스템과 자원 요청 형식을 분석한다. 4장에서는 이러한 장치 ID 시스템과 자원 요청 형식을 다른 플랫폼의 형태로 변환하여 주는 IoT DNS를 제안한다.

† 교신저자

II. 관련 연구

oneM2M 표준화 기구는 non-oneM2M 플랫폼(LWM2M, AllJoyn, IoTivity)과의 연동을 위해 IPE(Interworking Proxy Entity)를 개발 중이다. IPE는 non-oneM2M 플랫폼의 자원을 oneM2M 자원 형식으로 변환하고 기존 자원에 추가적인 자원을 생성한다. oneM2M 게이트웨이에 연결된 non-oneM2M 플랫폼 기반 장치는 일정한 주기로 게이트웨이에 자원 검색 결과를 제공하고 게이트웨이는 결과를 oneM2M 자원 트리 구조로 변환하여 게이트웨이에 저장한다. oneM2M 장치는 게이트웨이에서 oneM2M 리소스 트리 구조로 변환 된 non-oneM2M 플랫폼 장치 자원 및 서비스를 요청할 수 있다[1]. 그러나 게이트웨이에 모든 플랫폼 장치의 자원이 생성되어 존재하므로 게이트웨이 성능에 영향을 받는다. 우리는 게이트웨이에 직접 자원을 생성하지 않고 자원 요청을 변환을 목적으로 한다. 기존 연구, J.H. Koo et al. [2], 에서는 자원 요청 변환을 위해 자원 요청에 사용되는 장치 ID를 각 플랫폼마다 분석하였다.

III. 이종 IoT 플랫폼 분석

3장에서는 이종 IoT 플랫폼의 장치 ID 시스템과 자원 요청 형식을 분석한다. 이종 IoT 플랫폼으로는 oneM2M, GS1 'Oliot', OCF 'IoTivity', IBM 'Watson IoT'를 선택하여 분석하였다. 이 IoT 플랫폼들은 오픈 소스로 개발 중이고 국제적으로 영향력 있는 역할을 한다. oneM2M은 M2M(Machine to Machine) 표준을 구현하기 위해 구성된 국제 표준화기구로서 객체 통신, IoT 아키텍처, API(Application Programming Interface) 규격 및 보안 솔루션 요구 사항을 제공한다. GS1(Global Standard 1)은 유통 및 물류를 포함한 모든 산업에서 사용되는 제품 식별 바코드, 전자 문서, 전자 카탈로그 등을 표준화하는 민간 국제 표준 기관이며 IBM(International Business Machines Corporation)은 IT 서비스 및 컨설팅 분야의 글로벌 리더로서 글로벌 컴퓨터 시장의 약 50 %를 차지한다. OCF(Open Connectivity Foundation)는 IoT 산업에 대한 상호 운용성 지침, 표준 개발 및 인증 프로그램을 제공하는 조직이다.

표 1. 이종 IoT 플랫폼의 장치 식별 체계

Platform	Feature	Identification Format
oneM2M	OID based	OID(Higher Arc).ManufacturerID.DeviceTypeID.DeviceSerialNo
GS1 Oliot	OID based	GS1 OID({2.51}).Identification Keys(1).[ID Key Type], Value
IBM Watson IoT	Client ID	d:orgID:deviceType:deviceID
OCF IoTivity	resource type, device ID	[di], rt: oic.wk.d , oic.d.[*]

3.1 장치 ID 시스템

각 플랫폼이 사용하는 장치 ID 시스템은 표 1과 같다. 현재 oneM2M은 같은 플랫폼 기반 장치를 식별하기 위하여 장치 ID를 사용하지 않고 자원 ID를 사용한다. 그러나 장치 간의 구별을 위해서 장치 ID(di; Device Identifier) 개념은 존재한다. oneM2M의 장치 ID는 OID(Object Identifier)를 기반으로 하며, 제조사, 장치유형, 일련번호를 추가하여 사용한다. OID란 객체의 식별을 위한 트리 구조의 체계이며 국제전기통신연합(ITU-T; International Telecommunications Union - Telecommunication), 국제표준화기구(ISO; International Organization for Standardization) 및 국제전기기술위원회(IEC; International Electronical Committee)에서 지원 및 권장되며 ITU-T 권장 사항과 ISO, IEC 국제 표준에 사용된다. GS1 Oliot도 OID를 기반으로 하는 장치 ID 시스템을 사용하며 추가적으로 GS1 표준에 맞는 키 유형(GS1 ID Key)과 값을 사용한다. IBM Watson IoT는 'Client ID'라는 장치 ID를 사용하며 사용자 조직 ID, 장치 유형, 장치 ID 값을 포함한다. OCF IoTivity는 자원유형(rt; resource type)과 장치 ID(di; device ID) 값을 사용하여 장치를 식별한다[1].

3.2 자원 요청 포맷

장치 ID는 oneM2M 이외의 다른 플랫폼(Oliot, Watson IoT, IoTivity)에서 자원을 요청하기 위한 파라미터로서 사용된다. oneM2M은 그림 1과 같은 자원 구조를 가지며 IN(Infrastructure Node), MN(Middle Node), ASN(Application Service Node) 및 ADN(Application Dedicated Node)으로 구성된다. ADN을 제외한 다른 노드에는 CSE(Common Service Entitiy)가 있다. CSE는 그림 1과 같이 트리 형식으로 나타내며 각각의 자원은 자원 ID(Resource ID)와 자원명(Resource Name)을 가지며 이러한 값을 사용하여 자원에 접근할 수 있다. 예를 들어, resourceID가 '006'이고 resourceName이 'dev02'인 ASN-AE를 요청하려면 자원 경로로 'server01/gateway01/asn01/ dev02'을 사용한다. GS1 'Oliot'은 EPCIS(Electronic Product Code Information Service)를 사용하여 장치 및 자원을 이벤트 형식으로 저장하고 관리한다. Oliot 장치의 자원을 식별하고 요청하기 위해서 'urn:epc:id:[GS1 ID key Type]:[GS1 ID Key Value]' 형식이 사용된다. GS1 ID key type은 GS1 표준에 정의된 Key 유형이며 GS1 ID Key Value에는 Company Number와 장치 일련번호가 포함된다. IBM 'Watson IoT'는 특정 장치의 자원을 요청할 경우, 표 2와 같은 형식을 사용한다. 'Watson IoT'는 HTTP와 MQTT 프로토콜 사용에 따라 차이가 있는 형식을 지원하며 두 형식에는 공통적으로 장치 유형, 장치 ID, logicalInterfaceID가 포함된다. logicalInterfaceID는 논리 인터페이스나 사용자 지정 얼라이어스 이름을 위해 사용된다. OCF 'IoTivity'에서 사용하는 리소스 요청 형식은 'ocf://<authority>/<path>?<Query>'이며 이 형식에서 'di'는 <authority>에 포함되고 UUID(Universally Unique Identifier)를 사용하는 36자리의 값이다. 예를 들어, OCF 'IoTivity'의 요청이 'GET ocf://<di>/a/room/1?if=oic.if.ll'이면 /a/room/1은 <path>가 되고 뒷 부분은 쿼리문이다[1, 3].

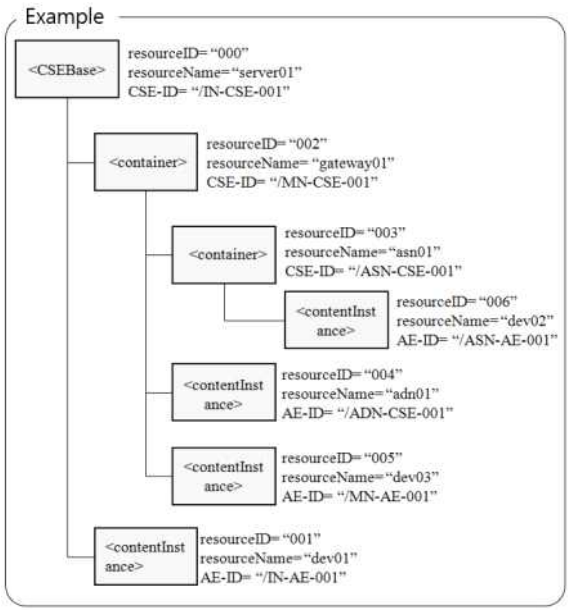


그림 1. oneM2M 자원 트리 구조 예시

표 2. Watson IoT 자원 요청 형식

Protocol	Format
MQTT	iot-2/type/${typeId}/id/${deviceId}/intf/${logicalInterfaceId}/evt/state
HTTP	GET /device/types/{typeId}/devices/{deviceId}/state/{logicalInterfaceId}

IV. IoT Device Name System

분석한 결과에 따르면 4개의 IoT 플랫폼은 서로 다른 장치 식별자 및 자원 요청 형식을 사용하며 다른 플랫폼 기반의 장치와 자원 식별이 어려운 상황이다. 이러한 문제를 해결하기 위하여 IoT 플랫폼의 장치 간 식별체계를 해석하고 다른 플랫폼의 체계로 변환하는 IoT

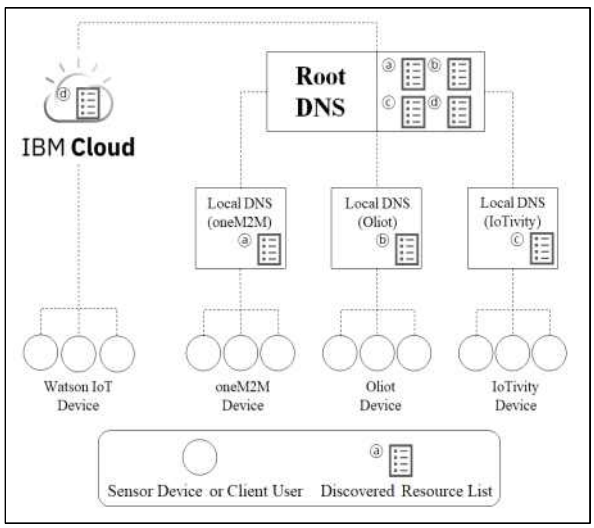


그림 2. IoT DNS 아키텍처

표 3. Root DNS 자원 매핑 테이블 예시

PN	Resource Name	Device ID	Resource Path	use-token-auth
1	Humidity	0.2.481.1.10.303.401	http://:210.107.234.131:8000/yt/ae01/humidity	null
1	LED	0.2.481.1.10.303.1001	http://192.168.0.8:7579/mobius_yt/ae-test02/cnt-led	null
1	Temperature	0.2.481.1.10.303.400	http://:210.107.234.131:8000/yt/ae02/temperature	null
3	Humidity	Testdev123	http://8slmq7.messaging.internetofthings.ibmcloud.com:1883/api/v0002/device/types/testraspberry01/devices/dev01/events/measure	+HdleToN2soZKTP+q0
3	LED	sensor01	http://0tvvm2.messaging.internetofthings.ibmcloud.com:1883/api/v0002/device/types/raspberry/devices/sensor01/events/blink	*QsetH12SuzHST_+7t
3	Temperature	sensor	http://8slmq7.messaging.internetofthings.ibmcloud.com:1883/api/v0002/device/types/testraspberry02/devices/dev02/events/calculate	*sdleT82TsoZKT_+7t

Device Name System(DNS)을 제안한다. IoT DNS의 아키텍처는 그림 2와 같다. Local DNS는 각 플랫폼의 자원 검색(Resource Discovery) 결과를 저장하고 있으며 Root DNS는 이러한 결과를 일정 주기마다 요청하여 하나의 테이블로 저장하고 관리한다. 사용자는 접근하여 사용할 수 있는 각 플랫폼의 자원 검색 결과를 얻을 수 있다. 사용자가 자원을 선택하고 요청하면 Local DNS를 통하여 Root DNS로 요청을 전달한다. Root DNS는 Local DNS로부터의 요청이 어느 플랫폼의 장치인지 식별하고 선택된 자원의 플랫폼 요청 포맷에 맞게 재구성한다. Root DNS는 IoT 플랫폼으로부터 받는 요청문을 사용자가 선택한 자원의 플랫폼이 사용하는 요청 형태로 변환하기 위한 매핑 테이블을 갖고 있다. 이 매핑테이블에는 사용자가 장치를 등록할 때 필요 정보가 기입된다. IBM Watson IoT'의 경우, 사용자 인증 토큰(use-token-auth)값을 추가적으로 기입 받는다. RooT DNS는 변환되어 재구성된 요청문을 해당 플랫폼을 향해 요청하고 사용자는 원하는 자원에 접근하거나 서비스를 제공받을 수 있다.

V.결론

본 논문에서는 이종 IoT 플랫폼 간의 장치와 자원의 상호운용에 관한 문제를 해결하기 위해 장치 식별 시스템을 기반으로한 관리체계인 IoT DNS를 제안한다. IoT DNS는 이종 IoT 플랫폼의 식별체계를 해석하고 저장된 요청 형식에 맞게 변환하기 때문에 oneM2M 표준화 기구에서 연구중인 IPE와는 달리 게이트웨이에 모든 리소스를 저장하지 않으므로 게이트웨이 성능에 크게 영향을 받지 않는다. 현재 IoT DNS는 테스트베드로서 oneM2M과 Watson IoT 간에 상호운용이 가능하고 추후 Oliot과 IoTivity와도 가능하도록 구현 중이다.

[Acknowledgement]

이 연구는 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2017-0-00756, IoT 이종 식별체계 상호연동 및 관리 체계 기술 개발)

[참고문헌]

[1] Jaeseok Yun, Ramnath Chekka Teja, Nan Chen, Mak-Myoung Sung, Jaeho Kim, "Interworking of oneM2M-based IoT systems and legacy systems for consumer products", ICTC, October, 2016

[2] Jahoon Koo, Young-Gab Kim, "Interoperability of Device Identification in Heterogenous IoT Platforms", ICENCO, December, 2017

[3] 이원석, 차홍기, 전종홍, "사물인터넷 오픈소스 기술 - IoTivity", 한국통신학회논문지 32(12) pp.27-35, 2015

IoT 센서 데이터의 속성기반 암호화를 통한 안전한 서비스 모델 개발*

함나연[1], 김형종[1]

[1]서울여자대학교 정보보호학과

A Study on a secure model with attributes-based encryption(ABE) of IoT sensor data

Na-Youn Ham, Hyung-Jong Kim

Division of information security, Seoul Women University

요 약

인터넷을 통해 다양한 사물들을 연결하는 IoT의 시대가 도래하면서 데이터의 통신 또한 활발하게 이루어지고 있다. 하지만 개인 정보나 산업 정보와 관련된 데이터들이 무분별하게 통신 되면서 개인 정보의 침해가 우려되고 있다. 이에 따라 적절한 기준에 따라 암호화를 하고, 권한을 가진 사용자만이 복호화를 할 수 있도록 하는 정책의 필요성이 대두되고 있다. 이에 특정 속성과 일치하는 수신자만이 데이터를 열람할 수 있도록 하는 속성기반 암호화(ABE)정책을 사용하여 데이터의 허가 받지 않은 접근을 차단할 수 있다. IoT 센서의 데이터에 속성기반 암호화를 적용하면 개인의 속성을 기반으로 암호화가 이루어지며 속성의 집합과 일치하는 경우에만 복호화가 가능하다. IoT 센서들의 활용과 함께 데이터 암호화의 표준이 요구되고 있는 지금, 속성기반 암호화를 적용한 정책은 안전한 통신과 더불어 IoT의 활용을 증가시킬 수 있을 것으로 기대된다.

Keyword: IoT, Attribute Based Encryption, Privacy Protection

I. 서론

IoT 기기와 서비스의 규모가 커지면서 IoT의 센서 데이터를 이용한 통신 또한 활발하게 이루어지고 있다. 2020년까지 130조의 기기가 연결되어 가정, 의료, 산업 등 다양한 분야에서 활용될 것으로 예측된다[1]. 하지만, IoT의 성장과 더불어 IoT 보안의 필요성은 사이버 보안에서 제일 중요한 사안으로 대두되고 있다. 특히 센서를 사용하는 데이터의 경우 센서의 값들이 허가되지 않은 접근으로 인한 유출이 될 위험이 크다. 데이터의 무분별한 접근은 센서 데이터를 통해 개인의 현재 위치나 심장 박동수 같은 생체 정보에도 접근할 수 있으며 카메라 센서 데이터에 접근해 개인의 사생활도 침해할 수 있다는 문제가 있다. 2017년 FDA에서는 심장 박동 조절 장치의 송신기에 권한이 없는 사용자가 환자의 RF에 원격으로 액세스할 수 있다는 성명문을 발표했다. [2] 이러한 문제점을 해결하기 위해 속성기반 암호화 (ABE) 정책을 통해 센서 데이터를 개체의 속성에 따라 암복호화를 실시할 수 있도록 할 수 있다. 본 논문에서는 속성기반 암호화를 통해 IoT 센서 데이터가 안전하게 통신 될 수 있는 서비스 모델에 대해 제안하였다. 2장에서는 IoT 장치 간 통신에 관한 연구와 속성기반 암호화를 이용한 연구에 대해 소개하고, 3장에서는 속성기반 암호

*이 연구는 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2017R1D1A1B03034644).

화를 IoT 데이터에 적용한 서비스 모델에 대해 제안하고 있다.

II. 기존연구

2.1 IoT 장치 간 통신에 관한 연구

IoT 통신환경에서 사용되는 암호화 방식은 현재 대칭 키 암호화, 공개 키 암호화 방식을 사용한 인증 등이 사용되고 있다[3]. 국내에서는 세션 키 방식을 이용해 IoT 암호화 기술을 설계한 연구가 진행되었는데, 인증 및 키 관리를 위한 서버를 따로 추가하여 IoT 기기 사이에서 등록, 인증 중계, 세션 키 관리를 담당함으로써 데이터 암호화와 서버의 진위를 따지는 추가적인 인증 처리를 가능하게 하였다. 공개 키 암호화 방식을 사용한 인증기술을 변형하여 세션 키 공유를 할 수 있으며, 추가적인 인증을 통해 인증 및 키 관리 서버를 확인할 수 있다. 또한, 암호화 키가 유출되었을 때 세션키 방식으로 대처할 수 있다는 점을 시사하였다.

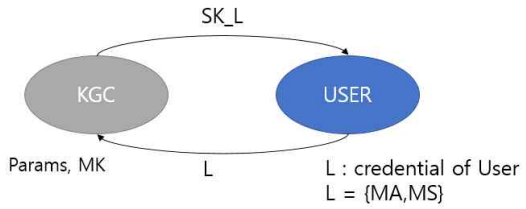
2.2 속성기반 암호화를 이용한 연구

속성기반 암호화는 ID 기반 암호화 (IBE)의 확장된 개념으로 각 개체의 속성을 기초로 암호화, 복호화가 실시되는 구조이다[4]. 해당기술을 메신저 환경에 적한 연구가 국내에서 제안되었으며, 디바이스 간의 안전한 암호화와 키 동기화 기능을 가능하게 하여 스마트 폰 기기 분실, 변경에 따른 메시지 동기화를 제안하였다[5]. 또 다른 연구에서는 속성기반 암호화 기술을 이용해 다중 서버 패스워드 프로토콜을 제안하였다[5]. 패스워드 인증 키 교환 프로토콜은 서버와 클라이언트가 서로 인증하고 키를 교환하는 알고리즘이다. 속성 기반 암호화 방식을 이에 적용하여 속성 값을 패스워드로 하여 별도의 키를 생성하지 않고, 공개키로 서버를 암호화 하는 방식을 고안하였다.

III. 연구내용

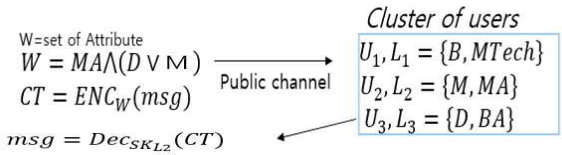
3.1 속성기반 암호화 배경지식

속성기반 암호화는 1-to-1 방식을 사용하는 PKE와 IBE 방식과는 달리, 1-to-many 방식을 사용하여 브로드캐스트의 문제를 해결할 수 있다는 장점이 있으며 Fine-Grained 접근제어를 사용한다는 특징이 있다. 개체의 속성 정보의 집합과 속성의 접근 구조를 바탕으로 암복호화를 실시하는 방식으로, 접근 구조를 암호화시에 지정하는 CP-ABE(Cipertext-Policy)와 키 생성이 접근구조를 지정하는 KP-ABE(Key Policy)로 나뉘게 된다.



[그림 1] KGC 키 발급 과정

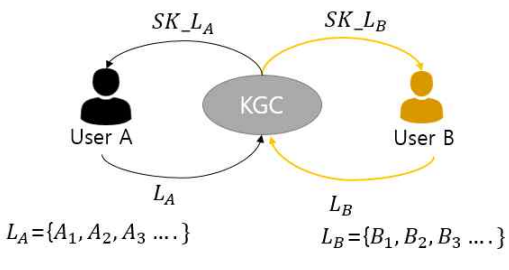
여기서는 암호문 생성시 접근구조를 사전에 정의하는 CP-ABE로 서비스를 설계하였다. 다음은 ABE가 사용자에게 키를 발급하는 과정이다. [그림 1]의 KGC는 Key Generation Center로 , 키 발급 센터이다. 사용자는 자신의 정보인 L를 KGC로 전송하고, KGC는 사용자의 속성을 Params와 MK(Master Key)로 받아 사용자를 식별하는 SK_L을 보내게 된다. 사용자의 개인 정보를 기반으로 한 키 발급이 이루어진 후에 암호화, 복호화가 가능하다. 다음 예시는 암호화와 복호화가 이루어지는 과정이다. 사용자들의 속성의 집합인 W로 메시지를 암호화하여 전송하며, 사용자의 속성으로 다시 복호화하게 된다. 학교 데이터베이스를 예시로 들었을 때, 복호화 규정인 W는 MA(Math)학과이고 학위는 D(Doctor) 이거나 M(Master)인 경우만 복호화가 가능하다고 규정하고 있다. 메시지인 msg는 KGC에서 사용자의 속성을 기반으로 발급하였던 비밀키를 기반으로 보호하게 된다. 속성의 집합에 의해 CT로 암호화 된 상태인 것이다. Cluster of Users에서 규정과 일치하는 U_2인 사용자가 복호화를 할 수 있는 속성을 갖게 되고 그 외의 사용자들은 접근이 제한되게 된다. 복호화는 암호문인 CT에 권한이 있는 사용자 U_2의 속성인 L_2를 기반으로 복호화 하게 된다.



[그림 2] 속성기반 암호화, 복호화 과정

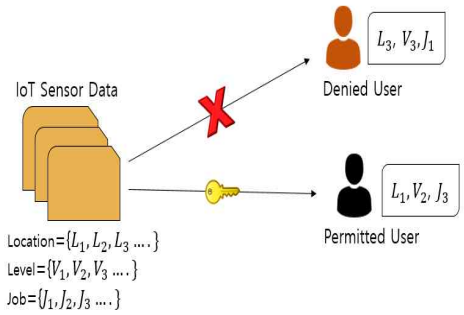
3.2 서비스 개요

제안하는 시스템은 키 발급기관 KGC(Key Generation Center), 데이터를 전송하는 IoT기기, 사용자 USer A와 UserB로 이루어져 있다. 제안하는 서비스에서 사용자들은 통신에 앞서 KGC에서 자신의 속성을 기반으로 키를 발급받아야 데이터 암, 복호화를 할수 있다고 가정한다. 이때 속성은 사용자를 식별할 수 있는 다양한 개인정보가 될 수 있다. 본 서비스에서는 사용자가 IoT기기와 데이터 통신을 할 때, 통신되는 데이터에 속성기반 암호화를 적용하였다. 사용자의 속성을 기반으로 데이터를 암호화 한 뒤, 일치하는 속성일 시에 복호화를 하는 것으로 고안하였다.



[그림 3] UserA ,UserB의 키 발급예시

위 그림은 UserA와 UserB가 암호화와 복호화를 하는 절차이다. 우선 User A와 User B는 키 발급센터인 KGC에서 통신에 사용할 키를 발급 받기 위해 자신의 속성 집합인 L_A와 L_B을 KGC로 보낸다. 그리고 KGC에서는 이 속성 집합을 기반으로 비밀키 SK_ L_A와 SK_L_B를 보내준다. 이때, 사용자가 보내는 속성은 개인의 생체정보, 실시간 위치 등 센서로 측정될 수 있는 다양한 데이터가 될 수 있다. KGC에서 발급 받은 키로 IoT 기기와 데이터를 주고받을 때, 데이터를 암호화하게 된다. 복호화 시에는 사용자의 속성 집합으로 생성하였던 비밀키를 기반으로 사용자의 속성과 일치하는 경우에 한 해, IoT의 데이터를 복호화할 수 있도록 한다. 키를 발급 받은 다음, IoT Sensor Data에 접근할 수 있다.

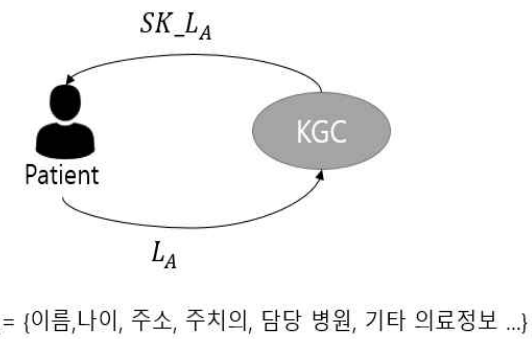


[그림 4] 권한 승인과 거절의 과정

Location	Level	Job
L_1	V_2	J_3
L_3	V_3	J_1

[표 1] 속성기반 암호화 규정

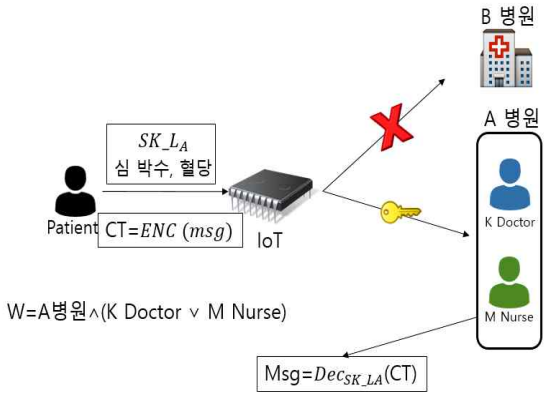
Location, Level, Job등 다양한 개인의 속성의 집합이 있을 시, 복호화 규정은 [표 1] 와 같이 구성될 수 있다. 이 규정과 일치하는 $\{L_1, V_2, J_3\}$, $\{L_3, V_3, J_1\}$의 속성을 가진 사용자만이 복호화 권한을 가질 수 있다. 따라서 [그림 4]와 같이 승인된 사용자와 거절된 사용자로 분류될 수 있다. [그림 5]는 속성기반 암호화를 헬스케어 IoT 데이터 센서에 적용하였을 때의 암호와와 복호화 예시이다.



[그림 5] 헬스케어 IoT 사용자의 키 발급 과정

우선, 자신의 이름과 나이, 주소 등 기본적인 개인정보와 주치의, 담당 병원 등 기타 의료정보를 키 발급센터에 보내 자신의 속성을 기반으로 한 비밀키 SK_{L_A}를 발급받는다. 발급 받은 키SK_{L_A}로 환자는 헬스케어 IoT 서비스에

자신의 심박수나 혈당 같은 센서로 측정되는 의료 데이터를 CT로 암호화 한다. 사용자가 사전에 정의한 데이터 복호화 규정이 W라고 가정한다면, A 병원에 한해서, K의사와 M간호사만이 이 환자의 데이터 정보를 볼 수 있다. 한편 B 병원은 데이터에 접근할 수 있는 속성을 갖고 있지 않기 때문에 권한이 없다. 복호화의 과정은 암호화된 데이터 CT를 다시 SK_{L_A}를 기반으로 복호화 하는 것이다.



[그림 6] IoT 센서 데이터의 암복호화 과정

Ⅳ. 결론

본 논문에서는 현재 사용되고 있는 IoT 장치 통신과 속성기반 암호화에 대해 알아보았다. IoT 통신은 앞으로도 계속해서 증가하고 있는 추세이지만, 이에 대한 암호화의 정책은 미비한 편이다. 이에 본 논문에서는 IoT의 센서 데이터 통신에 속성 기반 암호화를 적용하여 암호화와 복호화시에 개인의 속성을 기반으로 데이터에 접근할 수 있는 방식을 제안하였다. 속성 기반 암호화를 적용할 시, 기존의 암호화 방식을 제공할때와는 달리, 1-to-many의 방식으로 통신을 할 수 있을 뿐만 아니라, Fine-grained접근 제어를 사용할 수 있는 장점이 있다. 이는 IoT 데이터 통신의 보안성과 효율성을 동시에 높여줄 수 있는 방법을 제시하여 IoT의 활용을 더 높여줄 수 있을 것이다.

[참고문헌]

[1]"Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015,"Gartner, last modified: Nov. 11. 2014, accessed Nov. 11, 2018, https://www.gartner.com/newsroom/id/2905717

[2] "Cybersecurity Vulnerabilities Identified in St. Jude Medical's Implantable Cardiac Devices and Merlin@home Transmitter," FDA, last modified: Jan. 9. 2017, accessed Nov. 11, 2018, https://www.fda.gov/medicaldevices/safety/alertsandnotices/ucm535843.htm

[3] 박종빈, 전준걸, 최안철. "사물인터넷 장치간 End-to-End 정보 보안전송을 위한 암호화 기술 스터디와 세션 키 방식 보안기술의 설계." 대한전자공학회 학술대회, (2017.11): pp. 987-989.

[4] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-based encryption for fine-grained access control of encrypted data," Proceedings of the 13th ACM conference on Computer and communications security (CCS '06). ACM, New York, NY, USA, pp. 89-98. 2006

[5] 김연태, 김효승, 조효진, 이동훈. "속성기반 암호화 기법을 이용한 안전한 메신저 시스템," 보안공학연구논문지, 12(5), (2015.10) pp. 469-486

[6] 박민경, 조은상, 권태경. "속성 기반 암호화 방식을 이용한 다중 서버 패스워드 인증 키 교환." 한국통신학회논문지, 40.8 (2015.8): 1597-1605.

IoST 환경에 적합한 LEA 암호알고리즘 최적화 구현 및 성능분석

한주홍, 오진혁, 이재훈*, 이옥연

국민대학교 정보보안암호수학과, *국민대학교 금융정보보안학과

The Performance Analysis for the Optimized Implementation of LEA Algorithm on IoST Environment

Ju-Hong Han, Jin-Hyeok Oh, Jae-Hoon Lee*, Okyeon YI

Dept. of Information Sercurity Cryptography Mathematics, *Financial Information Security, Kookmin University

요 약

최근 IoST(Internet of Small Thing, 소물인터넷) 시장이 확대됨에 따라 마이크로프로세서를 활용한 다양한 응용서비스에 대한 보안문제점들이 대두되고 있다. 하지만 보안 문제에 대한 해결책이 있음에도 불구하고 펌웨어 환경에서는 제한된 자원을 사용해야 하므로 이를 적용하기 어려운 문제점이 있다. 기존의 보안 솔루션은 컴퓨터 또는 임베디드 시스템 환경 등에 최적화되어 있고, 이는 효율적인 자원사용이 요구되는 펌웨어에서는 적용되기 어렵다. 따라서 IoST 환경에서는 기존의 보안 솔루션을 한정된 메모리와 전력을 효율적으로 사용하는 펌웨어 환경에 맞도록 최적화 설계가 요구된다. 본 논문에서는 32-bit 마이크로프로세서인 Cortex-M4에서 경량 암호알고리즘 중 하나인 LEA(Lightweight Encryption Algorithm)를 Look-up Table 방법과 On-the-Fly 방법으로 구현할 경우 동작 속도와 메모리 사용량을 비교분석하고 소물인터넷 환경에 적합한 암호알고리즘 구현 방향을 제시하고자 한다.

I. 서론

최근 IoT(Internet of Thing, 사물인터넷) 시장이 보급되면서 Raspberry Pi, Edison, Artik과 같은 다양한 임베디드 장비들이 개발되었고, 이를 활용한 응용서비스가 개발되면서 현재 IoT 시장은 상용화 단계에 접어들어 우리 실생활에 다양하게 사용되고 있다. 이러한 기술 개발 및 시장확대는 LoRa, NB-IoT, LTE Cat.M1과 같은 저전력 기반의 무선통신기술개발과 함께 IoST 기술 개발연구가 진행되고 있다.

IoST란, 기존의 Raspberry Pi, Edison, Artik과 같은 운영체제 기반의 임베디드 시스템으로 운영되는 것이 아니라, 소형배터리를 탑재한 펌웨어 기반으로 운영되는 네트워크를 말한다. 과거에는 PAN(Personal Area Network) 크기의 네트워크 환경에서 활용되던 소형장비들이 최근 LoRa, NB-IoT, LTE Cat.M1과 같은 장거리 무선통신 기술과 접목되어 MAN(Metropolitan Area Network) 크기 환경으로 확대되면서 IoST 환경이 거론되고 있다. IoST 서비스는 날씨 정보를 활용한 응용서비스뿐만 아니라, 스마트 팜(Smart Farm), 스마트 그리드(Smart Grid), 국방 등 다양한 분야에서 활용될 수 있다. 이러한 IoST 기술은 정보의 중요도에 따라 데이터 보호가 이뤄져야 하며, 데이터 접근 및 잘못된 정보 등을 구분할 수 있는 능력이 요구된다. 하지만 IoST에 활용되는 장비들은 대부분 제한된 기능 수행과 제한된 자원을 사용해야 하는 열악한 환경이 주를 이루기 때문에 IoST 환경에 보안을 적용하기 위해서는 효율적으로 자원을 사용하는 설계를 통해 보안 적용으로 인한 본래의 서비스에 영향을 최소화해야 한다.

본 논문에서는 보안강도 128-bit 이상을 제공하는 LEA(Lightweight Encryption Algorithm) 경량암호알고리즘을 활용하고자 한다. 소형장비에 최적화된 암호알고리즘의 구조를 분석하여 MCU 환경에서 효율적으로 사용할 수 있는 방안을 제시하고 이에 대한 실험을 통해 IoST 환경에서 암호알고리즘을 구현할 경우 고려해야 할 사항 등을 서술하고자 한다.

II. 관련 연구

2.1. 국내 경량 블록 암호 알고리즘 LEA

LEA 암호알고리즘은 2013년 국내에서 개발된 암호알고리즘으로 ARX(Addition, Rotation, XOR) 구조로 설계된 경량암호알고리즘이다. 과거 ARIA, AES와 같은 복합한 행렬연산과 S-Box 치환을 사용하는 SPN(Substitution Permutation Network)과 달리 기본 연산인 덧셈과 순환 연산을 사용한 암호알고리즘이다. 이러한 구조는 행렬연산과 같은 복잡 연산으로 인해 소비되는 동작 시간을 줄여주며, S-Box 치환에서 사용되는 S-Box를 사용하지 않음으로써 메모리 사용을 효율적으로 줄이는 효과가 발생한다. 가장 우수한 것은 이러한 경량화 설계에도 같은 보안강도를 제공한다는 것이다[1].

이러한 LEA의 구조상의 특징으로 LEA는 소량의 메모리를 사용하는 펌웨어 기반의 시스템에 매우 친화적인 성향을 보인다. ARIA의 경우 4개의 S-Box가 필요하며, 각 S-Box당 256 Byte의 메모리를 사용하고, 총 1 KByte의 메모리가 필요하다. 일반적인 PC 또는 추가적인 메모리를 사용하는 임베디드 환경에서는 크지 않은 메모리 사용량이지만, 보통 32~256 Kbyte의 메모리 크기를 갖는 펌웨어 환경에서는 무시할 수 없는 크기이다. 따라서 S-Box를 사용한 치환구조를 사용하지 않는 LEA가 펌웨어 환경에 적합하다고 할 수 있다. 하지만 LEA 암호알고리즘은 SPN 구조에서 ARX 구조로 바뀌면서 내부 라운드 함수의 동작 횟수가 증가하고, 이로 인해 암호알고리즘 동작 시 생성되는 라운드키의 메모리 크기가 증가하게 된다. 128-bit 보안강도를 제공하는 ARIA의 경우 총 12라운드와 마지막 라운드에서 사용되는 128-bit 라운드 키 13개가 필요하고 총 208 Byte가 필요하지만, 128-bit 보안강도를 제공하는 LEA의 경우 총 24라운드에 사용되는 192-bit 라운드키 24개 필요하고 이는 576 Byte가 필요하다. 따라서 결론적으로 448 Byte 메모리 절감효과를 얻게 된 셈이다.

2.2. IoST 환경에서의 암호알고리즘 구현

IoST 기술은 앞서 언급되었듯이, 펌웨어 기반의 센서들로 구성된 장비로써 소량의 데이터를 무선네트워크를 활용해 필요한 정보를 송수신하는 기술이다. 이러한 IoST 환경은 제한된 전력을 사용해야 하므로 저전력 기반의 설계 구조를 가져야 하며 많은 양의 데이터를 잦은 빈도로 송수신하는 네트워크 환경과는 차이점을 보인다. 대부분 적게는 1시간에 한 번에서 많게는 1분에 한 번씩 데이터 전송을 목표로 하며, 대부분의 시간은 절전모드 상태를 유지하게 된다. 또한, 응용서비스별로 차이점은 보이나, 보통 20~100 Byte 내외의 소량 데이터를 송수신한다.

따라서 IoST 환경에서는 응용서비스에서 수집되는 데이터양과 처리량에 따라 적절한 메모리 크기를 갖고 저전력 기반으로 동작하는 MCU를 사용한다. 이러한 MCU에서는 주어진 기능을 수행하기 위해 대부분의 전력과 메모리를 사용하기 때문에 보안 적용에 따른 메모리 사용량과 전력소모량이 매우 중요한 요소로 작용한다. 따라서 적절한 보안강도를 제공하면서 IoST 서비스에 영향을 미치지 않는 구현기법이 적용되어야 한다.

일반적으로 암호알고리즘 최적화를 수행하는 경우, 메모리 사용량을 늘려서 연산 수행 시간을 단축하는 방법이 가장 많이 사용되며, Loop 문 동작에 따른 시간을 최소화하는 방법, inline 또는 메크로 함수 등을 활용하여 코드 메모리를 크게 가져가는 대신 함수 호출에 따른 지연시간을 줄이는 등의 방법을 사용하게 된다. 이러한 방식을 통해서 구현된 암호알고리즘은 현

저하게 빨라지는 효과를 볼 수 있고 이로 인해 다양한 효과를 얻을 수 있다. 하지만 이러한 구현 방식은 펌웨어 환경에서 오히려 많은 양의 자원을 낭비하게 되는 역효과로 이어진다. 또한, IoST 환경에서는 고속의 처리능력을 요구하지 않기 때문에 불필요한 자원낭비를 줄이는 방법이 필요하다.

본 논문에서는 IoST 환경에서 최적화된 암호알고리즘 구현을 위해 기존의 최적화 기술이 적용된 암호알고리즘과 On-the-Fly 방법이 적용된 암호알고리즘 간의 성능 측정을 통해 동작 시간 및 메모리 사용량을 비교 분석하였다.

III.실험 환경 및 실험 결과

3.1. 암호알고리즘 동작 환경

본 논문에서는 Cortex-M4 마이크로프로세서를 사용하여 실험을 진행하였다. [표 1]은 실험에 사용한 Cortex-M4의 동작 환경이다.

구분	내 용
CPU	Cortex-M4 120MHz(32bit)
Data Memory	SRAM 176 KB
ProgramIDE	Flash 512 KB
IDE	Atmel Studio 7.0
Compiler	ARM/GCC 6.3.1

[표 2] Cortex-M4 동작 환경

3.2. 실험 방법

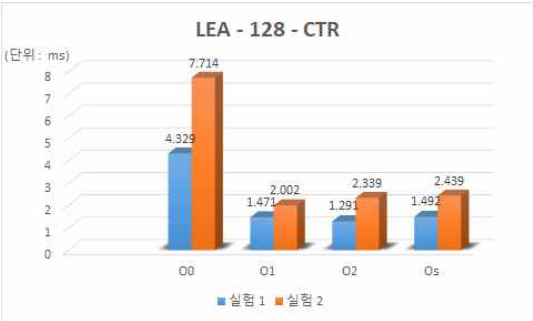
본 실험에서는 KISA(한국인터넷진흥원)에 공개된 LEA 소스코드[3]를 사용하였다. 해당 소스코드는 Look-up Table 방법과 Loop문 전개 방식으로 구현되어 있고, 이를 On-the-Fly 방법과 Loop문 병합 방식으로 구현해 실험을 진행하였다. 또한, 블록암호알고리즘 동작 시 단일 블록암호알고리즘으로 사용되지 않고 운영모드를 활용하는데, CTR 운영모드의 경우 암호화 함수와 복호화 함수 중 암호화 함수로도 데이터 암/복호화가 가능하므로 메모리가 열악한 환경에서는 CTR 운영모드가 적합하다. 따라서 본 논문에서는 LEA 128-bit CTR을 사용하였고 IoST 환경을 고려하여 최대 256 Byte를 처리하는 환경을 가정하였고, 그에 대한 처리시간을 측정하였다.

컴파일러 옵션에 따라 동작 속도와 코드 메모리, 데이터 메모리를 비교하였으며, 컴파일 옵션은 -O0, -O1, -O2, -Os의 결과를 비교 분석하였다[2].

본 논문에서는 Cortex-M4에서 LEA 암호알고리즘을 구현 및 동작의 결과를 통해 성능분석을 진행하였다. Cortex-M4 환경 특징상 오실로스코프를 활용하여 실험의 결과를 얻었다.

3.3. 실험 결과 및 분석

아래 [그림 1] 은 데이터 크기가 256Byte일 때, Look-up- Table 방법과 Loop문 전개 방식으로 구현된 코드(실험 1)을 컴파일러 옵션을 변경하여 얻은 LEA-128-CTR 처리 시간과 On-the-fly 방법과 Loop문 병합 방식으로 구현한 코드(실험 2)를 컴파일러 옵션을 변경하여 얻은 LEA-128-CTR 처리 시간을 그래프로 비교한 그림이다.



[그림 1] 컴파일 옵션 별 LEA-128-CTR 처리 시간

아래 [그림 2]는 실험 1과 실험 2의 코드 사이즈를 비교한 그래프이다.

[그림 2] 컴파일 옵션 별 LEA-128-CTR 코드사이즈

	Look-up Table (Bytes)	On-the-Fly (Bytes)
Delta	384(96*4)	16(4*4)
라운드 키	576(6*24*4)	32(8*4)
총합	960	48

[표 3] 데이터 메모리 비교

[표 4]는 Look-up Table과 On-the-Fly 각각에서 사용한 데이터 메모리 사용량을 비교한 표이다. Look-up Table 방법에서는 라운드키 생성 시 Word 크기의 미리 계산된 96개 Delta 테이블을 사용하고, 192-bit 라운드 키 24개를 사용한다. On-the-Fly 방법의 경우 Word 크기의 4개 Delta 값을 사용하여 각 라운드에서 라운드키를 계산하고 이때 8개의 임시변수를 사용한다. 따라서 Look-up Table 방법과 On-the-Fly 방법에서 사용된 데이터 메모리 크기를 비교하면 95%의 메모리 효율을 갖는다.

본 실험 결과를 코드 사이즈 및 데이터 메모리 관점에서 분석하기 위해 컴파일 옵션 -Os를 기준으로 분석한 결과 Look-up Table 방법을 사용한 실험 1에서는 LEA-128-CTR 처리 시간이 1.492ms 이었고, 이 경우 5664 Bytes의 코드 메모리를 사용하고 960 Bytes의 데이터 메모리를 사용한다. 반면 On-the-Fly 방법을 사용한 실험 2에서는 LEA-128-CTR 처리 시간이2.439ms로 실험 1에서 사용한 Look-up Table 방법보다 약 1.6배 느린 속도를 보이지만 2439 Bytes의 코드 메모리를 사용하고 48 Bytes의 데이터 메모리를 사용하여 3225 Bytes의 코드 메모리 절감과 912 Bytes의 데이터 메모리 절감효과를 보았다.

IoST 환경에서 1분에 1회 잦은 빈도로 송수신하는 운영환경을 고려해 봤을 때, 0.947ms 처리 시간 차이는 매 분 약 0.0016%의 영향을 미치는 것으로 큰 영향을 준다고 보기는 어렵다. 전력소모량은 처리시간에 비례하기 때문에 전력소모량에도 큰 영향을 준다고 보기는 어렵다.

하지만 메모리 사용량의 경우, 응용서비스가 사용하지 않는 영역에 암호알고리즘이 구현되는 것을 고려한다면 코드 메모리 약 3.2 Kbyte와 데이터 메모리 약 1 Kbyte의 절감효과는 매우 크다.

IV. 결론

펌웨어기반의 IoST환경은 임베디드 시스템을 활용한 IoT 환경보다 더욱 열악하고 제한된 조건에서 동작한다. 이러한 환경에도 불구하고 보안에 대한 문제제기가 되고 있으며 이를 해결하기 위한 다양한 연구가 진행되고 있다. 하지만 기존에 서버 또는 PC, 임베디드 시스템에서 활용되던 최적화 암호알고리즘 코드를 펌웨어 기반의 장비에 그대로 사용하는 것은 오히려 제한된 전력과 메모리를 낭비하는 결과를 초래할 수 있다. 따라서 IoST 환경에 대한 충분한 이해를 기반으로 펌웨어 환경에서 동작할 수 있는 암호알고리즘 최적화 구현은 다른 연구 방향이 제시되어야 한다.

본 논문에서 확인 할 수 있듯이, On-the-Fly 방법을 사용한 LEA 암호알고리즘은 Look-up Table 방법보다 0.947ms 느리지만 코드 메모리와 데이터 메모리에서 각각 3.2 Kbyte, 1 Kbyte의 절감효과를 확인할 수 있었다. 향후 효율적인 메모리 사용과 성능에 대한 비교분석 실험 등을 통해 IoST 환경에서의 암호알고리즘 최적화 구현 방안에 관한 연구를 진행할 예정이다.

[참고문헌]

[1] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, "LEA: A 128-bit block cipher for fast encryption on common processors," In Information Security Applications, WISA 2013, pp. 3-27, 2013.

[2] Hyunjung An, Seunghwan Yun, Okyeon Yi, "Performance Evaluation of Firmware Cryptographic Modules on Various MCU Environments", PlatCon 2016.

[3] Taeyean Kwon, Jaehoon Lee, Hyunduk Choi, Okyeon Yi, Seongho Ju, "Efficiency of LEA compared with AES", DFIS 2014.

2018년 정보보호학술논문발표회

사물인터넷 디바이스로부터의 서브도메인 디도스 공격을 재귀적 네임서버에서 방어하기

서양진*

*(주)코트캔디

tim@cottoncandy.co.kr

Mitigating Random Subdomain DDoS Attacks from IoT Devices at Recursive DNS Servers

Yangjin Seo*

*CottonCandy Co., Ltd.

Abstract

In this paper, we focus on random subdomain DDoS attacks from IoT devices. To respond to queries from many defected IoT devices, recursive DNS servers send a huge volume of DNS queries to authoritative DNS servers, and it will result in a severe failure or degradation of DNS infrastructure. We suggest a mitigation method to deal with the attack at recursive DNS servers which locates between defected IoT devices and authoritative DNS servers. The method includes a multi-stage filter and classifier to distinguish legitimate subdomain names from malicious ones. Our experiment shows that this approach can be an effective and efficient countermeasure against random subdomain DDoS attacks from IoT devices.

I. Introduction

It seems that DDoS attacks recently got stronger as shown in the third quarter 2016 report of Akamai [1]. Concerning such a trend, the report explains that IoT devices played an important role, which means that they unintentionally helped the success of DDoS attackers. Indeed, a series of DDoS attacks in the second half of 2016 shows that a severe DDoS attack from IoT devices is a reality, not a possibility.

In this paper, we want to address one of the incidents, which is random subdomain DDoS attacks from IoT devices, and how we can handle the challenge effectively. In section 2, we will explain random subdomain DDoS attacks. Section 3 will give related works and evaluation on them, and in section 4, we will explain our method to protect a DNS infrastructure from random subdomain DDoS attacks from the IoT devices. We conclude with an evaluation in section 5.

II. Random Subdomain DDoS Attacks

In October 2016, Dyn, a DNS host and Internet management company, experienced three DDoS attacks on their managed DNS servers. The attacks resulted in unavailable services of many companies. According to some reports, the attack magnitude reached in the 1.2 Tbps range. The attacks against Dyn were a high-volume flood of TCP and UDP packets to DNS port 53 from 10s of millions of IP addresses. But, Dyn identified that only 100,000 out of those IP addresses were malicious endpoints infected by Mirai

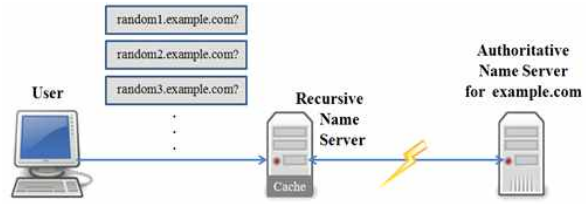


Figure 1 Random Subdomain Attack Scenario

malware. Dale Drew, CSO of Level 3, analyzed that the attack also included a random subdomain attack, one of attack vectors Mirai has in its armory.

The mechanism of random subdomain attack is shown in Figure 1. For an architecture using a cache, a cache miss will cause an expensive fetch operation. If there are many DNS queries with non-existent domain names whether they are mere mistakes or not, it will give a big burden on both the recursive name server and the authoritative name server. When there are a huge number of such queries, it will reach beyond what the authoritative server can bear. As a result, the server will fall into an unresponsive status for a while or for a long time, and the status would get worse because users or recursive name servers will send their queries again and again. That is what exactly happened when 100,000 IoT devices mobilized from all over the world simultaneously sent a huge number of DNS queries having domain names with randomly generated prefix like ax1kgpov.example.com.

III. Related Work

From the viewpoint of recursive name severs, it is hard to find out that they are exploited by DDoS attackers for random subdomain attacks from IoT devices. To resolve the issue effectively, we need a method to distinguish legitimate DNS queries from the queries having a random subdomain name at recursive name servers that they can block the malicious requests. Namely, we need a mechanism that can find out whether a domain name in a DNS query has a randomly generated prefix constituting a non-existent domain. In so doing, what makes the problem difficult is that not all randomly generated prefixes are for a malicious purpose. Many companies use automatically generated domain names for their own specific, legitimate purposes, which are called disposable domains [2] as shown in Figure 2. It means that it is not enough just to identify a randomly generated prefix. What we need is to know not only whether a prefix is randomly generated but also whether it is a malicious one.

d1jdgm35warl9f.cloudfront.net,
d2d735512y8kb.cloudfront.net,
d2lv4zbk7v5f93.cloudfront.net,

r10---sn-o097znez.googlevideo.com,
r6---sn-n4v7sn7l.googlevideo.com

Figure 2 Disposable Domain Name Example

In [2], Afek, Bremler-Barr, Cohen, Feibish, and Shagam suggest a mechanism to identify distinct heavy hitters for random subdomain attacks detection. During peacetime, it builds a whitelist that will include information of which domain/zone uses many distinct subdomain names and what prefix instances are used for the domain, and with the whitelist, it will be able to identify malicious queries having an automatically generated subdomain. In [3], Takeuchi, Yoshida, Kobayashi, Kato, and Kishimoto suggest a mechanism that can handle a feature in randomly generated subdomain names using a Naïve Bayes classifier [4]. They use four attributes for the classifier: the number of occurrences of same prefix for a given period (window), the length of prefix, the number of labels in the prefix, and bigrams made from

the prefix. Their evaluation results show that the mechanism can correctly detect 94.57% with the false positive of 1.14% and the false negative of 4.28%. Here the false positive is a case that a legitimate query is predicted as a malicious one, and the false negative is a case that a malicious query is predicted as a legitimate one. The two numbers mean that out of 10,000 legitimate queries, 114 will not be answered, and out of 10,000 attack queries, 428 will be forwarded to authoritative name severs. If malicious queries forwarded to authoritative name severs could be reduced at that number, it would be labeled as a successful detection and mitigation. But, the truth is that this mechanism does not distinguish randomly generated legitimate subdomain names from randomly generated malicious subdomain names. We want to address this limitation with our multi-stage filter including a classifier per domain/zone.

IV. Multi-stage Filter Identifying Legitimate Queries

Figure 3 is the entire architecture of the mechanism we suggest. Our mechanism is activated when the recursive name sever receives a query from a user and the domain name is not in the cache. Our Multi-Stage Filter (hereafter MSF) checks whether the zone (domain) is one using disposable domain names, automatically generated legitimate domain names. If it is not, MSF will check whether within a timeframe designated, this zone had the total number of queries having a distinct name more than the threshold for rate limiting. If that is the case, the query will be answered with a response having the value of non-existent domain. MSF will collect the necessary information to decide which zone uses disposable domain names from queries history. This task will be done periodically during peaceful and idle time. MSF will categorize a zone as one using disposable domain names if it used distinct domain names more than a designated value for past period, for example 1,000 distinct names for 24 hours. During the process, MSF will build necessary information of regular names, which are considered as the names generated by humans, disposable name patterns and each pattern's features of the number of labels and length for prefix. By using those simple features, we can filter out most of malicious domain names, randomly generated domain names, and the queries still on the process will be filtered by the Naïve Bayes classifier built during collecting information. Even if there is a new disposable name pattern, it will have a strong possibility being answered normally unless there were already distinct unknown domain names for the zone within the time frame designated.

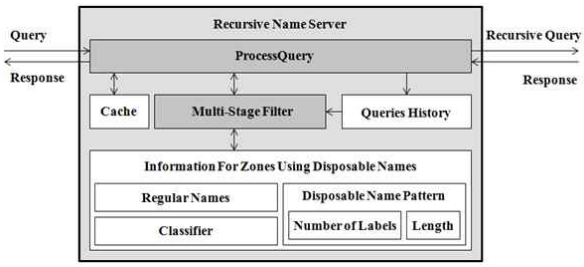


Figure 3 Entire Architecture of Multi-Stage Filter for Random Subdomain DDoS Attacks

For the Naïve Bayes classifier, we used three features of the number of labels, length, and bigrams for the prefix of a domain name and adopted the attribute definition from [3]. There were 4 test scenarios. The scenario 2 is according to [3] and our main reference model. In this experiment, a Naïve Bayes classifier was built from all instances of

		Predicted	
		Malicious	Legitimate
Actual	Malicious	987 (73.71%)	13 (0.97%)
	Legitimate	66 (4.94%)	273 (20.39%)
Accuracy: 94.1 % Error Rate: 5.9 %			

(a) Confusion Matrix for Scenario 1

		Predicted	
		Malicious	Legitimate
Actual	Malicious	989 (73.86%)	11 (0.82%)
	Legitimate	10 (0.78%)	329 (24.57%)
Accuracy: 98.4 % Error Rate: 1.6 %			

(b) Confusion Matrix for Scenario 2

		Predicted	
		Malicious	Legitimate
Actual	Malicious	2000 (97.8%)	0 (0%)
	Legitimate	0 (0%)	45 (2.2%)
Accuracy: 100 % Error Rate: 0 %			

(c) Confusion Matrix for Scenario 3

		Predicted	
		Malicious	Legitimate
Actual	Malicious	200 (81.6%)	0 (0%)
	Legitimate	0 (0%)	45 (18.4%)
Accuracy: 100 % Error Rate: 0 %			

(d) Confusion Matrix for Scenario 4

Figure 4 Confusion Matrix

legitimate and malicious prefixes and no separation between zones. In the case of scenario 1, we used Shannon Entropy instead of bigrams to check whether using bigrams to identify randomly generated prefixes is a better choice. The scenario 3 was the case to apply our mechanism. Because we did not have abundant instances that could give disposable name pattern per zone, we just include total two patterns from two different zones. In the scenario 3, compared to the number of instances for legitimate prefixes, the number of malicious prefixes was big. So, we reduced the instances for malicious ones and did an additional test in the scenario 4. We applied 10-fold cross validation for all cases and evaluated each case with a confusion matrix, and they are given in Figure 4.

V. Evaluation and Conclusion

In this paper, we suggest a mechanism to mitigate the random subdomain attacks. The core of our suggestion is to build a Naïve Bayes classifier for each zone using disposable domain names at recursive DNS severs and with the classifier to distinguish automatically generated legitimate domain names from randomly generated malicious domain names. From the overall test results, we conclude that our multi-stage filter is an effective and efficient countermeasure against random subdomain attacks from IoT devices.

[참고문헌]

[1] McKeay, Martin. "Akamai's State of the Internet Security Q3 2016," 2016.

[2] Chen, Yizheng, Manos Antonakakis, Roberto Perdisci, Yacin Nadji, David Dagon, and Wenke Lee. "DNS noise: Measuring the pervasiveness of disposable domains in modern DNS traffic." In Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on, pp. 598-609. IEEE, 2014.

[3] Takeuchi, Yuya, Takuro Yoshida, Ryotaro Kobayashi, Masahiko Kato, and Hiroyuki Kishimoto. "Detection of the DNS Water Torture Attack by Analyzing Features of the Subdomain Name." Journal of Information Processing 24, no. 5 (2016): 793-801.

[4] Witten, Ian H. and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2005.

Z-Wave 보안 취약점 분석

임지환[1] · 조기윤[1] · 성민석[1] · 정영호[1] · 김성범[1] · 김경곤[1] · 이경문[1] · 서화정[2]

[1]한국정보기술연구원 차세대 보안리더 양성 프로그램

[2]한성대학교 IT융합공학부

Ji-Hwan Lim[1] · Ki-Yoon Cho[1] · Min-Suk Sung[1] · Sung-Bum Kim[1] · Young-Ho Jung[1] · Anesra[1] · GilGil[1] · Hwa-Jeong Seo[2]

[1]Best of the Best

[2]Department of IT, Hansung University, Seoul, 02876 Korea

요 약

현재 IoT 기기들은 각각의 기능과 필요에 맞는 무선 통신을 위하여 Wifi, BLE, Z-Wave, Zigbee, NFC 등의 여러 프로토콜을 사용한다. 본 논문은 그중 최근 홈 IoT 시장에서 주목받고 있는 사용성과 확장성이 좋고 저전력 프로토콜인 Z-Wave 무선 통신 프로토콜을 사용하는 제품들의 보안 취약점을 기술한다. 본 논문은 Z-Wave에 대한 관련 연구 및 동향을 소개하며 직접 분석한 국내 제품들에 대한 보안관점의 문제점을 간략히 나타내고, 몇 가지 공격 형태에 대해 설명한다. 또한, Z-Wave 프로토콜에 대해 분석하여 어떤 데이터를 주고받는지 기술한다. 본 논문을 통해 무선 사물인터넷 시장에 대한 보안성을 높일 수 있는 방안을 연구하는 좋은 출발점이 되었으면 한다.

I. 서론

Z-Wave는 통신, 네트워크 및 응용 계층 프로토콜을 포함하는 IoT 최적화 기반의 구현이다. 사용자는 Z-Wave 센서, Actuator, 컨트롤러, 라우터 및 인터넷 게이트웨이를 구성하여 스마트 홈 및 사무 자동화 시스템을 구축할 수 있다. 스마트 홈 자동화 시스템은 난방, 환기 및 에어컨, 조명 및 물리적 보안 시스템을 위한 중앙 집중식 제어 및 모니터링 기능을 제공한다. 이러한 중앙제어 시스템, 보안 센서 및 경보 시스템과 같은 다양한 가정 장치는 무선 또는 유선 통신 링크를 통해 한 개의 컨트롤러 장치에 최대 232개의 노드가 있는 Mesh 네트워크를 형성하고 이를 통해 스마트 홈을 형성한다.

Z-Wave의 프로토콜 계층은 물리, 전송, 라우팅, 어플리케이션 크게 4가지 계층으로 구성되고, 통신 기반은 물리, 맥 계층이 ITU.G.9959에 정의되어 있는 독점 무선 스택을 사용한다. 또한, 애플리케이션 계층의 상당 부분이 OpenZwave [1]를 통하여 오픈 소스코드로 공개 되어있다.

스마트 홈 및 사무공정 자동화 시스템을 위해 최적화되어 효과적으로 운용되는 시스템에도 불구하고, 타 프로토콜과 마찬가지로 공격자가 무선 패킷을 쉽게 Sniffing 하거나, Spoofing 할 수 있다는 단점이 있고, 이러한 시스템을 지속시키기 위해서는 무선 통신의 기밀성과 무결성을 보호하는 것이 매우 중요하다. 따라서 키 설정, 암호화 방식, 프레임 무결성 보호 및 장치 인증과 같은 보안 서비스가 사용되고 있다. 따라서 본 논문에서 Z-Wave 사용 보안성 강화를 위해 국내에서 사용하는 제품의 취약점을 분석한다.

본 논문의 구성은 다음과 같다. 2장 관련 연구 동향에서 Z-Wave 프로토콜에 대해 분석하고, 3장 본론에서 Z-Wave 사용 취약점에 대해 분석한다. 그리고 4장 성능평가에서 DFD를 기

반으로 몇 가지 공격벡터를 기술하고, 마지막 5장에서 본 논문의 결론을 맺는다.

II. 관련 연구 동향

2.1 Z-Wave 프로토콜 분석

모든 Z-Wave PAN (Personal Area Network)에는 Z-Wave 컨트롤러에 내장되어있는 고유의 Home-ID가 있다. 이 Home-ID는 제조 시에 설정되고 유저에 의해서 변경될 수 없도록 설계되어있다. 이렇게 하는 이유는 특정 Home-ID를 갖는 컨트롤러가 인접 Z-Wave 네트워크를 침범하지 않기 위해서 이다. 따라서 Z-Wave PAN 내부의 모든 노드는 컨트롤러와 동일한 Home-ID와 Node-ID가 조합되어 고유하게 식별된다. 만약 서로의 범위 내에 여러 Z-Wave 컨트롤러가 작동한다면 각각의 장치는 자신이 속한 컨트롤러의 Home-ID에 의해 식별될 수 있을 것이다. Z-Wave의 기본 프레임 구조는 그림 1과 같다. Z-Wave MAC 프레임의 기본 구조는 아래와 같이 P, SOF, User-Data, EOF로 구성된다. 또한, Broadcast, Multicast, 라우팅과 탐색기 프레임은 User-Data 섹션 내부의 Home-ID, Src, Dst 등 다른 필드를 사용한다. Application Data는 실질적으로 사용자가 전달하고자 하는 Z-Wave Comand Class 명령 또는 메시지 전달에 사용된다. Application Data는 AES-128로 암호화 되어 통신한다 [2].

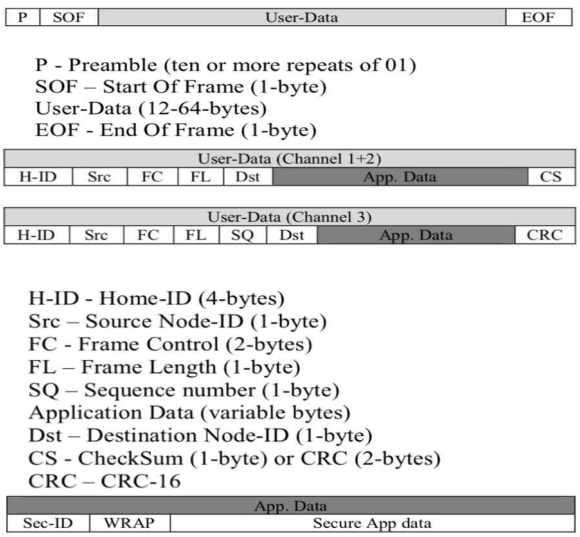


Fig. 1 Z-Wave packet frame

2.2 Z-Wave 보안 프로토콜 등급

Z-Wave의 초기의 100 시리즈는 TDES (Triple Data Encryption Standard)가 있었고, 암호화는 항상 Vendors에게 맡겼다. 모든 Vendors가 암호화 옵션을 사용하지는 않았기 때문에, Z-Wave 장치의 취약점을 보여주는 연구가 지속적으로 있어왔다. 따라서 Z-Wave는 S2 보안 프레임워크로 인증을 대체할 것을 알렸다. S2 보안 프레임워크는 데이터링크용 AES-128과 키 교환을 위해 ECDH를 기반으로 한다. 간단히 정의하면 Z-Wave S2 보안 프레임워크는 기존의 S0 방식의 키 교환 부분의 취약한 부분을 ECDH 암호화를 적용하여 문제를 해결한 것으로 볼 수 있다. Z-Wave S0 키 교환 및 통신 과정은 그림 2와 같고, 사용되는 암호화 방식은 표 1과 같다.

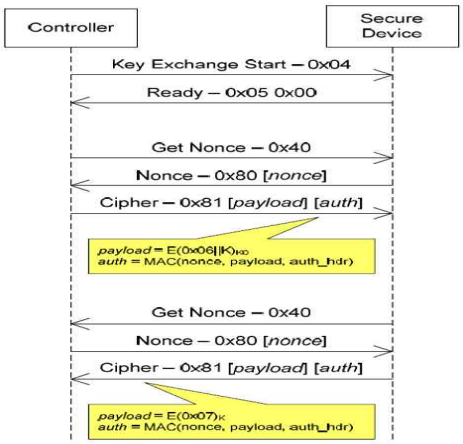


Fig. 2 Key exchange protocol (S0) [3]

Table. 1 Encryption method used in Z-Wave S0 security

$$K_c = AES-ECB_{k_n}(Passwd_c)$$

$$K_m = AES-ECB_{k_n}(Passwd_m)$$

$$C = AES-OFB_{k_c}(IV, P)$$

$$MAC = AES-CBCMAC_{k_m}(IV, SH \| SRC \| DST \| LEN \| C)$$

III. 본론

본 장에서는 국내 Z-Wave 제품의 보안성 검사와 Z-Wave 보안 취약성 부분을 기술한다. 우선 테스트를 위한 장비는 Silicon Labs에서 판매하는 Dev Kit [4]을 구매하였다. Z-Wave

무선 패킷을 분석하기 위한 EU (868MHz), US (908MHz), Asia (920MHz) 대역의 패킷을 Sniffing할 수 있는 동글 (Zniffer)은 Dev Kit에서만 구할 수 있다. Zniffer를 통하여 작동하고 있는 무선 패킷을 Wireshark와 유사한 Slicon Labs에서 제공해주는 App을 통해서 분석할 수 있다.

3.1 국내 Z-Wave 제품 보안성 검사

국내 Z-Wave 제품의 취약점 분석 및 테스트를 수행하기 위해 망 구성도를 그림 3과 같이 구성하였다.

Fig. 3 Overall network configuration for analyzed products

분석 결과 국내 Z-Wave 제품들은 대부분 Non Secure 모드 이거나, S0 보안 방식을 채택하여 취약한 상태임을 확인하였다. 2016년 7월 Sigma Designs에서 Z-Wave S2 보안 방식을 공표 [5]하였지만, 아직까지는 국내에서는 S2 보안 방식을 사용하는 컨트롤러는 구하기 힘든 상황이다.

공격 시연 테스트는 현재까지는 2가지 방식으로 진행하였다. Sigma Designs에서 제공해주는 Z-Wave PC Controller 5 App을 변조하여 Spoofing 툴로 사용하였다. 우선 공격을 수행하기 위해서 필요한 정보인 Home-ID를 Sniffing을 통해 탈취한 후, 해커가 자신의 Z-Wave 컨트롤러의 Home-ID를 Target Home-ID로 변조하고, 나머지 패킷을 수정하여 명령을 전송하였다. 그 결과 성공적으로 공격이 수행되는 것을 확인하였다.

사용자가 보내는 Z-Wave 패킷은 그림 1의 User-Data 부분이다. 해당 공격 방식이 의미있는 이유는 Asia 대역에 아직까지는 공격 툴이 없기 때문이다. 기존의 공격은 Z-Force Attack을 통하여 수행할 수 있었지만, 해당 공격은 EU, US 대역에서만 사용할 수 있다. 따라서 국내 대역에서 Z-Wave 공격을 성공하기 위해서 오픈 소스를 참조하여 Non-Secure, S0, S2까지 공격할 수 있는 새로운 GUI Type Spoofing Tool을 구현하였다. HackRF를 사용하는 공격 방식으로는 Replay Attack을 시연하였다. 해당 방법은 현재까지는 Z-Wave Non-Secure Mode에서만 적용한 상태이다.

3.2 Z-Wave 보안 취약성

국내 Z-Wave를 사용하는 Node 제품에 임베디드 장비 취약점을 활용한 공격으로 그림 4와 같은 방식으로 Flash Memory에 있는 펌웨어를 추출해본 결과 일부는 Flash Memory에 접근할 수 있는 Uart, SPI 통신 디버깅 포인트를 사전에 끊어놓은 상태로 제품을 출시하여 접근할 수 없었다. 하지만, 일부는 접근이 허용되는 부분이 발견되어 Z-Wave 보안등급 S0 기준으로 초기에 설정되어있는 $Passwd_c$,, $Passwd_m$를 탈취하였다. S0 방식은 공개키 암호화 방식이 아닌 대칭키 암호화 방식이므로 공격자가 페어링 시 Sniffing으로 정보를 탈취하거나, 강제 재 페어링 공격에 성공한다면 PAN에서 모든 노드를 공격자가 원하는 대로 제어할 수 있는 취약점이 있다.

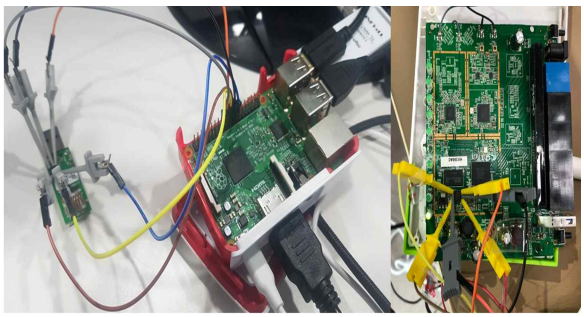


Fig. 4 Exploitation of firmware extraction on embedded equipment

DFD는 분석 대상 시스템의 데이터 흐름을 추상적으로 보여주는 방법으로 Z-Wave S0 보안 프로토콜에 대해 그림 5와 같이 간략히 나타내었다. 이에 대해 설명하면 과정 1은 사용자

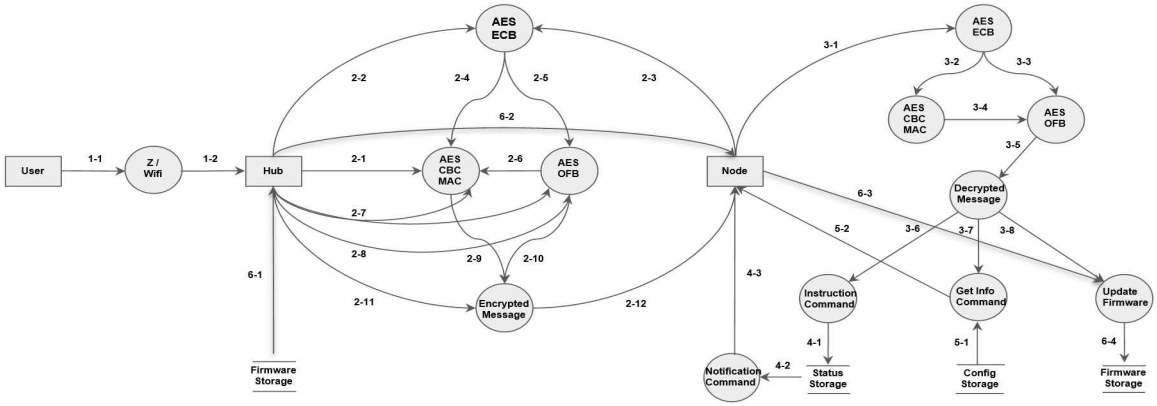


Fig. 5 Z-Wave S0 Data Flow Diagram (DFD)

와 Hub 간의 통신이고, 과정 2, 3은 허브와 노드간의 통신 데이터 송수신 과정에 사용되는 암호화 방식이다. 과정 4, 5, 6은 Controller에서 받은 명령을 노드에서 처리하는 방식이다. DFD 도식화 과정을 거치고 난 뒤, 취약점을 식별해 보니 명령 데이터를 보내는 부분에서 취약점이 주로 발생하는 것으로 확인하였다.

IV. 성능평가

본 장에서는 식별된 위협요소에 대해서 STRIDE 분석 방법을 사용하였다. STRIDE 분석은 그림 5의 DFD 기반으로 그림 6과 같으며, 그 중 Sniffing을 통한 중요 데이터 탈취와 공격자가 생성한 임의의 Frame을 통한 Spoofing과 Firmware 변조 등의 위협을 최종적으로 식별하였다.

STRIDE	Explain
D	Jamming을 통해 Transfer Presentaion 프레임 전송을 방해하여 페어링 과정을 방해
S, T	노드가 전송하는 프레임의 NodeID 및 Command를 수정하여 다른 장비로 속인다.
S, T	AssignID를 수정한 Fake 프레임을 보내 강제 페어링
T	HomeID를 변조시킨 Fake AssignID 프레임을 보내 강제로 연결 해지
D	Find nodes in range 프레임을 지속적으로 보내어 명령 처리 대기 큐를 형성시켜 각 노드들에 대한 서비스 지연
T	모든 노드들의 Neighbor List 정보를 가진 컨트롤러의 Neighbor List 에 Fake 프레임을 보내 공격자가 변경한 정보 (노드 Neigbor List)로 변경
S	노드, 컨트롤러에 대한 인증이 Home ID 밖에 없기 때문에 공격자가 Nonce 를 정상적으로 요청
S, T, D	Jamming 또는 DoS 공격 후 공격자가 Fake Nonce Report 프레임을 각 노드에게 전달하여 공격자의 명령을 수행한다 .
I	Nonce Report 를 통해 전달되는 Nonce 값을 스니핑 하여 공격자가 정상적인 프레임을 생성할 때 사용
I	암호화에 사용하는 노드 장비 펌웨어 내 하드코딩 된 Passwc , Passwd 값을 공격자가 탈취 가능, 허브와 노드 간 패킷의 중요 정보 (HomdID, Command Class, Command, NodeID)를 Sniffing 하여 정상적인 암호문 및 인증 값을 생성
S	Firmware Updata Command Class를 이용해 각 노드를 악성코드에 감염
S	정상 프레임들을 Sniffing 한 후 Replay 패킷을 보내 노드 강제 제어

Fig. 6 Classification of identified threats using STRIDE model

V. 결론

본 논문에서는 Z-Wave 보안 프로토콜 Non-Secure, S0에 대해 취약점을 분석하고, 실제 환경을 구축하여 테스트하였다. S0 보안 프로토콜에서는 공정과정에서 HW에 셋팅되어 있는 $Passwd_c$,, $Passwd_m$와 Sniffing을 통하여 원격 강제 제어가 가능한 상황이다. 또한, 본 논문에서 S0 프로토콜의 일부 프로세스에 대한 간략한 DFD를 나타내었으며, 차후 완성한 DFD를 바탕으로 STRIDE를 도출하여 Attack Tree를 구성할 것 이다. 추가적으로 프로토콜에 최적화된 새로운 STRIDE 모델까지 도출할 예정이다. 최종 향후 수행 목록으로 직접 구현한 Tool에 MITM, ARP Spoofing의 기술을 추가하여 단일 Tool 공격의 완성도를 높일 예정이다.

[Reference]

[1] Github. OpenZWave [Internet]. Available: https://github.com/OpenZWave.

[2] Badenhop, C. W., et al. "The Z-Wave routing protocol and its security implications." *Computers & Security*, vol. 68, pp. 112-129, Apr, 2017.

[3] Fouladi, B, and Sahand Ghanoun. "Security evaluation of the Z-Wave wireless protocol." *Black hat USA*, vol. 24, pp. 1-2, 2013.

[4] Digi-Key. Slicon Labs Dev Kit [Internet]. Available: http://bitly.kr/hdWS.

[5] ZWave Alliance. Z-Wave public specification [Internet]. Available: https://z-wavealliance.org/z-wave-public-specification/.

로봇청소기 제품군 Attack Surface 분석 방법

임준태, 석지원, 최원빈, 이재휴, 이경수, 안용호, 고영우, 이인형, 이상섭, 이원
한국정보기술연구원 차세대 보안리더 양성 프로그램(Best of the Best)

Robotic Vacuum Attack Surface Analysis Method

Jun Tae Im, Ji Won Seok, Won Bin Choi, Jae Hyu Lee, Gyeong Su Lee, Yong Ho An, Young Woo Go, In Hyeong Lee, Sang Sup Lee, Won Lee
KITRI 차세대 보안리더 양성 프로그램(Best of the Best)

요 약

최근 로봇청소기와 IoT 제품의 판매량이 증가함에 따라 매핑 기능, Wi-Fi 통신, 홈 뷰 기능 등 다양한 기능을 탑재한 로봇청소기들이 출시되고 가정에 보급되면서 이에 따른 공격벡터가 증가하였다. 로봇청소기 제품군에서 발생할 수 있는 보안 문제에 대한 Attack Surface를 기능 분석을 통해 청소기와 앱 간의 블루투스 통신, 청소기와 앱 간의 Wi-Fi 통신, 악성 앱 업데이트, 펌웨어 추출 및 기능 분석으로 나누어서 이에 따른 로봇청소기 Attack Surface 분석 방법을 제시한다.

I. 서론

최근 로봇청소기와 IoT 제품의 판매량이 증가함에 따라 매핑 기능, Wi-Fi 통신, 홈 뷰 기능 등 다양한 기능을 탑재한 로봇청소기들이 출시되면서 이에 따른 공격벡터가 증가하였다. 그리고 최근 카메라가 장착된 로봇청소기가 출시되기 시작하면서 해킹 공격으로 카메라를 장악할 수 있다면, 사생활 침해의 위험성도 높아진다. 따라서 로봇청소기 제품군의 보안성 향상에 대한 필요성이 증가하였다.

본 논문에서는 로봇청소기 제품군에서 발생할 수 있는 보안 문제에 대한 Attack Surface를 기능 분석을 통해 보안 문제가 발생할 수 있는 청소기와 앱 간의 블루투스 통신, Wi-Fi 통신, 악성 앱 업데이트, 펌웨어 추출 및 기능 분석으로 나누어서 이에 따른 로봇청소기 Attack Surface 분석 방법을 제시한다.

로봇청소기는 카메라 모듈을 탑재하고 있고 시장 점유율과 소비자 인지도가 높은 회사 5곳의 제품들을 분석 대상으로 선정하고 진행하였다.

II. 청소기와 앱 간의 블루투스 통신

[그림 1]은 블루투스 기능이 있는 로봇청소기의 통신 방식에 대한 구조도이다.

[그림 1] 로봇청소기 블루투스 통신 방식

블루투스 기능을 탑재한 로봇청소기 같은 경우, 블루투스 기반의 컨트롤 앱과 로봇청소기 간의 통신이 이루어지는 것을 파악을 하였고 앱 동작이 어떻게 이루어지는지 파악하기 위해서 앱 분석을 진행하였다.

그리고 [그림 2]와 같이 Frida를 이용하여 블루투스 통신 부분을 후킹해 블루투스 데이터를 확인할 수 있다.

```
[*] Connect... OK
[+] Execute script
[*] Bluetooth Hooked!
[*] [-86, 3, 1, 0, 10, -20]
[*] [-86, 3, 1, 0, 10, -20]
[*] [-86, 3, 1, 0, 11, -20]
[*] [-86, 3, 1, 0, 11, -20]
```

[그림 2] Frida 후킹한 블루투스 데이터

그리고 안드로이드의 블루투스 HCI 스누프 로그를 이용하여 앱과 청소기 간의 블루투스 패킷을 스니핑하고 Wireshark를 이용하여 앱 업데이트 및 버전 체크, 특정 URL 접속, 청소기 제어 블루투스 데이터 등의 패킷을 확인할 수 있다.

로봇청소기의 블루투스 기능을 분석한 결과 블루투스 명령 패킷은 [그림 3]처럼 0xAA로 시작하여 0xEC로 끝나는 것을 알 수 있다. 그리고 올바르지 않은 CMD 코드를 전송할 경우 조작이 불가능해지는 취약점이 발생하는 것을 확인할 수 있다.

구분	Start	CMD				End
전원	0xAA	0x04	0x01	0x00	0x07	0xEC
전원	0xAA	0x03	0x07		0x01	0xEC

[그림 3] 블루투스 명령 패킷

Ⅲ. 청소기와 앱 간의 Wi-Fi 통신

[그림 4]는 Wi-Fi 기능이 있는 로봇청소기의 통신 방식을 나타낸 구조도이다.

[그림 4] 로봇청소기 Wi-Fi 통신 방식

Wi-Fi 통신을 하는 로봇청소기 같은 경우 초기 등록 부분에서는 앱과 직접적인 통신이 이루어지고 그 이후부터는 클라우드 서버를 거쳐서 통신 한다는 것을 파악하였다. 그리고 서버와의 통신을 확인해본 결과, 평문이 아닌 SSL 통신을 하는 것을 확인했다.

앱 분석을 시작했을 때, 앱 상에서의 난독화나 루팅 탐지 등이 적용되어있는지 확인을 해본 결과, 분석을 진행한 로봇청소기 앱에서는 로컬에서 루팅을 탐지하는 루틴이 발견되었다. 이를 우회하기 위해 앱을 디패키징 하여 Smali 코드를 수정하고, 실시간으로 Frida를 이용한 동적 디버깅을 통해 루팅 탐지 로그를 확인할 수 있다.

JNI로 so파일 열어서 [그림 5]처럼 바이너리 패치를 하여 루팅 탐지 루틴을 우회를 할 수 있었다.

```
if ( v6[10] && !IsDebugMode(v8, v5) )
{
  v9 = 4;
  goto LABEL_7;
}
_android_log_print(3, "SecurityCheckerNat
_android_log_print(3, "SecurityCheckerNat
if ( v6[9] && !IsTampered(v6, v8, v5) )
```

[그림 5] 바이너리 패치를 수행한 네이티브 코드 일부

디패키징 한 앱 내부에서 피닝이 되지 않은 인증서와 Keystore파일들이 있었고 정적으로 인증서 비밀번호가 노출되어 있었다.

그리고 MITM Proxy를 사용하여 TLS MITM을 수행해 서버와 로봇청기와 서버간의 통신 URL과 파라미터를 중간자 공격을 통해 알아낼 수 있었다.

```
POST https://api.[redacted].com/v
    ←200 application/json 5
POST https://kic-service.[redacted]
    =post&re… HTTP/2.0
    ←200 text/plain 601b 45
GET https://api.[redacted].com/v2
    ←200 application/json 1.
POST https://api.[redacted].com/v
```

[그림 6] MITM Proxy를 사용해 TLS MITM을 수행하는 사진

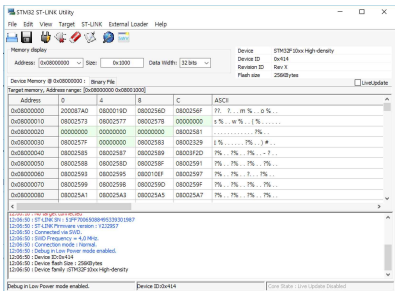
발견된 URL을 공격자의 서버로 변조 할 수 있다면, 공격자의 악성 펌웨어 업데이트 서버로 연결시키거나, 영상 데이터를 공격자의 서버로 유도하는 등 애플리케이션을 외부와 통신하는 부분을 통해 공격을 수행할 수도 있다.

그리고 앱에서 유출된 인증서 비밀번호와 인증서가 피닝되지 않은 부분을 이용하여, 변조한 인증서로 중간자 공격을 시도하여 계정을 탈취

하거나 영상을 유출하는 공격도 가능하다.

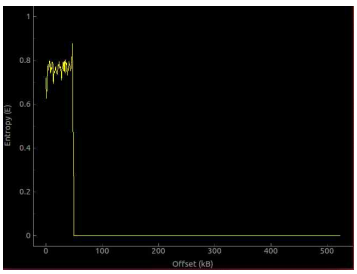
Ⅳ. 펌웨어 추출 및 기능 분석

펌웨어를 추출을 위해 우선 칩 전수조사를 해서 메인 MCU와 UART, JTAG 등의 디버깅 포트를 식별한다. 디버깅 포트 식별이 어려울 경우 메인 MCU의 데이터시트, 특히 핀 맵을 참고해 UART, JTAG 등의 디버깅 포트를 오실로스코프나 멀티미터기 등을 이용하여 식별할 수 있다. 식별한 디버깅 포트에 알맞은 디버깅 도구, 진행한 로봇청소기 같은 경우 STM32F103 MCU를 사용하므로 해당하는 JTAG 디버거인 ST-LINK/V2를 이용해 [그림 7]과 같이 펌웨어를 추출하였다.



[그림 7] ST-LINK를 이용한 펌웨어 추출

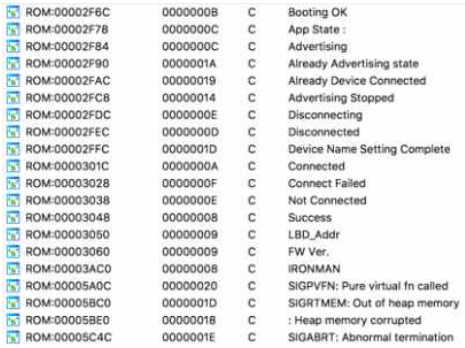
추출한 펌웨어 바이너리 파일에 binwalk를 사용하여 아키텍처, 파일 시스템, 엔트로피 분석([그림 8] 참고) 등 다양한 방면으로 펌웨어 분석을 할 수 있다. 특히, 펌웨어가 암호화가 되어 있을 경우 binwalk -E 옵션과 binvis 등으로 분석한 엔트로피 결과가 비정상적으로 나올 것이고, 이를 토대로 펌웨어 암호화 여부도 판단 할 수 있다.



[그림 8] binwalk를 이용한 엔트로피 분석

binwalk를 통해 얻은 정보를 토대로 FMK를 사용하여 펌웨어내의 파일을 추출할 수 있다. 이후, 추출한 펌웨어를 QEMU를 이용해 가상머신에 추출한 펌웨어를 구동시켜 분석을 진행할 수 있다.

하지만, 필자의 경우처럼 파일 시스템이 없고 펌웨어가 단일 파일로 동작하는 경우도 있다. 이런 경우는 binwalk를 이용하기 어려우므로 추출한 바이너리를 바로 IDA를 이용해 분석한다. 이때, IDA 분석 기능을 이용하면 만들어진 함수로 펌웨어를 분석 할 수 있다. 그러나 파일 시스템이 없는 경우에서 보통 바이너리의 심볼이 없을 확률이 높아서 분석을 위한 main을 찾기가 힘들어질 수 있다. 이 경우, IDA의 기능중 하나인 문자열 추출 기능을 사용하면 효율적으로 정적 분석이 가능하다.



[그림 9] IDA를 이용하여 추출된 펌웨어 내부 문자열

[그림 9]과 같이 문자열 추출 데이터를 토대로 분석을 진행 한다면, 어떤 동작을 수행하는 함수인지 식별하기 수월해진다. 예를 들어 문자열 데이터 중 "Connected" 같은 경우, 블루투스 연결을 진행하는 함수와 로봇청소기를 제어하는 함수가 존재했다. 또한, "Out of heap memory" 같은 경우는 추적 결과, 힙 메모리 OOB가 발생했을 경우 디버그 모드로 전환되는 함수가 존재했다. 함수를 호출하는 부분이 어디인지 확인하려면 IDA의 xrefs 기능을 통해 역추적이 가능하다.

또한 ST-LINK/V2는 디버거 자체적으로 동적 디버깅을 지원하기 때문에 디버거를 이용한 동적 디버깅을 수행할 수도 있다.

펌웨어 분석을 통해 신뢰되지 않은 입력을

받아 힙 메모리를 오염시킬 수 있는 루틴이 있거나, 민감한 데이터가 정적으로 있거나, 커맨드 인젝션을 통해 비정상적인 동작을 수행하도록 유도하는 등 다양한 취약점이 발생할 수 있다.

Ⅴ. 결론

본 논문은 점차 다양한 기능을 탑재되고 출시되는 로봇청소기 제품군의 Attack Surface 분석 방법에 대해서 로봇청소기와 앱 간의 블루투스 통신, 청소기와 앱간의 Wi-Fi 통신에서 발생할 수 있는 보안 취약점을 분석하는 방법, 앱 위변조를 통해 분석 환경 구축 및 애플리케이션과 클라우드 또는 청소기 간의 패킷을 스니핑할 수 있었다. 또한 펌웨어를 추출해 파일 시스템의 유무에 따라 QEMU를 이용해 가상머신에 펌웨어를 올려 분석하거나, IDA를 사용해 문자열 추출을 통해 main함수를 찾거나, 블루투스 통신을 수행하는 함수를 찾는 등의 로우 레벨단에서의 분석을 통해 신뢰되지 않은 입력을 받는 지점을 함수 역추적을 이용해 찾아낼 수 있었다.

위 분석을 토대로 블루투스 부분에서는 올바르지 않은 CMD 코드로 청소기 조작을 불가능하게 만드는 공격이 있었다. Wi-Fi 부분에서는 디패키징 한 앱 내부에 피닝되지 않은 인증서와 Keystore파일들, 그리고 인증서 비밀번호가 정적으로 노출되어있는 문제점을 이용해 중간자 공격을 시도하여 계정 탈취와 영상을 유출시키는 공격이 가능하다는 것을 알 수 있었다. 펌웨어 부분에서는 민감한 데이터를 유출하거나 비정상적인 동작을 수행하도록 하는 커맨드 인젝션 공격 등의 취약점이 발생할 가능성을 확인 할 수 있었다.

언급한 Attack Surface를 통해서 로봇청소기 제품의 보안 측면에서 참고 자료나 제품 보안 패치시 체크리스트를 만드는 등 로봇청소기의 보안성 향상을 기대할 수 있다.

[참고문헌]

[1] Koopman, Philip. "Embedded system security." Computer 37.7 (2004): 95-97.

[2] Parameswaran, Sri, and Tilman Wolf. "Embedded systems security－an overview." Design Automation for Embedded Systems 12.3 (2008): 173-183.

[3] Ravi, Srivaths, et al. "Security in embedded systems: Design challenges." ACM Transactions on Embedded Computing Systems (TECS) 3.3 (2004): 461-491.

[4] Chen, Daming D., et al. "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware." NDSS. 2016.

[5] F. Bellard, "QEMU, a fast and portable dynamic translator," in Proceedings of the USENIX 2005 Annual Technical Conference. USENIX, 2005, pp. 41 - 46. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/usenix05/tech/freenix/bellard.html

[6] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler, "A few billion lines of code later," Communications of the ACM, vol. 53, no. 2, pp. 66 - 75, 2010.

특정 oneM2M 오픈소스 기반 암호 키관리 기능 분석

김도원*, 최은영*, 박해룡*
*KISA 암호기술팀

Analysis on the Cryptographic Key-Management based on Specific oneM2M Open Source

Dowon Kim*, Eun-young Choi*, Haeryong Park*
*Cryptography Technology Team, KISA.

요 약

IoT 환경은 서버와 다양한 디바이스들로 구성되어 있고, 많은 오픈플랫폼들은 IoT 제품을 개발하는데에 도움을 주고 있다. 많은 오픈플랫폼들의 보안 기능은 SSL/TLS에 의존하고 있으며 실제 어플리케이션 데이터에 대해서는 관여하기 어렵기 때문에, 실질적인 암호 키에 대한 관리는 이루어지기 어렵다. 이러한 오픈플랫폼을 통해 제품을 개발할 경우, 암호 키관리 정책을 가능한 만족하는 개발이 필요하다.

I. 서론

다양한 IoT 환경은 여러 종류의 디바이스로 구성되어 있다. 시스템의 규모에 따라 대용량의 서버 혹은 작은 PC급의 서버가 활용될 수 있으며, 어떤 서비스를 지원하느냐에 따라 다양한 센서 및 소형 기기들을 이용할 수 있다.

이렇게 IoT에 활용되는 다양한 디바이스의 개발을 돕기 위한 오픈플랫폼들이 존재하며, 대표적인 IoT 오픈플랫폼으로는 oneM2M, OCF, LWM2M 등이 있다. IoT 오픈플랫폼들은 서버뿐만 아니라 라즈베리파이 등 소형 기기의 개발도 지원한다. 최근에는 라즈베리파이 같은 소형기기들도 성능이 많이 좋아져 TLS 같은 보안프로토콜도 충분히 적용이 가능하며, IoT 오픈플랫폼들도 이러한 방법을 이용하고 있다.

보안 프로토콜의 핵심 안전성은 암호 키에서 나온다. 취약한 암호 키를 이용하면, 어떠한 시스템도 안전성을 상실하게 된다. 현재 IoT 오픈플랫폼들은 범용성을 위하여 통신상에서 TLS 등을 적용하지만, 이런 오픈플랫폼을 이용하여 제품을 개발할 때 실제 어플리케이션에 대한 보안은 적용되지 않는 경우가 대부분이다.

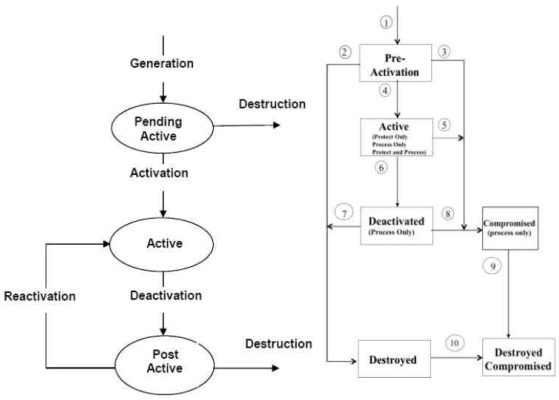
본 논문에서는 오픈플랫폼 중 oneM2M 중 공개적으로 개발되어 있는 오픈소스를 이용하여 서버-클라이언트 환경을 구축하고, TLS를 적용한다. 그리고 네트워크 구간에서 기존 NIST SP800-57A나 ISO/IEC11770-1 등을 기반으로 암호 키관리 기능을 분석하고, IoT 환경에서 사용되는 암호 키에 대하여 논의해본다.

II. 암호 키관리 정책

본 논문에서는 다양한 암호 키관리 정책 중, ISO/IEC 11770-1[1]과 NIST SP800-57A[2]를 중점적으로 살펴본다.

암호 키관리 정책에서 가장 중요한건 키의 생명주기(life cycle)이다. 키가 생성되어 삭제될 때까지의 기간을 의미하며, 두 정책의 키 생명

주기는 아래 그림과 같다.



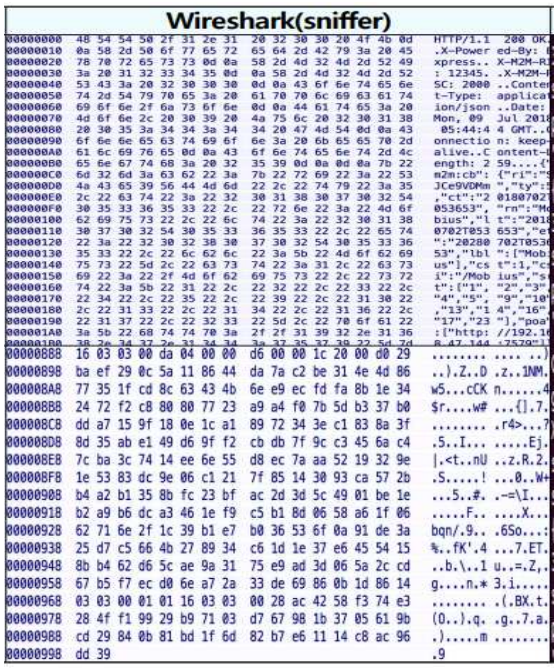
(그림 1) ISO/IEC 11770-1(왼쪽)과 NIST SP800-57A(오른쪽)의 키 생명주기

두 생명주기는 생성-활성화(actvie)-비활성화(deactive)의 개념은 동일하게 구분하고 있다. 각각 키 생명주기를 정의하고, ISO/IEC 11770-1에서는 키의 상태에 따라 발생할 수 있는 요구사항들을 간단히 설명하고 있으며, NIST SP800-57A에서는 키 유형(용도)을 구분하고 각 용도에 따라 필요한 요구사항들을 설명하고 있다.

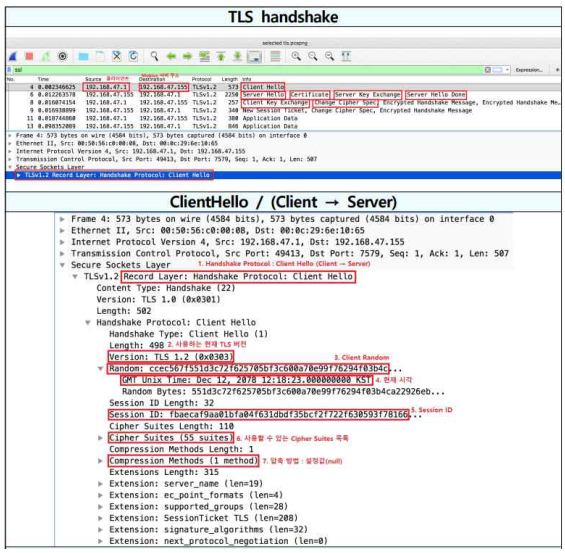
또한, ISO/IEC 11770-1에서는 삭제(deleted) 및 손상(compromised)에 대한 내용을 언급하고 있다. 예를 들어, 더 이상 필요하지 않은 키는 삭제해야 한다고 간단히 언급하고 있다. 하지만 NIST SP800-57A에서는 삭제 및 손상을 키 상태(key state)로 정의하고, 삭제 및 손상이 발생하였을 때의 요구사항을 별도로 설명하고 있다.

III. oneM2M 오픈소스 키관리 기능 분석

본 장에서는 oneM2M 오픈소스를 기반으로 서버와 디바이스간 통신환경을 구축하고 TLS 프로토콜을 적용한 뒤, 키관리 기능을 분석하였다. 중간 과정에서 스니퍼를 이용하여 패킷을 캡쳐하였으며, TLS 프로토콜의 핸드쉐이크 과정과 적용 전후는 아래 (그림 2)와 같다.



(그림 2) oneM2M 오픈소스에 대한 TLS 적용 전후 패킷 분석



(그림 3) TLS핸드쉐이크 패킷 캡쳐와 ClientHello 패킷 내용

통신 채널에 대하여, TLS를 적용하여 패킷이 암호화된 것을 확인할 수 있다. 하지만, TLS 적용 전 패킷에 대하여 평문 데이터가 그대로 전송되는 것을 확인할 수 있다.

실제로 오픈플랫폼을 따르는 대부분의 오픈소스들은 대부분 SSL/TLS 프로토콜이나 MQTT같은 경량환경에 적합한 프로토콜을 이용하며, 그 어플리케이션 데이터까지 관여하지

않는다. 암호학적인 관점에서 안전성을 요구하는 기존 암호 키관리 정책을 기반으로 살펴보았을 때, 아무런 키관리 기능하고 있지 않은 것이다.

IV. IoT 환경에서의 암호 키관리 기능

난수성 평가나 수집한 잡음원에 대한 엔트로피 검증 등과 키 삭제 후 메모리 및 저장공간 확인 등 암호 키에 대한 안전한 생성부터 삭제까지, IoT 환경에서 기존 암호 키관리 정책들에서 요구하는 상황을 충족시키기는 어렵다. 또한, 고성능을 요구하는 암호알고리즘들을 구동할 수 있을만큼 실제 디바이스 성능이 좋아졌다고는 하나, 용도에 따라 다양한 암호 키를 보유하고 관리하는 것은 더욱이 어렵다.

그렇기 때문에, 최초 개발시 보안 기능을 정확하게 검토하는 것이 바람직하다. 실제 IoT 디바이스들은 종류가 다양하지만, 한 번 서버와 연동되어 구동되면 무중단 서비스를 제공하는 제품이 많다. 특히 많은 IoT 제품들은 내부에 자체적으로 보유한 암호 키 혹은 난수를 이용하는 경우가 많은데, 이러한 경우 취약점이 발견될 경우 수정이 어려우며 해당 키를 이용하여 생성된 다른 키에 영향을 주게 된다.

따라서, 암호 키관리 정책에서 요구하는 사항 중 현실적인 부분에서 적용가능한 요소들을 가능한 만족하면서, 최초 암호 키를 안전하게 생성하는 것이 중요하다.

V. 결론

본 논문에서는 oneM2M을 따르는 특정 오픈소스를 이용하여 환경을 구축하고, 구현되어 있는 암호 키관리 기능을 살펴보았다.

IoT 환경은 너무나 다양하고 여러 가지 어려움이 존재하기 때문에, 실제 구현되는 디바이스들은 암호학적인 관점에서 요구하는 암호 키관리 정책의 요구사항들을 만족하기 어렵다. 모든 요구사항을 만족할 순 없지만 가능한 안전한 방식을 택하여 디바이스에 대한 보안을 유지하여야 한다.

[참고문헌]

[1] Information technology - Security techniques - Key management - Part 1: Framework, ISO/IEC 11770-1:2010, 2012.

[2] Elaine Barker (NIST), William Barker (NIST), William Burr (NIST), W. Polk (NIST), Miles Smid (Orion Security Solutions), Recommendation for Key Management, Part 1: General(Revised), NIST SP800-57 Part 1 Revised, July, 2012.

Security management for backhaul-aware 5G-V2X

Vishal Sharma, Yongho Ko, Jiyoon Kim, Ilsun You

Information Security Engineering Department, Soonchunhyang University, Asan-si-31538, The Republic of Korea

vishal_sharma2012@hotmail.com, koyh0911@gmail.com, 74jykim@gmail.com, ilsunu@gmail.com

Abstract

Security is a primary concern for the networks aiming at the utilization of Cellular (C) services for connecting Vehicles to Everything (V2X). At present, C-V2X is observing a paradigm shift from Long Term Evolution (LTE) - Evolved Universal Terrestrial Radio Access Network (E-UTRAN) to Fifth Generation (5G) based functional architecture. However, security and credential management are still concerns to be resolved under 5G-V2X. A sizably voluminous number of key updates and non-availability of sub-functions at the edge cause adscititious overheads and decrement the performance while alarming the possibilities of variants of cyber attacks. In this paper, security management is studied as a principle of sustainability and its tradeoff is evaluated with the number of key-updates required to maintain an authenticated connection of a vehicle to the 5G-terminals keeping intact the security functions at the backhaul. A numerical study is presented to determine the claims and understand the proposed tradeoff.

I. INTRODUCTION

With the availability of new radio access technologies, provisioning different types of services across the Cellular-Vehicle to Everything (C-V2X) is seen as the forefront of 5G networks [1][2][3]. This has been dominatingly termed as 5G-V2X where the core security and general functions are considered for facilitating the security and services for vehicles involved in the formation of the Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I) and Vehicle to Pedestrian (V2P) networks as a part of V2X. Connecting fronthaul entities to the network and ensuring much of the operations at the edge require specific divisibility phenomenon which can simultaneously acquire as well as manage the backhaul operations [3][4].

Most of the existing studies have presented this as a resource allocation problem [5][6]. However, there is no evident study available which aims at providing a strategic management of security for 5G-V2X while considering the appropriate security for the backhaul formed between the Terminal (TM) and the hub as a part of the basic architecture. The initial reports on 5G have provided a specific security function architecture which can be integrated with the required base model as well as a security protocol to authenticate the devices involved in the transmissions [2][7]. Initially, 5G-Authentication and Key Agreement

(AKA) protocol and Extensible Authentication Protocol (EAP)-AKA prime are seen as competitive solutions for authenticating network entities [2]. However, the initial reports do not confirm the overheads associated with the periodic key updates, mobility management of vehicles, as well as the sustainability of V2X against a known set of cyber threats [8][9]. Periodic key updates undoubtedly enhance the security of a network, but this procedure, in the case of limited requirements, causes an uncertain burden on the entities and may result in high computational complexity as well as high operational cost [6][10]. Thus, this article helps to understand the requirements of security management for V2X and also considers the possibilities of sub-dividing the 5G security functions to make it suitable for handling operations as well as authentication procedures of V2X at a high rate of sustainability with lesser key-updates.

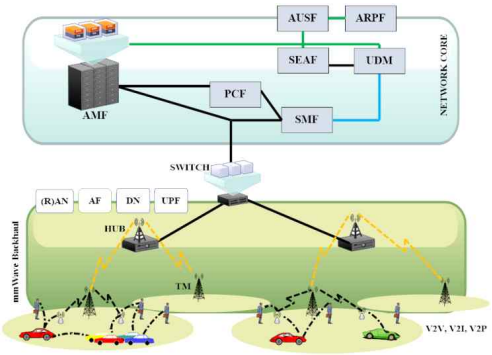
The remaining part of this article is organized as follows: Section II presents the problem statement and highlights our contribution. Section III discusses the system model. The proposed approach is presented in Section IV. Section V covers the numerical evaluations. Finally, Section VI concludes the article.

II. PROBLEM STATEMENT AND OUR CONTRIBUTION

Managing security for backhaul-aware 5G-V2X is tedious as it depends on the architectural deployment and 5G function-mappings to the underlaid network. Moreover, with a key focus on edge-initiation, bringing security-aspects near to users raise concerns of trustworthy security facilitations, especially when the involved entities are vehicles. With varying mobility and high dynamics in V2V, V2I and V2P formations, management of security needs an efficient solution which can protect the network under major attack scenarios. In this paper, the key exchange passes between the vehicles and 5G-security functions, which are deployed on the core, hub and the switches, are considered. The article also considers the backhaul between the TM and the hub for management as a part of the 5G deployment. The primary objective is to identify the time slots when the keys should be updated in order to protect the network against known cyber threats. Alongside, identification of fail-safe points up to which the network can be operated with the current keys is to be determined. Credential management, and certifying the key-hierarchy principles or enabling edge-initiated security are further considered as a part of the derived optimization problem. It should be considered with an extreme importance that updating keys is a crucial part of the network which has to deal with a large number of applications. Network with a large number of key-updates can be secured up to much extent but at the cost of performance. Thus, this tradeoff needs to be balanced for attaining an efficient backhaul-aware 5G-V2X. Based on the solutions presented in this article, the core contributions can be marked as follows:

- Dual security management solution is presented that considers security through long-range and short-range authentications
- A fail-safe point is identified as a part of the optimization solution, until which, the network can be operated without changing the derived keys.

- A new hierarchy principle is presented for managing key derivations from the 5G security functions.



[Fig 1] An illustration of 5G-V2X with mmWave Backhaul between the Hub and the terminal based on 5G security and general network functions.

III. SYSTEM MODEL

The network comprises 5G security functions which are mapped across the core, TMs, switches, User Equipment (UE), and hubs, as shown in Fig.1. The UEs involve the vehicles and general users that form V2V, V2I and V2P links during their operations. The core involves the Access and Mobility Management Function (AMF), which operates together with the Session Management Function (SMF) and Policy Control Function (PCF), and as described in the initial TS by 3GPP [2], the Security Anchor Function (SEAF) is placed in their periphery which is located deep in the network without any knowledge to the edge as well as vehicles. As it is tedious to consider such a deployment for supporting edge-initiated security as well as authentication for V2X, the proposed part modifies it and presents a new disintegrated structure to make the 5G-drafted version suitable for V2X. SEAF keys and authentication are driven by the Authentication Server Function (AUSF) and the Authentication Credential Repository and Processing Function (ARPF) [2].

The mmWave backhaul supports the Unified Data Management (UDM), Application Function (AF), (Radio) Access Network ((R)-AN), and User Plane Function (UPF). In accordance with the problem statement, the issue is with the security management that involves the V2X authentication and TM authentication with the hub along with the placement of major details at the edge without costing the performance. This article considers security to be directly proportional to the number of key-updates performed in the V2X setup. However, with an excessive number of updates, the network induces certain overheads, which slow down the operations and increase the burden of credential management. To resolve this, at first, a timing problem is formulated. According to which, let t be the time taken by an adversary to launch an attack, out of which t' be the minimum time for which the keys should not be changed. Considering this, the key-utilization time, t_u, must be identified, such that $t_u < t'$ $(<=t)$, for which the currently allocated keys can be safe from known attacks. Moreover, intermediate-key updates, U_k, should be minimized while maximizing the overall sustainability, S_N, of the network. Here, S_N is derived as a function of induced overheads by using IPAT formulations [11], such that

$$S_N = \frac{n U_k}{D.P.Q}, \quad (1)$$

where $D(\leq N) = \int_{r_1}^{r_2} C(x,y)dx$ is the number of cars in the periphery, r, of a particular TM for a given density function $C(x,y)$. P is the probability of loss in connectivity; Q is the number of passes between the entities, n is the sustainability

balancing constant depicting the inverse of number of hops between the vehicles and the entity dealing with the particular request generated for a 5G security function (key or service requests), N is the end devices (vehicles, UEs, or users), and E is the overall involved entities. Now, based on the above-discussed security issues, the optimization problem can be formulated as:

$$\max(S_N), \quad \forall E, \forall N, \quad (2)$$

s.t.

$$\max(t_u), \forall N,$$

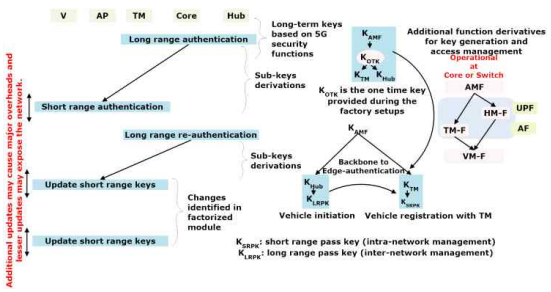
$$\underbrace{\min(\quad U_N \quad)}_{\text{tradeoff with } S_N}, \text{and} U_N \geq U_N{}' \ ,$$

$$0 < D \leq N,$$

$$0 < \frac{n^{-1}(n^{-1}-1)}{2} \leq \frac{E(E-1)}{2}, n^{-1} \neq E,$$

$$\min(t_u - t' \ ). \quad (3)$$

Here, $U_N{}'$ is the mandatory key-updates below which it is difficult to evaluate the network for security and perform any tasks, such as mobility management, re-authentication



[Fig 2] An overview of the security considerations, key generations, long and short-range authentication and additional functions derived for managing sub-keys.

IV. PROPOSED APPROACH

The proposed approach establishes a security management framework which operates as a dual security driving system based on two modes for authentication-long range authentication and short-range authentication. Both these help to provide a strong authentication for V2X while ensuring the security of backhaul between the TM and the hub. The proposed approach maintains a plan for authentication via key exchanges that are performed by utilizing the deployed 5G security functions. The long-range authentications are performed for the backhaul whereas the short-range authentications are performed for the edge-initiated network formations. The initial mode is dependent on the derivations of keys from the core, which is a part of long-range authentication and sub-division of initial keys support the procedures for short-range authentication. The long-range authentication is operated over a powerful mmWave communications, which can be facilitated by existing encryption algorithms and can be motivated to update the keys once the devices are deployed or reconfigured. However, the major issue prevails at the edge-initiated authentication for the vehicles, which under high dynamics, causes an additional burden of key-regenerations as well as credential management. To further resolve this issue, a factorized module is considered in the short-range authentication, which evaluates the speed (S), location (L), last updates for keys (U_T), shared sessions (A_S), refreshing rate of keys (F_R), total keys (T_K), zone traversals (Z_T), and associatively (V_A) of a vehicle to generate new keys or continue with the existing keys while maintaining the conditions stated in (2). An overview of the proposed approach with key dependencies is presented in Fig.2 with the description as follows:

- The AMF function is sub-divided into the terminal function (TM-F) and Hub

function (HM-F) that derive their keys from K_{OTK}, which is derived from K_{AMF}.

- The derived keys are used for initial long-range authentications and short-range authentications. The sub-derived keys are used for authenticating vehicles' fronthaul to the edge and the backhaul between the TM and the hub.
- The sub-derivative functions help to support the intra- and inter-mode of secure communications in V2X by deriving their core operations from the 5G security functions.

At present, the details of internal procedures for the proposed framework and authentication mechanisms are omitted from the article and prime focus has been given to understand the sustainability of the backhaul-aware 5G-V2X under the given constraints of key updates. These can further be observed from the following results:

Lemma-1: S_N is divergent if the initial information on the key generation is unavailable. However, in the case of known timestamps, the network sustainability can be modeled over the available key updates and vehicle movement, such that for instances t_1 and t_2:

$$S_N = \frac{\alpha^2}{2\beta N(1-\frac{n^{-1}}{E})^N Q}(Ei(\frac{\beta-\alpha}{t_1}) - Ei(\frac{\beta-\alpha}{t_2})), \quad (4)$$

at t_2-t_1>0, E-n^{-1}>0, β-α>0, and Ei is the exponential integral.

Proof: Considering (1) to be following Poisson distribution for vehicles approaching at a rate of β vehicles per unit time and operating with α number of key-updates per unit time, U_k can be given by $e^{\frac{-\alpha}{t}}\frac{(\frac{\alpha}{t})^X}{X!}, (X=2)$, as only 2 keys are used for authentication (long range and short range keys) and D can be written as $e^{\frac{-\beta}{t}}\frac{(\frac{\beta}{t})^{X'}}{X'!}, (X'=1)$, as each vehicle maintains its connectivity with one source from the network. Now, considering the involved entities in the network for authenticating a vehicle, and using the model in [12], the probability of no connectivity *(P)* can be given as $(1-\frac{n^{-1}}{E})^N$. The number of passes remains at the discretion of the used protocol and is constant for this evaluation. By using these values in (1) and under fixed interval, the observed equation can be written as:

$$S_N = \frac{1}{N(1-\frac{n^{-1}}{E})^N Q}\int_{t_1}^{t_2}\frac{e^{\frac{-\alpha}{t}}\frac{(\frac{\alpha}{t})^2}{2!}}{e^{\frac{-\beta}{t}}\frac{(\frac{\beta}{t})^1}{1!}}dt. \quad (5)$$

On solving, at t_2-t_1>0, E-n^{-1}>0, β-α>0, the observation is

$$S_N = \frac{\alpha^2}{2\beta N(1-\frac{n^{-1}}{E})^N Q}(Ei(\frac{\beta-\alpha}{t_1}) - Ei(\frac{\beta-\alpha}{t_2})), \quad (6)$$

which is the desired output.

Lemma-2: For extreme large rates of key exchanges and vehicle dynamics, S_N is convergent to a linear function under asymptotic observations. This can be further used to identify fail-safe points with high accuracy up to which the network can be operated without much overheads and security breaches.

Proof: In continuation from Lemma-1, the result in (6) can be asymptotically analyzed for evaluating the behavior of the curve determining the sustainability of the V2X under given constraints. According to which, if α and β increases to a larger value, the (6) reduces to a

linear function, such that S_N can be evaluated as a function comprising two metrics, i.e. $S_N = f(\frac{\alpha}{t}, \frac{\beta}{t})$ and by following strict principles and limits, it can be given as $S_N = \frac{\alpha}{\beta}$. Moreover, in such a case, the behavior is not integrally divergent and the network can be evaluated with linear timestamping. Now, considering these observations, the network fail-safe points up to which there is no need to update or refresh the keys can be evaluated as:

$$F_S = \begin{cases} t \text{ at } S_N \geq S_N^{TH}, \text{ if } t_1 \neq 0 \text{ and unknown} \\ t \text{ at } M_O \leq M_O^{TH}, \text{otherwise} \end{cases}. \quad (7)$$

Here, M_O is the message overhead evaluated as

$$M_O = O_S \frac{(1-P)}{E.P}, \quad (8)$$

where O_S is said to be increasing exponentially. This is due to the reason that the key updates and vehicles follow the Poisson distribution with randomness observable from Lemma-1, based on which, O_s can be modeled as $O_b(1-\frac{\alpha}{t})^t$, where O_b is the overheads for initial authentication. This can be predicted between the instances, as shown in Lemma-1, only at a fixed number of updates $(\alpha' = \frac{\alpha}{t})$,, such that

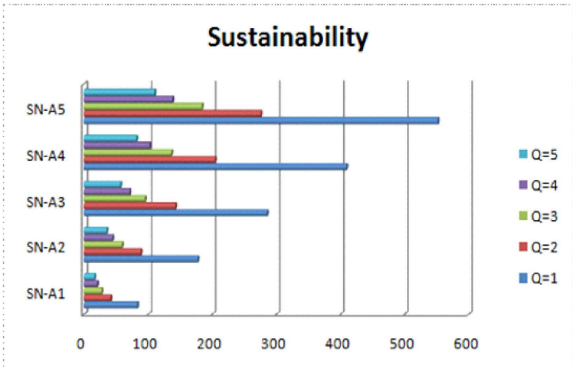
$$O_S = \frac{O_b(\frac{n^{-1}}{E})^N}{E(1-\frac{n^{-1}}{E})^N} \cdot \frac{\ln(1-\alpha')^{t_2} - \ln(1-\alpha')^{t_1}}{\ln(1-\alpha')}. \quad (9)$$

These observations are required to determine the exactness of updates for the available values of sustainability as well as message overheads. The thresholds are identified based on the data pool available for the time required to launch an attack particularly on the protocol used for the authentication of vehicles to everything or TM to the hub.

Metrics	Values	Metrics	Values
β	2-10 (step 2) (A1-A5)	n^{-1}	5
α	β/2	Q	1-5
N	10	t_1	5 s
E	1-10	t_2	105 s

[Table 1] Parameter Settings

Security of backhaul-aware 5G-V2X cannot be quantified and it depends on the operations of the involved authentication protocol, which will be presented in our future reports. However, the timing to update keys and managing security based on the arrival rate of vehicles is quantifiable and it can be observed in the form of sustainability as expressed in the earlier part of this article. To further understand the impact of metrics on the sustainability and the location of fail-safe points, a numerical case study is conducted by using metrics defined in Table 1.The values are based on a general setup described in the initial parts of this article and other metrics are assumed within the constraints of Lemma-1 and Lemma-2. The result is recorded for S_N, as shown in Fig.3.



[Fig 3] Network sustainability w.r.t. the number of protocol passes at different rates for incoming vehicles and the number of key updates.

The results suggest that with the proposed strategy, it is feasible to effectively trace the network activity and manage its security by keeping a check on unnecessary key updates. By this, significant overheads can be reduced from

the network and the security can be provided at the edge while keeping a close associatively with the network-backhaul. It is observed that the number of passes and key-updates used for authentication pose a significant impact on the performance as well as the security of the network. Numerically, the results vary between 53.1% and 84.9% compared to initial observed output at constant arrival and update rates for vehicles and keys, respectively, with a varying value of Q. It is evident that the sustainability of the network can be enhanced even with an increase in the number of vehicles or key-updates by reducing the number of passes between the vehicle and the infrastructure. Thus, it is important to carefully select the authentication protocol and it must not cause excessive signaling overheads. For this, the proposed approach leverages on the divisibility of 5G security functions based on the requirements at the specific edge leading to the formations of TM-F and HM-F, which help to effectively manage the security in a backhaul-aware 5G-V2X.

V.CONCLUSION

In this article, a tradeoff between the sustainability and the number of key-updates is managed in a backhaul-aware 5G-V2X. A security management framework is proposed which considers security through long- and short-range authentications. Analytical evaluations are presented to study the impact of key-exchanges, the arrival rate of vehicles and the number of authentication passes on the sustainability of the network. Following which a fail-safe point is identified as a part of the optimization solution. To generalize, new functions are derived to handle the conceptualization of the proposed solution.

This is an incremental article and further information on the authentication procedures, key-exchanges, and operational details will be presented in our future reports.

[ACKNOWLEDGMENT]

This work was supported by 'The Cross-Ministry Giga KOREA Project' grant funded by the Korea government(MSIT) (No.GK18N0600, Development of 20Gbps P2MP wireless backhaul for 5G convergence service)

[REFERENCES]

[1] Wang N, Hossain E, Bhargava VK. Backhauling 5G small cells: A radio resource management perspective. IEEE Wireless Communications. 2015 Oct;22(5): 41-9.

[2] "Technical specification -3GPP TS 33.501 V0.7.1 (2018-01)." http://www.3gpp.org/ftp /Specs/archive/33 series/33.501/ [Last Accessed -September 2018].

[3] Chen S, Hu J, Shi Y, Peng Y, Fang J, Zhao R, Zhao L. Vehicle-to-everything (v2x) services supported by LTE-based systems and 5G. IEEE Communications Standards Magazine. 2017;1(2):70-6.

[4] Molina-Masegosa R, Gozalvez J. LTE-V for sidelink 5G V2X vehicular communications: a new 5G technology for short-range vehicle-to-everything communications. IEEE Vehicular Technology Magazine. 2017 Dec;12(4):30-9.

[5] Sharma V, You I, Kumar R. Resource-based mobility management for video users in 5G using catalytic computing. Computer Communications. 2018 Mar 1;118:120-39.

[6] Sharma V, Kim J, Kwon S, You I, Leu FY. An overview of 802.21 a-2012 and its incorporation into iot-fog networks using osmotic framework. InInternational Conference on Internet of Things as a Service 2017 Sep 20 (pp. 64-72). Springer, Cham.

[7] Campolo C, Molinaro A, Iera A, Menichella F. 5G network slicing for vehicle-to-everything services. IEEE Wireless Communications. 2017 Dec;24(6):38-45.

[8] Bian K, Zhang G, Song L. Toward Secure Crowd Sensing in Vehicle-to-Everything Networks. IEEE Network. 2018 Mar;32(2):126-31..

[9] Muhammad M, Safdar GA. Survey on existing authentication issues for cellular-assisted V2X communication. Vehicular Communications, volume 12, April 2018, Pages 50-65.

[10] Sharma V, Kim J, Kwon S, You I, Chen HC. Fuzzy-based protocol for secure remote diagnosis of iot devices in 5g networks. InInternational Conference on Internet of Things as a Service 2017 Sep 20 (pp. 54-63). Springer, Cham.

[11] Chertow MR. The IPAT equation and its variants. Journal of industrial ecology. 2000 Oct;4(4):13-29.

[12] Keykhaie S, Mahmoudifar A. Study of connectivity in a vehicular ad hoc network with random node speed distribution. In 6th International Conference on New Technologies, Mobility and Security (NTMS), 2014 Mar 30 (pp. 1-4)

경매알고리즘기반 분산블록체인마이닝 취약점 분석

김중헌*, 신명재*, 전주형*, 김준희*
*중앙대학교 소프트웨어학부

Risk Analysis in Auction-based Distributed Blockchain Mining

Joongheon Kim*, MyungJae Shin*, Joohyung Jeon*, Junhui Kim*
*School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea

요 약

본 논문은 블록체인의 분산 마이닝을 5G네트워킹이 가능한 Internet-of-Things (IoT) 및 모바일 스마트폰 기기에서 수행하는 시스템을 가정한다. 이러한 시스템에서 최근 이루어진 주된 연구 결과는 경매(auction)알고리즘을 사용하는 것이다. 그러나 이러한 경매 알고리즘을 사용함은 전체 시스템 안정성에 문제를 야기할 가능성이 있다. 본 논문은 해당 문제점을 정의하고 심각성을 논한다.

I. 서론

블록체인 기반 가상화폐 관련 기술이 최근 많은 각광을 받고 있다[1-5]. 이러한 가상화폐에서 이득을 취하기 위하여 사용자가 수행하는 일 중의 하나가 가상화폐 마이닝(cryptocurrency mining)이다. 해당 마이닝 기술을 통하여 이득을 취하기 위하여 대용량 고성능 컴퓨팅(High Performance Computing) 및 슈퍼컴퓨팅(Supercomputer)환경을 구축하는 것이 일반적인 모습이다. 그러나 이러한 고성능 컴퓨팅 환경을 구축하는 것은 모바일 환경에서는 불가능하다는 것이 일반적인 통념이며 현재 모바일 컴퓨팅 시스템을 지원하는 하드웨어의 트랜드를 봐서도 불가능하다는 점은 자명하다.

최근 엣지 컴퓨팅(Edge Computing)등 5G 통신 및 네트워킹 기술의 발전으로 인하여 네트워킹의 속도 향상을 가져오게 되었으며 더불어 고성능 컴퓨팅 하드웨어의 경량화와 스마트폰의 성능 향상에 근거하여 모바일 스마트폰 플랫폼에서 블록체인 마이닝을 고려하는 것이 가능해 지게 되었다[1-2].

그러나 이러한 블록체인 마이닝을 완전하게 모바일 스마트폰에서만 수행하는 것은 아직은 큰 무리가 따른다. 즉 스마트폰에서 일부의 가상화폐 마이닝 Workload를 클라우드에 넘겨주고 더불어 해당 Workload 수행에 대한 Payment를 정하여 클라우드에게 넘겨준다. 이러한 방식은 일반적으로 신뢰적(Truthful)이라는 것이 잘 알려진 사실이다. 그러나 본 알고리즘에 대한 이러한 개념은 본 시스템에서 통용되지 않을 수 있다. 본 논문에서는 이러한 위협요소에 대해서 논하도록 한다.

본 논문은 다음과 같이 구성되어 있다. 제2장에서는 경매 알고리즘 기반 알고리즘의 기본적인 개념과 그 시스템이 가지는 위협요소에 대해서 알아보도록 한다. 마지막 제3장에서는 본 논문의 결말을 맺는다.

II. 위협모델

2.1 시스템 모델

본 논문에서는 논하는 시스템은 다음과 같이 다수의 모바일 기기가 존재하고 있고 하나의 클라우드 컴퓨터가 존재하고 있다. 이 하나의 클라우드 컴퓨터가 마이닝 일부를 수행해 줄 하나의 모바일 기기를 선택하는 것이 본 시스템에서 경매 알고리즘을 운영하는 목적이다.

각각의 모바일 기기는 자신이 수행하고 있는 마이닝의 Workload 일부를 bid로 설정하여 경매에 참여하고 마이닝을 통하여 얻는 이득의 일부를 클라우드에 지불하므로 이를 payment로 설정한다. 클라우드는 이 big와 payment를 보고 경매 당사자 모바일 기기 중에서 하나의 기기를 선별한다.

일반적인 경매알고리즘은 참여당사자들이 해당 자원을 얻기 위하여 무리하게 payment를 할 경우에는 쉽게 자신의 자원을 소모하게 되므로 자율적으로 신뢰성(Truthful) 있는 동작을 하게 된다는 것이 분산 네트워크 자원 분배 문제에서 가정하는 기본적인 원칙이다.

2.2 위협 모델

그러나 이러한 신뢰성있는 동작은 해당 블록체인 마이닝을 위한 분산 자원할당에서는 가능하지 않는 경우가 존재한다. 하나의 악의적인 모바일 기기가 있다고 가정하자. 이 경우에 그 악의적인 모바일 기기는 마이닝 Workload를 매우 적게 하여 big를 설계하고, 그에 대한 payment는 극단적으로 크게 함으로써 클라우드가 매번 그 모바일 기기를 선택하도록 유도한다. 일반적인 상황에서는 이러한 행동이 악의적인 모바일 기기의 자원을 빠르게 소모하게 하나 이 경우에는 payment가 새로 얻는 이득의 일부로 표현되기 때문에 자원이 완전히 소진되는 경우는 발생하지 않는다. 따라서 기존에 경매알고리즘 기반의 자원분배가 더 이상 신뢰성 있지 않게되는 경우가 발생하게 된다.

III. 결론

본 논문에서는 기존에 신뢰성(truthful)있다고 알려져 온 경매알고리즘 기반 분산자원분배 기법이 고려하고 있는 분산 가상화폐/블록체인 마이닝 시스템에 신뢰성 있지 않게 될 수 있는 경우에 대해서 알아보았다. 해당 문제점은 추후 bid 및 payment 설계를 지능적으로 함으로써 해결될 수 있다고 판단되며 이에 대한 연구를 추후 계속 이어갈 예정이다.

사사문구

본 연구는 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. 2018-0-00170, 이동체간 가상현실을 위한 5G 이동통신 기술 연구).

[참고문헌]

[1] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When Mobile Blockchain Meets Edge Computing: Challenges and Applications," *IEEE Communication Magazine*, 56(8):33-39, August 2018.

[2] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal Auction for Edge Computing Resource Management in Mobile Blockchain Networks: A Deep Learning Approach," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[3] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social Welfare Maximization Auction in Edge Computing Resource Allocation for Mobile Blockchain," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[4] S. Kim and J. Kim, "POSTER: Mining with Proof-of-Probability in Blockchain,"

in *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, Incheon, Korea, June 2018.

[5] H. Lee, M. Shin, K.S. Kim, Y. Kang, and J. Kim, "Recipient-Oriented Transaction for Preventing Double Spending Attacks in Private Blockchain," in *Proceedings of the IEEE International Conference on Sensing, Communication and Networking (SECON)*, Hong Kong, June 2018.

차세대 네트워크 시대를 대비한 네트워크 장비의 침해 사례 분석

김완수* 신대민**

*한국교육과정평가원 **순천향대학교 정보보호학과

Analysis of Network Equipment Infringement Case for the Next Generation Network

Wan-Soo Kim* Dae-Min Shin*

*Korea Institute for Curriculum and Evaluation

**Division of information security, Soonchunhyang University.

요 약

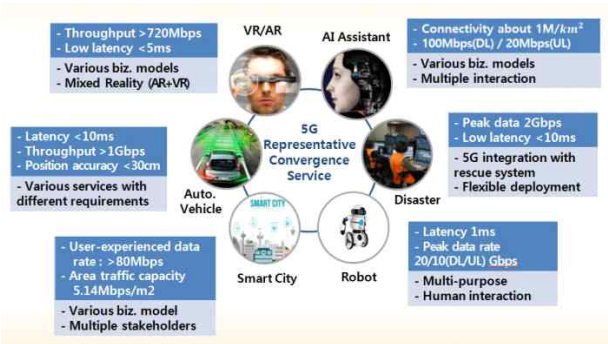
5G 이통통신은 가상현실, 자율주행차, 로봇 원격제어, 각종 스마트 기기와 각종 IoT 센서 연결 등을 가능하게 하는 기반 기술로 한 차원 높은 서비스 제공과 새로운 개념의 차세대 산업이 등장할 수 있는 기반이 된다. 그러나 5G 이동통신 상용화를 앞두고 5G 이동통신을 구현하는 기술적인 문제만이 아니라 외산 이동통신 네트워크 장비와 관련하여 보안 침해에 대한 문제가 제기되고 있다. 외산 네트워크 장비의 경우 해킹 또는 백도어와 같은 보안 침해로부터 자유로울 수 없다. 그러므로 이동통신 네트워크 장비의 보안 침해에 관한 인식을 제고하고 관련 보안산업 및 전문인력을 양성하며, 국가정보호기본법에서 이동통신 네트워크 장비의 보안 관련 제도를 명시하여 외산 네트워크 장비의 보안 취약점을 평가할 수 있도록 하여야 한다.

I. 서론

스마트폰 사용자의 데이터 사용량이 급격히 증가하여, 2016년 말 한 달 동안의 총 데이터 트래픽은 7.2EB(Exabytes)에 도달하였으며 이 수치는 지속적으로 증가하여 2021년에는 49EB에 달할 것으로 전망되고 있다. 또한 각 스마트폰에서 생성하는 트래픽 역시 2016년 월평균 1.6GB에서 2021년 6.8GB로 4배 이상 증가할 것으로 예측되고 있다(). 이러한 예견되는 데이터 급증에 대비하고 새로운 이동통신 서비스를 창출하기 위해 세계 각국 정부와 기업들 사이에서는 차세대 이동통신(이하 5G 이동통신) 상용화를 위한 경쟁이 벌어지고 있다. 차세대 이동통신 네트워크는 소비자들에게 자율주행차, 가상현실(Virtual Reality; VR), IoT 등 지금보다 한 차원 높은 서비스를 제공할 수 있으며, 새로운 개념의 차세대 산업이 대두될 수 있는 기반이 된다.

5G 이통통신을 간단히 정의하면 초고속, 초저지연, 초연결의 3가지 특성으로 설명할 수 있다. 첫째, 5G 이동통신은 한 사람의 이용자에게 최대 20Gbps(1초에 2.5GB 전송)의 빠른 전송속도를 제공하며, 네트워크가 열악한 지역에서도 100Mbps(1초에 12.5MB 전송) 속도를 유지하는 초고속 특성을 갖는다. 이에, 3차원 화상통화, 원격진료와 같이 가상현실과 증강현실을 이용한 업무 또는 엔터테인먼트가 활성화될 수 있다. 둘째, 네트워크의 안정성이 높고, 통신을 시작하는 데 걸리는 지연시간이 극도로 짧아 기존 수십 밀리세컨드였던 지연시간이 1ms 수준으로 짧아지는 고신뢰 및 초저지연 특성을 갖는다. 이런 특성은 로봇 원격제어, 자율주행차, 실시간 양방향 네트워크 서비스를 가능하게 할 수 있다. 셋째, 1km^{2}의 면적 안에서 100만개의

기기들을 연결할 수 있도록 지원하는 초연결의 특성을 갖는다. 이런 대량 연결을 통해 각종 스마트기기, 신발과 의복 등에 부착되는 센서, 가전제품과 수도·가스 등 각종 검침 및 계량기, 스마트 팜, 스마트 공장에서 사용되는 센서 등 기하급수적으로 증대될 IoT 장비를 수용할 수 있도록 한다. 5G 이동통신을 기반으로 한 서비스 시나리오는 [그림 1]과 같다.



*출처: 전자신문, 2017. 11. 23.

[그림 1] 5G 6대 융합 서비스

이렇듯 많은 서비스를 가능하게 할 것으로 기대를 받고 있는 5G 이동통신이지만 상용화를 앞두고 여러 문제가 제기되고 있다. 특히, 단순히 5G 이동통신을 구현하는 기술적인 문제만이 아니라 5G를 상용화하기 위한 국가와 기업 간의 이해관계로 인한 문제가 발생하고 있다. 현재 5G 상용화의 가장 큰 문제점은 장비 사업자 선정이다. 간단히 말하면 중국의 네트워크 기업인 화웨이(Huawei)의 장비를 쓸 것이냐에 대한 문제가 제기되고 있다. 화웨이는 현재 무선 네트워크 장비 시장 점유율 1위 업체이다. 화웨이의 강점은 저렴한 구축 비용과 기술력이다. 동급의 장비를 삼성전자, 노키아, 에릭슨 등 다른 네트워크 장비 회사보다 20~30% 정도 저렴하게 공급하고 있으며, 기술력 또한 다른 회사에 뒤지지 않는다.

그럼에도 불구하고 미국, 호주, 영국 등의 국가에서 통신망 장비의 정치적 보안위협과 안보 문제를 근거로 화웨이의 장비가 자국에 들어오는 것을 금지하고 있다. 화웨이와 관련해 보안 우려가 큰 것은 이 회사가 중국 인민해방군 출신들이 주축이 돼 설립됐기 때문이다. 런정페이 화웨이 회장은 인민해방군 통신장교 출신으로 1987년 회사 설립 후 인민해방군의 프로젝트를 독점 수주해 성장 발판을 마련했다. 정부와의 특수 유착관계로 사실상 국영기업과 다를 바 없다는 게 업계 일각의 시선이다. 이에 5G 이동통신과 같이 대용량의 데이터를 기반으로 다양하고 융합적인 서비스를 제공하는 국가기간망에 설치되는 통신장비를 외산 통신장비, 특히 화웨이 통신장비를 사용하는 것이 국가안보에 문제가 되지 않는가는 의문이 제기되고 있다.

II. 화웨이 관련 보안 침해 사례

네트워크 장비와 관련한 보안 이슈는 크게 해킹과 백도어와 관련한 문제가 제기되고 있다. 보안 논란이 일고 있는 화웨이 제품의 보안 취약점이 매년 급증하고 있는 것으로 나타났다. 지난 6월 기준으로 올해 화웨이 제품에서 드러난 보안 취약점만 152개에 달하는 것으로 나타났다. 이는 지난 한해 발견된 취약점 숫자(169개)와 유사한 수치이다. 화웨이의 보안 취약점은 매년 증가하는 추세로 2012년 3개, 2013년에 12개에서 2016년에는 100개로 급증했다. 이러한 보안 취약점은 곧 외부에서 해킹이나, 정보유출 등이 쉽게 이뤄질 수 있는 통로가 늘어난다는 점에 그 심각성이 있다. 2018년 화웨이 제품에서 확인된 보안 취약점을 살펴보면 네트워크 장비부터 서버·소프트웨어(SW)·운영체제(OS) 등 다양한 제품에 망라돼 있다. 특히 오버플로우 취약점만 41개에 달했다. 오버플로우 취약점의 경우 이를 통해 관리자 권한의 획득, 정보유출, 네트워크 교란 등이 가능한 것으로 알려져 있다.

미국은 일찌감치 중국산 제품의 보안문제를 내세우며 화웨이 제품의 미국 내 도입을 금지하고 있다. 2012년 미국 하원이 화웨이 장비가 미국 안보 시스템을 공격할 수 있다는 보고서를 발표했고, 2016년 미국에서 판매중인 화웨이 스마트폰에서 백도어가 발견돼 이를 뒷받침했다. 또한 화웨이의 LTE 통신장비를 도입한 LG U+가 미군지역 주변에서 화웨이 통신장비를 사용하지 못하도록 했다.

영국도 최근 화웨이 통신장비에 대한 보안문

제를 제기하였다. 영국 정부는 "중국 화웨이의 모바일 광대역 장비의 엔지니어링 프로세스에 결함이 발견됐으며 영국 통신 네트워크에 새로운 안보위협이 될 수 있다"는 보고서를 내놨다(HCSEC, 2018). 영국 정부통신본부(GCHQ)와 정부 보안기관 관계자 등이 참여한 '화웨이사이버보안평가센터(HCSEC)'는 보고서를 통해 "이 문제를 해결하기 위해서는 '명확한 작업(significant work)'이 필요하다"며, "화웨이 장비에 대해 국가안보에 위협이 되지 않는다는 것에 대해 '제한적 보장'만 할 수 있다"고 밝혔다.

영국정부는 지난 2010년 영국 통신망에 화웨이 장비를 도입하는 것을 허용하는 조건으로 장비개선 보안성 평가를 위해 HCSEC를 설치하고 매년 평가보고서를 발표하고 있다. 앞서 세차례의 연례보고서에서는 "영국의 국가 안보에 대한 화웨이의 위험이 완화되었다"는 평가를 내놨었다. 보고서는 특히 "화웨이가 제품 소프트웨어 코드를 정확하게 사용했는지, 숨겨진 부분은 없는지 점검했지만, 화웨이의 프로세스는 업계의 모범에 미치지 못했다"며 감독기관의 보증을 어렵게 만들었다고 강조했다. 그러면서 "이같은 문제를 해결하는데 진전이 없다는 것에 실망스럽다"고 덧붙였다. 그러면서 "HCSEC이 지난해 중국 심천의 화웨이 본사를 방문했을 당시 화웨이가 제 3자 구성 요소 사용에 대한 적절한 감시를 하지 못하고 있음이 드러났다"고 밝혔다. 이는 화웨이 장비에 접속하는 제 3자 사업자 또는 중국정부의 '백도어'에 대한 보안감시가 미흡하다는 것을 의미해 미국, 호주 등이 주장해온 중국 스파이 기관의 침투 가능성을 뒷받침 한다.

III. 대응 방안

앞에서 언급한 것처럼 5G 이통통신 상용화와 관련하여 대용량의 데이터를 기반으로 다양하고 융합적인 서비스를 제공하는 국가기간망에 설치되는 통신장비를 외산 통신장비를 사용하는 것과 관련해 외산 통신장비의 보안 침해 사례가 발생할 수 있음을 확인하였다. 이러한 통신 네트워크 장비의 보안 취약성에 대비하기 위해서는 강화된 정보보호 평가기준을 적용하고 관련 보안 전문인력을 육성해야 한다.

이를 위해 먼저 통신장비의 보안과 관련하여 인식제고가 필요하다. 미국과 영국 등에서 통신망 장비의 정치적 보안위협과 안보 문제를 제기한 바 있다. 이와 관련 우리나라도 외산 통신장비를 많이 도입해 운영하고 있는 상황에서 제기될 수 있는 보안문제를 짚어보고 대책을 마련하여야 한다. 이제는 통신과 보안을 별개로 생각해서는 안 되고 새로운 패러다임에 맞추어 변화해야 한다. 가장 중요한 문제는 공공기관에서 사이버위협에 대해 몇 초 안에 대응한다는 구체적인 목표가 없다는 것으로 먼저 명확한 목표를 가지고 구조적인 보안문제를 논의한 다음에 국가안보의 시각에서 문제에 접근해야 한다.

네트워크 자체가 국가 안보와 얼마나 직결되느냐가 중요하다. 국가안보 이슈는 사이버 공간에서 먼저 발생될 수 있다. 이러한 상황에서는 네트워크 장비에 대한 국가안보 차원에서의 검증이 필요하고, 이를 위한 관련 보안산업 및 전문인력 육성이 필요하다. 화웨이 장비의 침해사례에서 본 것처럼 오버플로우 취약점으로 인해 관리자 권한의 획득, 정보 유출, 네트워크 교란 등이 가능하며, 라우터를 통한 정보유출, 무선공유기 해킹 등의 관련 취약점 발표로 인해 심각한 보안위협이 초래될 수 있음이 나타났다. 즉, 기간 통신장비부터 말단 통신장비까지 보안 취약점이 연이어 발표됐고, 특정 구간의 보안 취약점으로 인해 심각한 보안위협이 발생할 수 있다. 그러므로 이러한 상황에서 외산 장비와 기술에만 의존할 경우 통제가 쉽지 않기 때문에 국가기간망에 도입되는 통신 네트워크 장비에 대해서는 훨씬 강화된 보안성 검증이 필요하다. 공공기관의 네트워크 장비에 대해 좀 더 강화된 정보보호 기준을 적용해야 하고, 국내 기술 및 장비 활용방안도 제도적으로 모색해야 한다. 특히, 외산 장비보다 국산 장비가 보안이 더 잘 되어 있다는 부분은 검증되지 않았기 때문에 국산 제품의 보안성 평가를 더욱 투명하게 하고, 기능을 강화시켜 관련 산업을 육성해야 한다.

우리나라의 경우 국가정보호기본법에 의해 정보보호 시스템 또는 네트워크 보안관련 제도를 명시하고 있다. 이는 행안부 장관과 관련 기관장이 협의해 정보보호 시스템 성능과 신뢰도 관련 기준을 정해 고시하도록 하고 있다. 특히, 신뢰도 평가 부분은 CC인증 평가를 통해 수행하고 있지만 성능 측면은 이와 관련된 기준이 아직 없다. 그리고 정보보호 제품과 네트워크 제품에는 차이가 있어 별도의 평가기준이 마련해야 한다. 특히 네트워크 장비는 근본적으로 보안문제가 나올 수밖에 없다. 중요한 문제는 이를 어떻게 관리하고 보안 통제하느냐이다. 네트워크 보안장비에 대해서는 보안적합성 검증 제도와 CC인증, 암호화 모듈 검증 제도 등을 통해 평가하고 있지만 네트워크 장비인 스위치·라우터 등에 대해서는 검증이나 평가할 수 있는 기준이 없으므로 이러한 네트워크 장비도 믿고 사용할 수 있는 검증 평가 제도가 마련되어야 하며, 안전성을 보장할 수 있는 지속적인 유지관리체계가 마련되어야 한다.

[참고문헌]

[1] Cisco, CISCO Visual networking index global mobile data traffic forecast update, 2016- 2021(white paper), 2017. 3. 28.

IV.[2] HCSEC, Huawei Cyber Security Evaluation Centre(HCSEC) Oversight Board Annual Report 2018, July, 2018

5G 환경에서의 커넥티드카 서비스를 위한 확장 보안 프레임워크

고병수*

*(주)디지캡

Extended security framework for connected car service in 5G environment

Byoungsoo Koh*

*DigiCAP Co., Ltd.

요 약

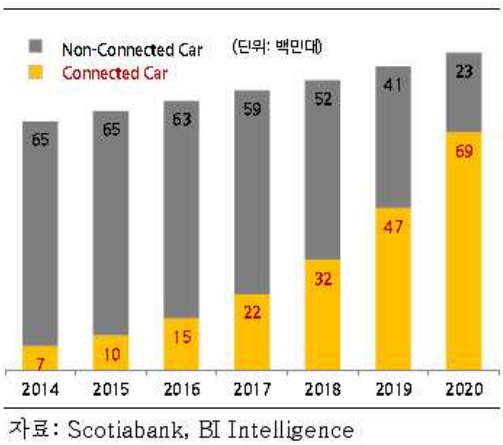
최근 5G 상용 서비스가 본격화됨에 따라 커넥티드카 환경에서 다소 제약적이었던 인포테인먼트 서비스등의 본격적인 발전이 기대되고 있다. 5G 기술과 더불어 인간, 사물등의 연결이 가능한 IoT 시대가 다가오고 있으며, 자동차도 하나의 IoT 기기처럼 포함되어 발전하고 있다. 커넥티드카는 기존 차량 내부 네트워크 중심의 서비스에서 4G/5G 네트워크를 활용한 다양한 서비스가 가능하다.

그러나 이러한 발전 과정 속에서 커넥티드카의 보안 및 다양한 콘텐츠의 유통에 따른 보안성의 문제는 해결해야할 숙제이다. 보안성이 확보되지 않고 단순히 서비스만 고려한다면 악의적인 공격을 통해 모든 서비스의 중단을 초래할 수 있다. 본 논문에서는 5G 통신 환경에서의 커넥티드카 서비스를 고려한 보안성을 지원하는 확장 프레임워크 활용 방안을 연구한다.

I. 서론

5G와 사물인터넷(IoT) 등 관련 기술의 발전과 함께 스마트 카 시장은 빠르게 성장할 것으로 전망되며, 이는 미래 IT환경 변화를 가속화하고 자동차 산업의 혁신을 주도할 것으로 예상되고 있다.

2015년 전세계 스마트 카 관련 산업은 매출액 기준 약 500억 달러 규모로 예상되며, 2020년경에는 약 1,600억 달러에 달할 것으로 전망되고 있으며, Gartner도 2020년에는 전 세계 주행 차량 다섯 대 가운데 한 대가 무선 네트워크로 연결된 차량으로, 초기 사물인터넷 시대의 주요 부분을 차지할 것으로 전망하고 있다.



(그림 1) 커넥티드카 세계 시장 전망

스마트 카는 자동차와 IT기술을 융합해 상시 네트워크에 연결된 차량을 의미하며, 차량, 인프라, 스마트 디바이스 등과의 실시간 소통을 통해 안전하고 편안한 운전 경험 제공을 추구한다.

스마트 카는 무선통신과의 연결을 통해 차량의 위치 추적, 원격 차량진단, 사고감지 등 안전 중심 기능을 주로 제공하는 '텔레메틱스(Telematics)'와 지도, 내비게이션 등 이용자 에게 정보와 편의를 제공하는 '인포테인먼트(Infotainment)' 등이 고도화된 개념으로 고유 장비에 맞춰 서비스와 콘텐츠가 고정되었던 기존 시스템의 장점을 계승하면서 소프트웨어와 콘텐츠 등의 개방성과 다양성이 더욱 확장된 것이다.

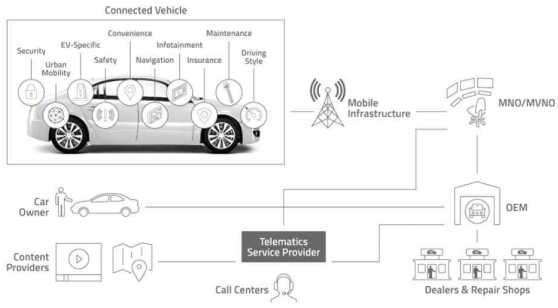
차량 제어·모니터링 서비스는 차량 내에 탑재되고, 미디어 컨텐츠 스트리밍 및 다양한 애플리케이션 서비스 등은 스마트폰과 연결해 이용하는 형태가 주를 이루 고 있으나, 궁극적으로는 모든 연결성, 플랫폼 및 솔루션이 차량 내에 탑재되어 자동차 자체가 하나의 '커넥티드 디바이스(Connected Device)'가 되는 형태로 진화하고 있다.

결국 차량 내·외부와의 강화된 연결성(Connetivity)을 통해 실시간 정보교환, 맞춤 형 컨텐츠 제공, 교통량 관리, 교통사고 방지, 위치 기반 서비스, 차량 상태 모니터링 ·관리 등이 용이하게 되어 이용자의 안전성과 편의성에 기여할 것으로 전망되고 있다.

이와 같은 스마트카의 서비스 영역중 차량의 제어 등과 같이 차량의 기능에 직접적으로 영향을 주는 서비스 뿐만 아니라, 멀티미디어 콘텐츠, 어플리케이션 등의 인포테인먼트 서비스까지 확장될 것으로 전망되고 있으며, 이에 멀티미디어 콘텐츠 등과 같이 스마트카에서 소비되는 콘텐츠에 대한 폰 커넥티비티 플랫폼을 지원하는 저작권 보호 기술의 개발이 요구되고 있는 상황이다.

1.1 인포테인먼트 플랫폼

커넥티드카(Connected Car)는 통신과 교통인프라 기술발전을 통해 도로를 인식하고 주변의 정보에 따라 차량을 제어하는 것을 넘어 다른 차량과의 통신 정보교환을 통해 도로 사정과 교통 상태 전반을 인식하고 주행하며 자동차에 IT기술을 융합하여 차량내의 네트워크 접속 기능이 포함되어 자동차내에서 다양한 정보 서비스 및 엔터테인먼트 서비스를 지원가능한 자동차이다.



자료 : Sierra Wireless

(그림 2) ICT 기반 커넥티드카 운용 개념도

사물인터넷 기술의 발전으로 통신 기능을 보유한 사물·기기들이 증가하면서, 자동차와 다른 기기간의 상호 연결성 확대를 통해 편리한 서비스 제공할 수 있으며, 완성차 업체들은 초기 애플과 구글의 플랫폼을 탑재하는 것이 미온적이었으나, 이들이 제공하는 컨텐츠의 다양성과 기존 스마트폰과의 호환성을 경쟁력으로 인식하기 시작하면서 도입이 점차 확산되는 추세이다.

이와 관련하여 안드로이드 오토(Android Auto)와 애플 카플레이(Apple CarPlay)는 2016년부터 다양한 차량 제조사들이 적용하기 시작하고 있으며, 스마트폰으로 콘텐츠를 이용하는 데에 익숙한 사용자들에게 스마트폰이 제공하는 사용자 경험과 콘텐츠를 차량에서도 일관성 있게 경험할 수 있도록 도와주고 있다.

안드로이드 오토(Android Auto)와 애플 카플레이(Apple CarPlay)는 운전자의 스마트폰의 콘텐츠와 인터넷 데이터를 통해 차량용 인포테인먼트 시스템(IVI: In-Vehicle Infotainment)을 구현하고 있으며, 안드로이드 오토(Android Auto)와 애플의 카플레이(Apple CarPlay)에서 이용 가능한 콘텐츠는 각 플랫폼이 제공하는 네비게이션 애플리케이션과 음성명령 기능, 그리고 스마트폰에 저장된 멀티미디어 콘텐츠이다.

IT기업	화면	제휴 기업
애플 (CarPlay)		아우디, 폭스바겐 현대 등 40개사
구글 (androidauto)		벤츠, 포드, GM, 볼보 등 40개사
바이두 (CarLife)		아우디, 벤츠, 폭스바겐, 현대 등
LeTV (IoV주)		Aston Martin

주: IoV = Internet of Vehicle
자료: 각 사 및 언론기사 참조

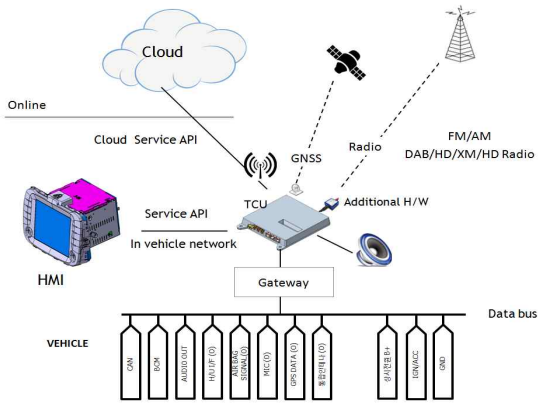
(그림 3) 주요 IT 기업의 인포테인먼트 플랫폼 개발 현황

서드파티(3rd party) 콘텐츠로는 아이하트라디오(iHeartRadio), 스포티파이(Spotify)가 있으며 애플의 카플레이(Apple CarPlay)는 애플사가 제공하는 애플리케이션만을 이용 가능한 반면, 안드로이드 오토(Android Auto)는 이용자가 애플리케이션을 추가하여 설치할 수 있으나 구글에서 안드로이드 오토(Android Auto)용 애플리케이션에 대해서는 추가로 사전 심의를 하고 있다.

1.2 확장 프레임워크

자율주행 차량을 위해 차량에 LTE, Wifi, BT등 무선 통신 기술을 적용하여, 이를 활용한 다양한 서비스를 고려하고 있음. 이로 인해 차량에서 소비할 수 있는 다양한 콘텐츠 확보가 필요할 것으로 예측된다.

자율주행 시대에 대비하여 사용자가 차량 내부에서 할 수 있는 다양한 행동을 분석하고 있으며, 이에 관련하여 인포테인먼트가 가장 현실적인 서비스로 인지하고 있다. 현재 차량 제조사와 1차 벤더(Tier 1)는 커넥티드카를 위한 IT 융합 기술 적용을 적극적으로 추진하고 있으며, 서비스개발과 보안 적용이 완료되는 시점에서 좀더 다양한 인포테인먼트 서비스를 추진할 것으로 예상된다.



(그림 4) 5G을 활용한 텔레매틱스 서비스 확장 프레임워크

Telematics 기능의 탑재율 증가 및 5G 차량 통신의 핵심 모듈로 활용도 요구되면서 처리능력(Computing Power) 강화 및 하이퍼바이져(Hypervisor) 기술 발전으로 다양한 서비스 수용 가능한 구조로 발전하고 있다. 이와 함께 차세대 서비스 기능의 확장이 가능하고 외부 고속 통신 네트워크를 활용한 다양한 서비스를 고려한 커넥티드카 통신모듈(Telematics)의 확장 프레임워크 기술이 대두되고 있다.

1.3 확장 보안 프레임워크

현재 커넥티드카 환경에서의 연결성은 사용자의 스마트폰을 이용한 폰 커넥티비티 방식, LTE망을 이용한 텔레매틱스 방식, 미러링 방식으로 구분된다.

<표 1> 인포테인먼트 보안 발전 방향

구분	기존	향후
DRM Watermarking	End-to-End 방식의 저작권 보호 기술 적용	Multi-DRM 및 Multi-Device를 위한 저작권 보호 기술 개발 필요 IoT 디바이스와 스마트기기간 콘텐츠 전송 보호 필요
지원 네트워크	LTE/Wifi	LTE/USB/Wifi/BT
개발 플랫폼	Windows Android iOS	Linux, HTML5, 차량 전용 플랫폼
콘텐츠 소비 방식	Service Provider 혹은 Content Provider 제공 형식 미디어 다운로드와 스트리밍 중심	차량 제조사 혹은 After market 제조사의 직접 서비스 향후 콘텐츠 소싱을 위해 오픈 서비스 모델 지원 필요 네트워크 발전에 따른 미디어 스트리밍 가속화
표준화	De Facto Standard	차량 전용 표준

차량 인포테인먼트 시장에서 다양한 콘텐츠 서비스가 확대될 것으로 예상됨에 따라 콘텐츠 저작권의 보호를 위한 기술 및 프레임워크의 개발이 필요하다. 현재 차량의 IoT 기기를 이용한 미러링 서비스가 있으나 이는 콘텐츠의 이동이 아닌 화면 복사 방식으로 사용의 제약 및 저작권 보호를 지원하지 못하고 있다. 차량 인포테인먼트 시장 발전과 더불어 다양한 콘텐츠의 유통 및 소비에 대해 저작권 보호 기술의 선개발, 선대응 필요성 대두되며, 커넥티드카에 사용되는 네트워크는 기존 방식과는 상이하거나 다양성이 존재하므로 관련 네트워크에 적응적인 라이센스 브로커나 라이센스의 안전한 저장을 위한 샌드박스(Sand-box), Copy Protection 등과 같은 저작권 기술의 확장 필요하다. 차량 전용 플랫폼을 사용하는 자동차 제조사 뿐만아니라 After Market 시장에서도 다양한 서비스가 가능하도록 저작권 유통 프레임워크 및 툴박스(Toolbox) 형태의 저작권 기술 개발이 필요하다.

- 커넥티드카 플랫폼에서의 콘텐츠 사적사용을 보장하는 Multi-Right 기술
- 차량용 IoT 디바이스에서의 콘텐츠 보호를 위한 암호화키 및 미디어 커넥터 기술 개발
- 차량 스마트 기기 지원 저작권 보호 응용 기술 개발
- 차량 콘텐츠 유통 활성화 및 저작권 호환성 보장을 위한 표준 기술 개발

(그림 5) 커넥티드카 지원 콘텐츠 저작권 보호 기술

II. 결론

본 논문에서는 5G 상용화를 앞두고 커넥티드카 환경에서의 다양한 서비스 플랫폼을 고려하여 보안성을 강화하기 위한 기술에 대해 서술하였다. 통신 기술과 ICT 융합기술의 발전으로 커넥티드카는 운전자의 편의성과 안전성을 증가시키고 있지만, 내외부의 연결성으로 인해 기존보다 좀더 지능적이고 복합적인 공격의 가능성이 높아졌다.

본 연구를 통해 커넥티드카 확장 프레임워크를 고려하여 인포테인먼트 서비스를 하고자할 때 발생할 수 있는 공격 패턴을 인식하고 기존 보안기술과 더불어 DRM, 인증, 암호화, 접근제어를 활용한 보안 요구사항을 도출하였다. 따라서, 본 논문에서 제시한 기술 개발을 통해 5G 기술 발전에 따른 커넥티드카에서의 안전한 서비스 확장 및 콘텐츠 유통등 인포테인먼트 환경에서 적용되길 기대한다.

Acknowledgments: This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program (10070156, Extended service framework for telematics in a

connected car environment) funded By the Ministry of Trade, industry & Energy (MI, Korea)

[참고문헌]

[1] www.obigo.com, 오비고

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

구두발표 트랙 E

캐시 부채널 공격 방법들과 발전에 관한 연구

이영한*, 김기열*, 방인영*, 전성일*, 백윤흥*,
*서울대학교 전기 정보공학부

A Study of cache side channel attack methodologies and their development

Younghan Lee*, Giyeol Kim*, Inyoung Bang*, Seong Il Jeon* and Yunheung Paek*

*Department of Electrical and Computer Engineering,

Seoul National University

Abstract

In the recent development of cache side channel attack on major CPU architectures has enlightened us of the power and its application. Meltdown and Spectre attack have been introduced to the world earlier this year which caused a pandemonium showed that it can degrade the performance of CPUs hugely. In this paper, I will present the basic idea of such attacks and different variations of cache side channel attacks. Furthermore, few challenges which are brought about by applying the attacks in platforms other than x86 such as ARM or Android devices will be explained in detail.

I. Introduction

In order to understand the term "cache side channel attack", it is important to firstly understand what "side-channel attack". As the name suggests, any attack based on information gained or leaked from implementation of a computer system is called "side-channel attack". There are many types of such attacks, timing attacks, fault attacks, and power attacks to name a few. Therefore, "cache side channel attack" will take advantage of cache in a computer system to leak its information of sensitive data. Most popular attack methodologies are the following: Flush+Reload, Prime+Probe. They will be explained in Section III with details. Applying such attacks in different platforms as ARM or Android devices is associated with several challenges due to difference in their instruction set and cache inclusiveness. They can be handled with empirical approaches and they will be explained in Section IV. Software Guard Extensions (SGX) which is an isolation mechanism implemented to protect code and data from their leakage. Address-space layout randomization (ASLR) is also the defense against memory-related vulnerabilities which, in short, selects a random locations in the virtual address for placing code and data. Attacks against these security features will also be presented in the same section.

II. Background

2.1 Recent Issues

It is surprising that the cache side channel attack has been around for so long until we acknowledged its strength in perpetrating the security defense. Much information is actually stored in cache until it has been removed or replaced by other data which provides an attacker with reasonably large window of attack surface. Spectre and Meltdown attacks are at the peak of such exploitation causing chaos in the industry.

2.2 Spectre Attacks

There are several variant of this attack, however, in this sub-section I will only introduce variant 1 of spectre attack. the main idea behind it is the speculative execution of CPU. CPU, as designed in modern era, tries to predict the future by branch predictor. It predicts which path the execution will branch to based on events learned in the past. For example, as shown in Figure 1, branch predictor at if-statement will predict which path the execution will take for fast execution. "KEY" in data is the sensitive information which is prevented from accessing with bounds-check. Now lets presume that we increment index from 0 to 7. From the first three events the predictor

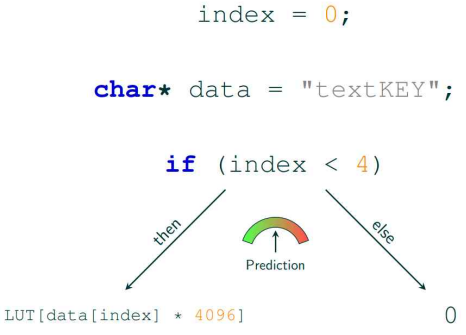


Figure 1

will lean toward "then" path when predicting the future. Hence, even if when index become four, the predictor will take "then" path and which will leave the trace in the cache. In conclusion, by influencing the CPU to mispredict the future, we can reveal the sensitive data from cache.

2.3 Meltdown Attacks

The main idea behind this powerful attack is utilizing out-of-order execution. Instructions are fetched and decoded in the front-end and dispatched to the back end. This implies that instructions are executed out-of-order meaning that the instructions wait until their dependencies are ready and eventually they retire in-order. In case of reading memory, if an application reads memory, it will first check permissions and if granted should the data be loaded otherwise crash. However, since instructions can be executed out-of-order, what happens if the permission checks are reordered? When tested, it was discovered by Flush+Reload that "unreachable" code line was actually executed due to the out-of-order execution. These instructions are named "transient instruction". The attackers managed to real the sensitive data with some smart coding of making the sensitive data the index of cache and monitoring cache hit or miss.

III. Cache side channel attack

3.1 Flush+Reload

This particular type of attack leverages the fact that Last Level Cache(LLC) is shared between cores. Also, it works on cache lines. In Figure 2, cache is assumed to be shared cache and in the first step attack will maps the memory to the shared cache. Secondly, attacker flushes the

cache line. Thirdly, victim happens to load the data to the same cache line while performing encryption. Finally, we measure the reload time of the cache line. If the victim loaded the line, it will cause cache hit with fast access.

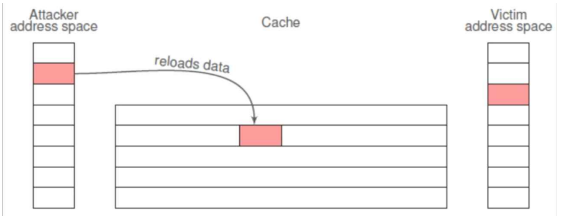


Figure 2

3.2 Prime+Probe

This is similar to the previous attack. However, it works on cache sets and the attacker needs to know virtual-to-physical mapping of the victim. Hence, there is no requirement for shared LLC. Attacker chooses a cache-sized memory buffer and accesses all the lines in the buffer, filling the cache set with its data. As victim executes, some of cache lines will be evicted. Eventually, attacker will measure the time to access the buffer. Evicted cache line will cause cache miss.

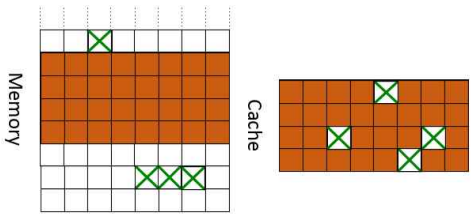


Figure 3

IV. Development of cache side channel attacks

4.1 Attack on ARM

ARM has different instruction set than x86 computer system. As we attempt to exploit Flush+Reload attack on ARM architecture, there are mainly two features we need to empirically prove for success. In Intel system, "clflush" instruction is used for unprivileged cache flush instruction. On top of this, cache inclusiveness is mandatory requirement for such attack. However, on ARM system, neither of this is clearly known. By carefully designing experiment, we could confirm that cache inclusiveness of last level cache is valid in Cortex A57 and A53. Moreover, "clearcache" system call has been proved to be suited for the purpose of flushing cache.

4.2 Attack against SGX

SGX is introduced with Skylake micro architecture which protects the execution of user programs in "enclave" that is isolated memory region for code and data. Here, we can utilize two techniques to bypass this security measure. Firstly, high-resolution timer such as "rdtsc" and "rdtscp" instructions are not permitted inside the enclave. Therefore, we must use global variable as a timing source. In other words, manual timer must be created. Secondly, Prime+Probe attack should be employed with the right eviction sets. "isgx_page_cache_init" creates a list of available physical pages and if a physical page is mapped duo to a page fault, it will remove and return the head of the list.

4.3 Attacking ASLR

In modern CPU architecture, Translation Lookaside Buffer(TLB) inside the

Memory Management Unit(MMU) maps the virtual memory address to the physical address. If there is a miss in TLB, Page Table walk will begin and we wish to leak this trace information. The scenario remains pretty much the same here as the previous cases. Precise timer is used to distinguish the cache hit and miss. SharedArrayBuffer() which allows the memory to be shared between multiple worker thread. By separating operation thread and timer thread, we can create a precise timer. Moreover, in this case, using Prime+Prove attack will be used to observe the effect of MMU's PT walk.

V. Conclusion

In this paper I have illustrated numerous types of cache side channel attack and their development for the purpose of expanding the scope of attack surface to difference platforms. Even though many have tried to prevent attack by reinforcing the security features such as SGX and ASLR. the attackers managed to find the way around them with attacks described in this paper. Preventing cache side channel entirely would be unrealistic due the overhead which the defense mechanism will cause to the computer system. However, it is important to be aware of such imminent treats of information leak and take appropriate precautions.

VI. ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Koreagovernment(MSIT)(NRF-2017R1A2A1A17069478). The ICT at Seoul National University provides research facilities for this study. This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-2015-0-00403) supervised by the IITP(Institute for Information & communications Technology Promotion)

[References]

[1] Schwarz, M., Weiser, S., Gruss, D., Maurice, C. and Mangard, S., 2017, July. Malware guard extension: Using SGX to conceal cache attacks. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 3-24). Springer, Cham.

[2] Gras, B., Razavi, K., Bosman, E., Bos, H. and Giuffrida, C., 2017. ASLR on the Line: Practical Cache Attacks on the MMU. NDSS (Feb. 2017).

[3] Oren, Y., Kemerlis, V.P., Sethumadhavan, S. and Keromytis, A.D., 2015, October. The spy in the sandbox: Practical cache attacks in javascript and their implications. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 1406-1418). ACM.

[4] Green, M., Rodrigues-Lima, L., Zankl, A., Irazoqui, G., Heyszl, J. and Eisenbarth, T., 2017, March. AutoLock: Why Cache Attacks on ARM Are Harder Than You Think. In 26th USENIX Security Symposium.

ChipWhisperer를 이용한 패스워드 기반 인증 방식의 단순 전력 분석 검증1)

최낙준* 이지은** 김광조*,**
KAIST *정보보호대학원 **전산학부

Simple Power Analysis on Password-based Authentication using ChipWhisperer

Nakjun Choi* Jeeun Lee** Kwangjo Kim*,**
*Graduate School of Information Security **School of Computing, KAIST

요 약

사용자가 입력한 문자열과 서버에 저장된 문자열의 일치여부만을 확인하는 패스워드 기반 인증 방식은, 그 간편성으로 인해 인터넷상의 각종 정보통신 서비스에서 광범위하게 사용되고 있다. 따라서 우리는 패스워드 기반 인증 방식의 서버와 사용자 간 전송 채널 상에서 획득한 부채널 정보(특히, 전력 소모 정보)를 이용하여 공격이 가능함을 검증하고자 한다. 본 논문에서는 캐나다 NewAE Technology(사)의 ChipWhisperer와 단순 전력 분석(Simple Power Analysis) 스크립트를 이용하여 부채널 공격을 수행한 결과, 간단하게 공격이 가능함을 보이고 추후 연구 방안을 제시한다.

I. 서론

패스워드 기반 인증 방식은 개인의 정보보호를 위한 다양한 방법 중 가장 기초적이고 간단한 방법이다. 추가적인 인증 없이 문자열의 일치 여부만 확인하는 간편함 덕분에 온라인 계정 로그인, 디지털 도어락, 은행 계좌 관리 등 온·오프라인을 가리지 않고 광범위하게 사용되고 있다. 하지만 패스워드 기반 인증 방식이 널리 사용되는 만큼, 이를 무너뜨리기 위한 새로운 공격 방법 또한 꾸준히 개발되고 있다. 만약 패스워드 기반 인증 방식이 이러한 공격에 취약한 내부적 결함을 가지고 있다면 이는 사회 전반적인 보안에 직접적인 영향을 미칠 것이 분명하다. 따라서 높은 안전성을 갖춘 고성능 패스워드 기반 인증 방식을 개발하는 것이 매우 중요하다.

웹 보안의 경우, 사용자가 패스워드 인증 방식을 통과하더라도 추가적인 인증 과정을 거쳐야 로그인이 가능하도록 시스템을 구성하는 경우가 있지만, 대부분의 하드웨어 기반 패스워드 인증 방식은 패스워드가 일치할 경우 곧바로 데이터에 접근이 가능하다. 또한 하드웨어의 물리적인 특성으로 인해 타이밍 공격(Timing Attack) 또는 오류 주입 공격(Fault Injection Attack) 등 부채널 공격의 위험에 항상 노출되어 있다. 따라서 패스워드 기반 인증 방식의 보안성이 인정받기 위해서는 해당 시스템이 부채널 공격에도 안전함을 보여야 한다.

본 논문에서는 캐나다의 NewAE Technology(사)에서 제공하는 패스워드 기반 인증 방식과 단순 전력 분석(Simple Power Analysis, SPA) 스크립트를 이용하여 부채널 공격을 구현해보고 이를 토대로 부채널 공격 대응 연구의 중요성에 대해 논한다 [1].

1) 이 논문은 SK Telecom Network R&D Center의 지원을 받아 수행된 연구임.

본 논문의 구성으로 Ⅱ장에서는 부채널 공격에 속하는 전력 분석 공격과 타이밍 공격에 대해 간단하게 설명하고, Ⅲ장에서는 실험을 진행한 환경에 대해서 서술한다. Ⅳ장에서는 이번 실험의 공격 대상인 패스워드 기반 인증 방식에 대해 설명하고, Ⅴ장에서 구체적인 공격 방법 및 결과에 대하여 서술한다. 마지막 Ⅵ장에서는 결론 및 이를 토대로 한 향후 연구 계획에 대해 논한다.

II. 배경지식

2.1 전력 분석 공격

[2]에서 처음 소개 된 전력 분석 공격은 공격 대상에 직접적인 손상 또는 변형을 가하지 않기 때문에 수동(Passive) 공격 및 비침입 공격으로 분류된다. 전력 분석 공격은 크게 단순 전력 분석과 차분 전력 분석(Differential Power Analysis, DPA)으로 나뉜다.

단순 전력 분석은 암호 장치의 연산 과정에서 소비되는 전력 변화를 측정 후 분석하여 비밀키 등의 유용한 정보를 얻어내는 공격 방법이고, 차분 전력 분석은 측정된 전력 소모량의 통계적 특성을 비교하여 비밀키를 추정해내는 방법이다. 본 논문에서는 단순 전력 분석 공격을 통하여 실험을 진행하였다. 전력 분석 공격을 막기 위해서는 연산에 소모되는 소비전력에 무작위성을 부여하거나 중간 값을 감추는 마스킹 기법 등이 사용되어야 한다.

2.2 타이밍 공격

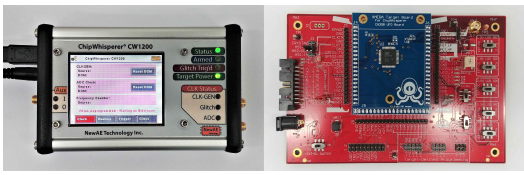
타이밍 공격은 알고리즘 연산에 소요되는 시간을 측정하여 유용한 정보를 얻어내는 공격으로서 부채널 공격 중 가장 기본이 되는 공격 방법이다. [3]에서 처음 소개된 타이밍 공격은 전력 분석 공격과 마찬가지로 수동 공격 및 비침입 공격으로 분류된다.

비교적 간단한 방법으로 공격을 구현할 수 있지만 연산과정에 의도적인 시간 지연을 발생시키는 방법으로 대부분의 타이밍 공격에 대한 방어가 가능하다. 본 논문에서 사용할 패스워드 기반 인증 방식 또한 타이밍 공격에 대한 방어 기법을 갖추었다.

III. 실험 환경

3.1 실험 장비 및 소프트웨어

본 실험에서는 캐나다의 NewAE Technology(사)에서 제작한 ChipWhisperer-Pro (CW1200)를 사용하여 전력 분석 공격을 구현하였다 [4]. ChipWhisperer-Pro는 오픈 소스 임베디드 보안 분석 플랫폼으로서 FPGA(Field Programmable Gate Array) 기반 펄스를 생성하여 클록 및 전압 오류 주입 공격 등 다양한 부채널 공격이 가능하도록 해준다.



[그림 1] ChipWhisperer-Pro(CW1200, 왼쪽), CW308 UFO 보드와 CW308T XMEGA 타겟 보드(오른쪽)

실험에 사용한 패스워드 기반 인증 방식은 프로그래밍이 가능한 FPGA 보드 중 하나인 CW308T XMEGA 타겟 보드에 소스코드를 업로드 하는 방법으로 구현하였다. [그림 1]은 우리가 실제로 실험에 사용한 ChipWhisperer 장비와 FPGA 보드를 나타낸다.

또한 소모되는 전력을 기록 및 분석하기 위해서 NewAE Technology(사)에서 기본적으로 제공하는 'ChipWhisperer Capture' 오픈 소스 소프트웨어를 사용하였다.

3.2 공격 모델

본 실험에서는 공격 대상인 패스워드 인증 방식의 소스코드를 공격자가 모두 확인 할 수 있고 이를 이용하여 전력 소모량이 변화하는 지점을 찾아내었기 때문에 화이트 박스 모델에 가깝다고 할 수 있다. 하지만 공격자가 소스코드에 접근 할 수 없다고 하더라도, 패스워드의 첫 문자에 전수공격을 가하는 방법을 통해 전력 소모량이 변화하는 구간을 찾아낼 수 있다. 따라서 블랙 박스 모델에도 동일한 공격 방식을 충분히 적용시킬 수 있지만 본 실험에서는 다루지 않았다.

IV. 패스워드 기반 인증 방식

본 실험에서는 NewAE Technology(사)가 기본적으로 제공하는 패스워드 인증 방식 소스코드를 토대로 시스템을 구현하였다. 원활한 실험을 위하여 다섯 자리의 문자열인 'ciscw'를 최초 패스워드로 설정 한 뒤, CW308T XMEGA 타겟 보드에 해당 소스코드를 업로드 하였다. [그림 2]는 패스워드 인증 방식의 실행 결과를 나타낸다. 사용자가 잘못된 비밀 번호인 'abcd'를 입력할 경우엔 패스워드가 일치하지 않는다는 메시지를 출력하고, 올바른 패스워드를 입력할 경우엔 접근이 허용되었다는 메시지가 출력된다.

[그림 2] 패스워드 인증 방식의 실행 결과

본 실험에서 사용한 패스워드 인증 방식은 입력 받은 패스워드를 1 Byte씩 비교하는 알고리즘을 취하고 있다. 즉, 사용자가 'abcd'를 입력하였을 경우, 패스워드 인증 시스템은 'abcd' 전체가 아닌 'a' 하나의 일치여부를 먼저 확인한다. 불일치할 경우 나머지 4 Byte의 확인 과정을 생략할 수 있기 때문에 연산 속도를 높일 수 있다는 장점이 있지만, 이는 연산 시간과 전력소비의 차이를 발생시키는 원인이 되기 때문에 타이밍 공격과 전력 분석 공격에 취약해질 수 있다. 본 실험에서 사용한 패스워드 인증 방식은 타이밍 공격에 대응하기 위하여 패스워드 불일치 시 랜덤한 시간 지연을 발생시키는 소스코드를 탑재하고 있고 이는 [그림 3]과 같다. 하지만 전력 분석 공격에 대한 대응 기법은 적용 되어 있지 않기 때문에 본 실험에서는 해당 패스워드 인증 방식에 단순 전력 분석 공격을 수행하였다.

```
if (passbad){
    //Stop them fancy timing attacks
     int wait = rand() % 100000;
    for(volatile int i = 0; i < wait; i++){
        ;
    }
    delay_200_ms();
    delay_200_ms();
    my_puts("PASSWORD FAIL₩n");
    led_error(1);
```

[그림 3] 타이밍 공격에 대응하기 위한 랜덤 시간 지연 소스코드

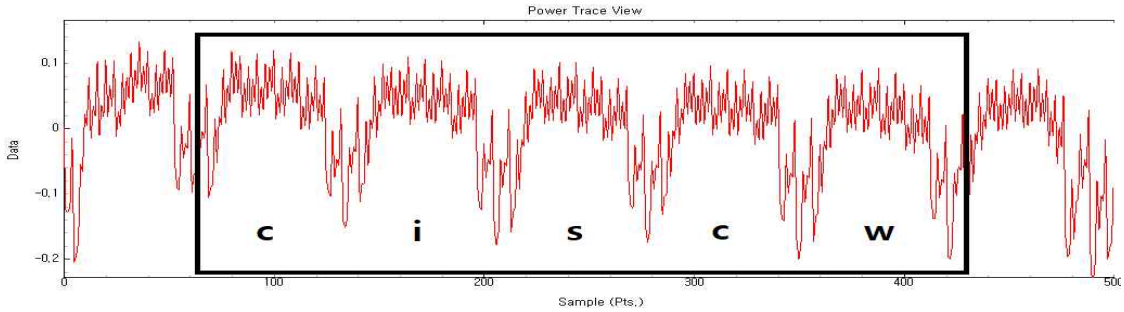
V. 단순 전력 분석 공격

우리는 ChipWhisperer Capture 소프트웨어를 이용하여 패스워드 인증 방식의 연산 시 소모되는 전력을 측정하였다. [그림 4]는 올바른 패스워드인 'ciscw'를 입력했을 때의 전력 소모량을 나타낸다. 시간이 흐름에 따라 전력소모량이 변화하는 것이 명확하게 나타나기 때문에 단순 전력 분석 공격을 통해 유용한 정보를 얻어 낼 수 있음을 쉽게 예상할 수 있다.

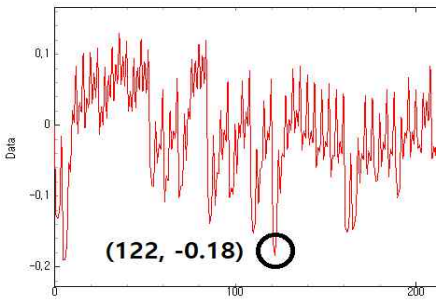
실제 공격을 구현하기 위하여 우리는 NewAE Technology(사)가 제공하는 단순 전력 분석 공격 스크립트를 사용하였고 공격 방법은 다음과 같다 [5].

1) 전력 소모 그래프에서의 타겟 포인트 결정
2) 패스워드의 첫 번째 Byte가 일치할 때와 일치 하지 않을 때의 타겟 포인트 변화량 측정
3) 매 Byte 마다 타겟 포인트 지점 전력이 일정 값을 초과 하는지 확인
4) 초과할 경우 올바른 Byte, 미만일 경우 잘못된 Byte로 판별
5) 모든 Byte를 진행 후 최종 패스워드를 출력

패스워드를 찾아내기 위해선 먼저 전력 소모 그래프에서 확연하게 구별이 가능한 타겟 포인트를 결정해야 한다. [그림 5]는 패스워드의 첫 번째 Byte가 일치하지 않을 때의 전력 소모량을 나타내는데, 검은색 원으로 표시한 122 포인트 지점이 상대적으로 낮은 전력소모량을 보이기 때문에 타겟 포인트로 결정하였다.



[그림 4] 올바른 패스워드 입력 시 전력 소모 그래프



[그림 5] 첫 번째 Byte 불일치 시 타겟 포인트

패스워드의 첫 번째 Byte가 일치하면 타겟 포인트는 194 지점으로 이동하는데 이를 통해 타겟 포인트 변화량이 74인 것을 알아 낼 수 있다. 즉, 우리는 122 포인트를 시작으로 74 포인트씩 이동한 지점의 전력 소모량이 일정 값을 초과하는지 확인해보는 방법을 통해 패스워드 일치 여부를 결정 할 수 있다. 본 실험에서는 측정되는 전압 -0.15mV를 기준으로 패스워드를 판별하였다.

타겟 포인트 변화량을 측정한 이후에는 단순 전력 분석 공격 스크립트 의해, 패스워드 인증 시스템에 'a'부터 '0'까지의 문자 및 숫자가 순차적으로 대입된다. 앞선 방법으로 패스워드의 첫 번째 Byte인 'c'를 찾아낸 후에는 다시 'ca'를 시작으로 'c0'까지 대입을 반복하며, 최종적으로 'ciscw'의 패스워드를 확정 한 후 공격이 종료된다.

만약 패스워드의 첫 번째 Byte가 일치하지 않았을 때 연산을 즉시 종료하지 않고 나머지 Byte의 비교 연산을 계속 수행하도록 알고리즘을 변경해준다면 성공적으로 단순 전력 분석 공격을 막아 낼 수 있다.

VI. 결론 및 향후 연구

본 논문에서는 NewAE Technology(사)에서 제공하는 패스워드 인증 방식과 단순 전력 분석 공격 스크립트를 이용하여 부채널 공격 방법을 구현하였다. 공격자가 패스워드 인증 방식의 전력 소비량 변화를 확인할 수 있다면 간단한 공격을 통하여 패스워드를 획득 할 수 있음을 보여주었다. 이는 패스워드 인증 방식을 광범위하게 사용하는 현대 보안산업에 큰 위협이 될 수 있다. 따라서 다양한 부채널 공격에 대응 할 수 있는 기법을 개발 및 적용하는 것이 매우 중요하다.

향후 연구 과제로는 난수 발생기 중 하나인 Xorshift128+에 다양한 부채널 공격을 시도하여 난수성을 낮추는 연구를 진행할 예정이다.

[참고문헌]

[1] NewAE Technology, https://newae.com/

[2] Power analysis attack, http://softknow.com/up2/file/20130930/20130930141438_0781.pdf

[3] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, *Annual International Cryptology Conference*. Springer, 104-113. 1996.

[4] ChipWhisperer-Pro, https://wiki.newae.com/Main_Page. https://newae.com/tools/chipwhisperer/

[5] https://wiki.newae.com/Tutorial_B3-1_Timing_Analysis_with_Power_for_Password_Bypass

선형 연산 특성을 이용한 전력 분석 최적화 방안*

박한별[1], 심보연[1], 문재근[2], 한동국[2,3]

[1]국민대학교 수학과, [2]국민대학교 금융정보보안학과, [3]국민대학교 정보보안암호수학과

Optimization for Power Analysis using Characteristic of Linear Operation

Han-Byeol Park[1], Bo-Yeon Sim[1], Jae-Geun Moon[2], Dong-Guk Han[2,3]

[1]Dept. of Mathematics, Kookmin University,
[2]Dept. of Financial Information Security, Kookmin University,
[3]Dept. of Information Security, Cryptology, and Mathematics, Kookmin University.

요 약

부채널 공격 기법(Side Channel Attack, SCA) 중 하나인 상관 전력 분석(Correlation Power Analysis, CPA)은 선형 연산을 공격지점으로 하여 수행하면 right key와 동일한 상관계수를 가지는 ghost key가 항상 발생한다. 반면 비선형 연산을 공격지점으로 하면 ghost key가 발생하지 않는다. 본 논문에서는 ghost key와 right key의 발생 위치를 이용하여 상관 전력 분석 위치를 집약적으로 선택함으로써 공격 복잡도를 줄일 수 있는 방법을 제안한다.

I. 서론

부채널 공격(Side Channel Attacks, SCA)이란 암호 알고리즘이 동작하는 동안 발생하는 전력 소비, 빛, 소리 등의 정보들을 기반으로 하여 비밀 키를 분석하는 공격 방법이다. 전력 분석 공격에는 단순 전력 분석(Simple Power Analysis, SPA), 차분 전력 분석(Differential Power Analysis, DPA), 상관 전력 분석(Correlation Power Analysis, CPA) 등 이 있다.

전력 분석 공격은 암호 알고리즘이 동작할 때 측정된 전력 소비 패턴과 전력 소비 모델의 통계적 특성을 분석하여 비밀 정보를 찾아내는 방법이다. 일반적으로 차분 전력 분석과 상관 전력 분석은 아날로그 전력 파형을 디지털 파형으로 샘플링한 후 샘플링 된 포인트의 값들의 통계적 특성에 기반하여 비밀 정보를 찾는 방법이다. 차분 전력 분석 혹은 상관 전력 분석에서 사용하는 포인트 수가 많을수록 공격에 필요한 시간이 증가한다.

본 논문에서는 차분 전력 분석이나 상관 전력 분석을 수행할 때, 선형 연산과 비선형 연산의 ghost key 특성을 기반으로 공격에 필요한 포인트 수를 감소시키는 방안을 제안한다. 즉, 공격에 필요한 포인트 수를 줄임으로써 차분 전력 분석 혹은 상관 전력 분석의 공격 복잡도를 줄일 수 있다.

ghost key의 발생 여부를 통해 전력 파형에서 선형 연산과 비선형 연산을 파악할 수 있음을 검증하기 위해 Tower Field를 기반으로 하여 설계된 S-Box가 적용된 AES-128 암호 알고리즘을 대상으로 설정하여 실험을 진행하였다.

본 논문은 2장에서 Tower Field를 기반으로 AES S-Box를 설계하는 원리를 제시한 후 3장을 통해 설계 단계에서 선형 연산이 사용되는 부분을 공격 지점으로 했을 때 발생하는 ghost key와 right key의 관계를 이론적으로 정리한다. 그 후 4장에서 실험을 통해 검증하여 5장에

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00520, SCR-Friendly 대칭키 암호 및 응용모드 개발)

서 결론을 도출한다.

II. 관련 연구

Table 1.은 본 논문에서 사용되는 기호들을 정의한 것이다.

Table 1. 기호 정의

기호	의미
δ	동형사상 $\delta: GF(2^8) \rightarrow GF(((2^2)^2)^2)$
k	비밀 키
x	평문 집합 ($x = \{x_1, \cdots, x_N\}$)
T	소비 전력 파형 집합 ($T = \{T_1, \cdots, T_N\}$)
RK	right key
GK	ghost key
f	임의의 연산
f_L	선형 연산 ($f_L \in f$)
$f(x,k)$	$f(x_i \oplus k)$ 집합 ($1 \le i \le N$)
$hw(f(x,k))$	$f(x_i \oplus k)$ 해밍 웨이트 값의 집합
$Corr(X,Y)$	집합 X와 Y의 상관계수

2.1 상관 전력 분석

상관 전력 분석은 고정된 비밀 키 k와 서로 다른 평문 x_i를 입력으로 암호모듈이 반복 수행되는 동안 수집한 다수의 전력 신호를 통계적으로 분석하여 비밀 키 k의 값을 찾는 방법이다 ($1 \le i \le N$). 암호모듈이 수행되는 동안 소비되는 전력 패턴 T가 연산되는 데이터 값 $f(x,k)$에 의존한다는 사실에 근거하며, 소프트웨어로 구현된 암호 알고리즘의 경우 일반적으로 전력 소비 모델이 중간 값 $f(x,k)$의 해밍 웨이트(hamming weight) 정보를 따른다는 것을 가정으로 한다.

이때 AES-128에서는 중간 값 $f(x,k)$의 모든 정보가 128비트이지만 분할 정복(divide and conqure) 방법을 적용하여 8비트 단위로 분석을 수행한다. 즉, 8비트 단위 부분 키 조사를 통해 전체 128비트의 비밀 키 정보를 획득할 수 있다.

따라서 상관 전력 분석 기법은 알려진 정보 x와 비밀 정보 k값을 추측하여 계산된 중간 값의 해밍 웨이트 $hw(f(x,k))$값과 전력 패턴 T의 피어슨 상관계수 $Corr(hw(f(x,k)), T)$를 계산하여 상관관계가 높을 때의 추측 키 k를 올바른 키라고 추정한다.

2.2 Tower Field기반 AES의 S-Box

본 논문의 대상이 되는 Tower Field 기반의 AES S-Box는 다음과 같은 6가지의 단계를 통해 설계할 수 있다[1]. S-Box의 구조에 의해 역원 연산이 수행되는데, $GF(2^8)$에서의 x의 역원은 $x^{-1} = x^{254}$ 이지만 $GF(2^4)$에서의 역원은 0이 아닌 y에 대하여 $y^{-1} = y^{14}$이므로 연산 소비량을 줄이기 위해 Tower Field를 이용한다.

step1. 먼저 S-Box의 입력 값인 $GF(2^8)$의 원소가 들어오면, 이것은 동형사상인 δ연산에 의해서 $GF(((2^2)^2)^2)$의 원소로 변환된다. δ와 δ^{-1}는 다음과 같다.

$$\delta = \begin{pmatrix} 10100000 \\ 11011110 \\ 10101100 \\ 10101110 \\ 11000110 \\ 10011110 \\ 01010010 \\ 01000011 \end{pmatrix} \qquad \delta^{-1} = \begin{pmatrix} 11100010 \\ 01000100 \\ 01100010 \\ 01110110 \\ 00111110 \\ 10011110 \\ 00110000 \\ 01110101 \end{pmatrix}$$

step2. S-Box의 입력 x에 대하여 동형사상 δ에 대한 함수 값을 $A = \delta(x)$라고 하자. A^{-1}를 구하기 위해서 먼저 A^{16}을 계산한다.

step3. A와 A^{16}을 이용하여 A^{17}을 계산한다.

이 때 $\alpha \in GF((2^n)^m)$, $r = \dfrac{2^{nm}-1}{2^n-1}$에 대하여 $\alpha^r \in GF(2^n)$라는 성질이 있다. $n=4, m=2$인 경우에 $A^{17} \in GF(2^4)$라는 사실을 알 수 있다.

step4. A^{17}의 역원 $(A^{17})^{-1} (= (A^{17})^{14})$을 계산해 준다.

step5. A^{16}과 A^{-17}을 이용하여 A^{-1}을 구한다.

step6. A^{-1}을 δ^{-1}를 이용해서 다시 $GF(2^8)$의 원소로 변환해주고 마지막 단계로 $Affine$변환을 통해 S-Box의 출력 값을 생성할 수 있다.

III. 선형 연산의 Ghost key 성질

상관 전력 분석을 수행할 때, 중간 값을 선형함수 출력 값으로 선택하는 경우 항상 right key와 동일한 상관계수를 가지는 ghost key가

존재한다. 반면, 비선형 함수의 출력 값으로 선택한 경우는 ghost key가 존재하지 않는다. 따라서 ghost key 발생 여부를 통해 전력 파형에서 연산 위치를 구분, 집약적인 상관 전력 분석 위치 선택을 통해 부채널 공격 복잡도를 줄일 수 있다.

본 장에서는 선형함수의 출력 값을 중간 값으로 설정하였을 때 발생하는 ghost key의 성질을 이론적으로 정리하고 4장에서 실험을 통해 검증한다.

정리 1. 8비트 해밍 웨이트 기반 상관 전력 분석 시, 상관계수는 다음과 같은 성질을 만족한다.

$$|Corr(hw(f_L(x,k)), T)| = |Corr(hw(f_L(x,k) \oplus 0xFF), T)|$$

증명. 두 변수 X와 Y의 상관계수는 다음공식을 이용하여 구할 수 있다.

$$Corr(X,Y) = \frac{N\sum XY - \sum X \sum Y}{\sqrt{(N\sum X^2 - (\sum X)^2)(N\sum Y^2 - (\sum Y)^2)}}$$

이 공식의 X의 값을 임의의 상수 a에 대하여, $a-X$로 변형하여 계산하여도 상관계수 공식의 성질에 의하여 $|Corr(X,Y)| = |Corr(a-X,Y)|$가 성립하게 된다. $f_L(x,y)$와 $f_L(x,y) \oplus 0xFF$는 비트가 반전되므로 8비트 해밍 웨이트 기반 상관 전력 분석에서 $hw(f_L(x,y))$와 $hw(f_L(x,y) \oplus 0xFF)$는 두 값의 합이 항상 8로 일정하다는 성질을 가지게 된다. 따라서 정리 1의 수식은 아래와 같이 표현된다.

$$\begin{aligned} &|Corr(hw(f_L(x,k) \oplus 0xFF), T)| \\ &= |Corr(8 - hw(f_L(x,y)), T)| \\ &= |Corr(hw(x,y), T)| \end{aligned}$$

□

정리 2. 8비트 해밍 웨이트 기반 상관 전력 분석 시, right key와 ghost key는 다음 성질을 만족한다. $RK \oplus GK = f_L^{-1}(0xFF)$

증명. 정리 1에 의하여, 다음이 성립한다. $|Corr(hw(f_L(x,k)), T)| = |Corr(hw(f_L(x,k) \oplus 0xFF), T)|$ 위 수식의 우변은 아래와 같이 나타낼 수 있다.

$$\begin{aligned} &|Corr(hw(f_L(x,k) \oplus f_L(f_L^{-1}(0xFF))), T)| \\ &= |Corr(hw(f_L(x,(k \oplus f_L^{-1}(0xFF)))), T)| \cdots (1) \end{aligned}$$

k를 right key 라고 하면 수식 (1)에 의하여 right key 이외에 $k \oplus f_L^{-1}(0xFF)$ 또한 동일한 상관 계수 값을 갖는다. 따라서 $k \oplus f_L^{-1}(0xFF)$를 ghost key로 정의하고, $GK = RK \oplus f_L^{-1}(0xFF)$라고 표현 할 수 있다. 즉, $RK \oplus GK = f_L^{-1}(0xFF)$라는 관계가 성립하게 된다. □

IV. 실험 결과

본 장에서는 Tower Field를 기반으로 한 AES-128 암호 알고리즘이 동작할 때 발생되는 소비 전력 파형을 수집하고, 선형 연산 부분을 공격포인트로 설정한 후 상관 전력 분석을 수행하여 ghost key와 right key의 관계를 수식적으로 보임으로써 **정리 1**과 **정리 2**를 실험적으로 검증한다.

4.1 δ연산에서 Ghost Key의 성질 검증

평문의 한 바이트가 비밀 키의 1바이트와 xor연산을 수행한 후 δ연산을 통해 $GF(((2^2)^2)^2)$의 원소로 변환되는 과정을 첫 번째 공격 지점으로 설정하였다. (이때 δ연산은 선형 연산이다.) 즉, **step1**부분의 출력 값을 공격 지점으로 설정한다.(2.2 참고)

δ연산은 선형 연산이므로 **정리 1, 정리 2**에 의해 $|Corr(hw(\delta(x,RK)), T)| = |Corr(hw(\delta(x,GK)), T)|$을 만족하는 right key와 ghost key가 동시에 존재함을 예측할 수 있다. 그리고 right key와 ghost key는 $RK \oplus GK = \delta^{-1}(0xFF) = 0x3D$라는 관계가 성립함을 예측할 수 있다. 즉, 이것으로 두 결과 값 $f_L(x,GK) = \delta(x \oplus GK)$ $f_L(x,RK) = \delta(x \oplus RK)$의 8개의 비트는 완벽하게 반전되므로, 두 값의 해밍 웨이트의 합이 8이 되는 것을 알 수 있다. 상관 전력 분석 결과는 Table 2.와 같다.

step1의 출력 값을 중간 값으로 하여 상관 전력 분석을 수행하였을 때, 항상 right key와

Table 2. $\delta(x \oplus Key)$를 중간 값으로 한 분석 결과

Byte	1	2	3	4	5	6	7	8
Right Key	0x2B	0x7E	0x15	0x16	0x28	0xAE	0xD2	0xA6
Ghost Key	0x16	0x43	0x28	0x28	0x15	0x93	0xEF	0x9B
RK⊕GK	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D
Byte	9	10	11	12	13	14	15	16
Right Key	0xAB	0xF7	0x15	0x88	0x09	0xCF	0x4F	0x3C
Ghost Key	0x96	0xCA	0x28	0x85	0x34	0xF2	0x72	0xD1
RK⊕GK	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D	0x3D

같은 상관계수를 가지는 ghost key가 존재한다는 것을 알 수 있었다. 예측했던 것과 같이 실제 분석 결과에서도 right key와 ghost key를 xor연산하면 $RK \oplus GK = 0\text{x}3D$ 라는 항상 일정한 값을 얻을 수 있었다. 즉, **정리 1**과 **정리 2**를 실험적으로 검증할 수 있다.

4.2 δ^{16}연산에서 Ghost Key의 성질 검증

두 번째 공격 지점은 $GF(((2^2)^2)^2)$의 원소로 변환된 $\delta(x \oplus K)$가 16번의 제곱 연산이 일어나는 부분으로 설정한 후 상관 전력 분석을 수행하였다. 즉, step2부분의 출력 값을 공격 지점으로 설정한다.

δ연산이 선형 연산이므로, δ^{16}또한 선형 연산임을 알 수 있다. 따라서 **정리 1**과 **정리 2**에 의해 $|Corr(hw(\delta^{16}(x, RK)), T)| = |Corr(hw(\delta^{16}(x, GK)), T)|$을 만족하는 right key와 ghost key가 동시에 존재함을 예측할 수 있고, 두 값은 $RK \oplus GK = (\delta^{16})^{-1}(0\text{x}FF) = 0\text{x}D1$의 관계를 만족함을 예측할 수 있다. 즉, 이것으로 두 결과 값

$f_L(x, GK) = \delta^{16}(x \oplus GK)$, $f_L(x, RK) = \delta^{16}(x \oplus RK)$

은 $0\text{x}FF$만큼 차이가 나므로 8개의 비트가 완벽하게 반전되어 해밍 웨이트 합이 8이 된다. 상관 전력 분석 결과는 다음의 Table 3.과 같다.

step2의 출력 값을 중간 값으로 하여 상관 전력 분석을 하였을 때도 4.1에서와 마찬가지로 항상 right key와 같은 상관계수를 가지는 ghost key가 존재한다는 것을 알 수 있었다. 예측했던 것과 같이 실제 분석 결과에서도 right key와 ghost key를 xor연산하면 $RK \oplus GK = 0\text{x}D1$ 라는 항상 일정한 값을 얻을 수 있었다. 즉, **정리 1**과 **정리 2**를 실험적으로 검증할 수 있다.

Table 3. $(\delta(x \oplus Key))^{16}$을 중간 값으로 한 분석 결과

Byte	1	2	3	4	5	6	7	8
Right Key	0x2B	0x7E	0x15	0x16	0x28	0xAE	0xD2	0xA6
Ghost Key	0xFA	0xAF	0xC4	0xC7	0xF9	0x7F	0x03	0x77
RK⊕GK	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1
Byte	9	10	11	12	13	14	15	16
Right Key	0xAB	0xF7	0x15	0x88	0x09	0xCF	0x4F	0x3C
Ghost Key	0x7A	0x26	0xC4	0x59	0xD8	0x1E	0x9E	0xED
RK⊕GK	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1	0xD1

V. 기대효과

정리 1, **정리 2**를 기반으로 선형 연산을 공격 지점으로 하여 상관 전력 분석을 하면, 항상 right key와 동일한 상관계수를 갖는 ghost key가 발생하지만 비선형 연산을 공격 지점으로 하여 상관 전력분석을 하면 right key만 발생하게 된다는 특성이 있다. 이 특성과 4.1, 4.2에서의 결과를 이용하면 ghost key와 right key를 가지고 Known Key Analysis를 수행하여 각각의 상관계수 파형의 peak 모양을 비교해 볼 수 있다.

이때 right key는 비선형 연산 위치와 선형 연산 위치 모두에서 상관계수의 peak가 발생하지만 ghost key는 선형 연산의 위치에서만 peak가 발생할 것이다. 즉, 선형 연산과 비선형 연산의 위치를 구분할 수 있으므로 상관 전력 분석 위치를 집약적으로 선택할 수 있어서 공격 연산량을 줄일 수 있다. 따라서 본 논문에서 제안하는 방법은 공격의 효율성을 높이는 데에 기여를 할 수 있다.

[참고문헌]

[1] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic", CHES 2001, LNCS, vol.2162, pp171-188

Intel SGX와 캐시 부채널 공격에 대한 연구

김기열*, 이영한*, 전성일*, 김현준*, 백윤흥*,
*서울대학교 전기정보공학부

A Study on Intel SGX and Cache Side-channel Attacks

Giyeol Kim*, Younghan Lee*, Seongil Jeon*, Hyunjun Kim* and Yunheung Paek*
*Department of Electrical and Computer Engineering, Seoul National University

요 약

클라우드가 보편적으로 활용되면서 클라우드 서버에 정보를 저장하는 일은 일상이 되었고, 다른 기기에서 프로그램을 실행시키는 일은 빈번해졌다. 그러나 다른 컴퓨터에 정보를 저장하거나 프로세스를 실행시키는 것은 공격에 취약한 환경을 만들어 보안 위협을 만든다. 정보를 보호하는 기법은 여러 가지가 있으나 여기에서는 Intel SGX 하드웨어 feature를 설명하고 어떻게 데이터가 보호되는 지를 간략하게 소개한다. 또한 SGX를 무력화시키기 위한 캐시 부채널 공격을 소개함으로써 모든 보안 솔루션들은 시간이 지남에 따라 안전하지 않다는 것을 상기시킴과 동시에 추후 연구에 대한 방향을 제시한다.

I. 서론

현대로 컴퓨터가 발전해오면서 점점 다양한 환경에서 시스템이 구축되고 여러 어플리케이션들이 그 위에서 실행된다. 다양한 시스템이 만들어짐에 따라 그에 맞는 공격 기법들이 등장하였다. 중요한 정보를 보호하기 위한 창과 방패의 대결은 각각 서로의 공격력과 방어력을 증가시켰다. 공격자의 강한 공격에 대응하기 위해 보안 연구자들은 강력한 위협 모델(thread model)을 설정하고 이에 대한 방어책을 연구하는데, 공격자와 피해자(victim)가 같은 시스템에서 프로세스를 돌린다는 것이 그 중 하나이다. 이는 잘 실현될 것 같지 않지만 날이 갈수록 그렇지 않다는 것이 더 확실해진다. 오늘날에는 클라우드 환경이나 클라이언트 기기에서 프로그램을 돌리는 일이 잦기 때문이다. 클라우드 환경에 민감한 데이터를 저장하거나 어떤 프로세스가 돌고 있을지 모르는 클라이언트 기기에 데이터를 넘겨주는 것은 굉장히 위험한 일이다. 이를 Intel이 제공하는 하드웨어 기능인 Intel SGX를 통하여 해결할 수 있다. 본 논문에서는 Intel SGX에 대한 설명과 함께 오늘날 존재하는 보안 위협에는 어떤 것이 존재하고 이를 위해서 Intel SGX를 활용하는 방안을 제시한다.

II. Intel SGX

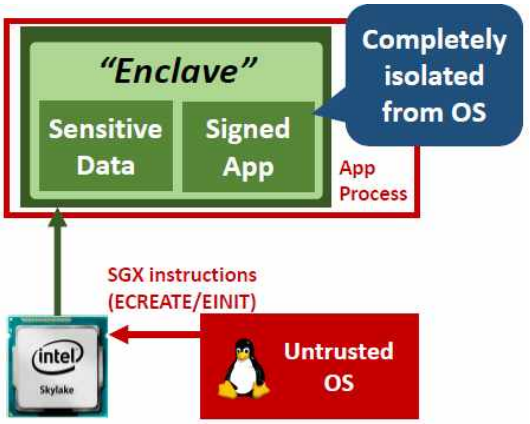


그림 1. Intel SGX 모식도[3]

Intel SGX는 Intel의 6세대 프로세서인 '스카이레이크'에서 처음으로 등장하였다. SGX는 CPU에 존재하는 하드웨어 특징으로서 소프트웨어적으로 구축된 보안 기법보다 강력한 보호 환경을 제공한다. SGX는 'enclave'라고 불리는 곳 안에서 프로그램을 실행할 수 있도록 한다. 각각의 enclave는 자신만 접근할 수 있는 메모리 영역이 존재하며[4] 민감한 연산을 이 안에서 수행함으로써 격리된(isolated) 실행 환경을 제공한다. 또한 각 enclave는 자신만의 가상 주소 공간을 가지고 있다.[1] enclave 안에 존재하는 코드와 데이터들은 CPU를 떠나 다른 곳으로 이동할 때 암호화를 유지한 채로 이동한다. 메모리에 있는 정보를 캐시(cache)로 읽어 들일 때 CPU가 복호화를 수행한 후 캐시 라인의 무결성(integrity)를 확인한 다음 virtual to physical mapping을 수행한다.[2] 이처럼 SGX는 enclave에서 밖으로 나가는 코드나 데이터에 대해서도 암호화를 유지하여 밖에서 봤을 때 이 안에서 실행되는 정보를 알 수 없도록 보안을 유지한다.

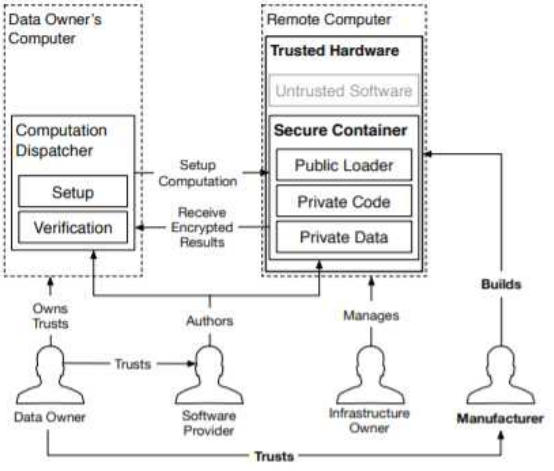


그림 3. Trusted Environment[3]

이는 피해자와 동일한 환경에서 프로세스를 실행시켜서 공격을 성공시키려는 공격자의 입장에서도 정보를 탈취하기 어려운 환경을 제공한다.

III. SGX를 활용한 어플리케이션 구현

SGX는 CPU에서 추가적으로 제공되는 x86 Instruction Set Architecture(ISA)이다.

Supervisor Instruction	Description	User Instruction	Description
ENCLS[EADD]	Add a page	ENCLU[EENTER]	Enter an Enclave
ENCLS[EBLOCK]	Block an EPC page	ENCLU[EEXIT]	Exit an Enclave
ENCLS[ECREATE]	Create an enclave	ENCLU[EGETKEY]	Create a cryptographic key
ENCLS[EDBGRD]	Read data by debugger	ENCLU[EREPORT]	Create a cryptographic report
ENCLS[EDBGWR]	Write data by debugger	ENCLU[ERESUME]	Re-enter an Enclave
ENCLS[EEXTEND]	Extend EPC page measurement		
ENCLS[EINIT]	Initialize an enclave		
ENCLS[ELDB]	Load an EPC page as blocked		
ENCLS[ELDU]	Load an EPC page as unblocked		
ENCLS[EPA]	Add version array		

그림 2. Intel SGX instructions[5]

따라서 해당 명령어들을 이용하여 SGX를 상에서 코드 수행을 할 수 있다. 라이브러리를 활용하면 이를 더 쉽고 강력하게 구현할 수 있다. 일례로 Graphene-SGX[2]에서는 TCB size를 고려했으며 SGX 상에서 unmodified application을 실행하는 것을 가능하게 하는 라이브러리를 제시한다. 이 외에도 다양한 라이브러리가 있으며 이에 대한 연구가 지속적으로 진행 중이다.

IV. SGX에 대한 공격 연구

앞서 살펴보았듯이 SGX는 앞으로 더욱더 널리 쓰이게 될 클라우드 환경에서 사용하기에 강력한 보안 솔루션 중 하나이다. 따라서 이에 대한 지속적인 공격이 연구되었다. 이 장에서는 기존의 캐시 부채널 공격(cache side-channel attack)을 알아보고 SGX에 대한 보다 발전된 공격에 대하여 알아본다.

3.1. 캐시 부채널 공격

부채널 공격은 공격자가 피해자(victim)의 시스템 권한을 탈취할 수 없으며 코드나 데이터에 대한 정보를 간단하게 알 수 없을 때 사용되는 공격 방법이다. 부채널 공격이라는 이름에서도 볼 수 있듯, 정석적인 방법이 통하지 않기 때문에 다른 활로를 개척하여 공격하는 방법이다. 때문에 기존의 공격보다는 많은 노력이 들고 실패할 가능성도 존재한다. 하지만 보안 위협 중 하나라는 사실에는 변함이 없으며 실제로 구현되어 비밀번호 등의 정보를 얻을 수 있음이 밝혀졌다.

캐시 부채널 공격은 부채널 공격 중 캐시의 이용하는 공격이다. 현대 컴퓨터 아키텍처는 메

모리에서 데이터를 가져올 때 속도를 더 빠르게 하기위해 캐시(cache)가 존재한다. 캐시는 한 번 메모리에서 가져온 값을 가지고 있다가 다시 그 값을 필요로 하게 되면 메모리까지 가지 않아도 해당 값을 제공하는 역할을 한다. 명령어(instruction)에서 특정 메모리 주소에 있는 값이 필요해졌을 때 먼저 캐시를 확인하는데, 캐시에 해당 값이 존재하면 캐시 hit이라 하고, 해당 값이 존재하지 않으면 캐시 miss라 한다. 캐시 miss가 발생했을 때는 다시 메모리에서 값을 가져오는 작업을 수행한다. 공격자는 hit과 miss의 차이를 이용하여 공격한다. 컴퓨터의 특정 캐시 라인 위치에다 특정 값을 쓰고 관찰한다. 데이터를 가져오는 시간이 빠르면 hit이 발생한 것이고, 데이터를 가져오는 시간이 느리면 miss가 발생한 것이다. 이 차이를 이용하여 지금 캐시에 어떤 데이터가 존재하는지를 확인할 수 있고 이를 충분히 많은 횟수동안 반복하여 원하는 만큼 데이터를 복구할 수 있다.

3.2. SGX에 대한 캐시 부채널 공격

앞서 서술한대로 SGX는 enclave를 생성할 수 있으며 각 enclave는 고유한 메모리 영역이 존재하여 특정 enclave만 접근할 수 있도록 되어있다. 또한 VM exit과 같은 일이 발생하지 않도록 하기 위해 몇몇 명령어들의 실행이 제한되어 있다.[2] 캐시 부채널 공격을 성공시키기 위해서는 정밀한 '타이머'가 필요하다. 캐시 hit과 miss에 대한 시간을 측정하여 그 차이를 이용하여 공격을 성공시키기 때문이다. 중요한 것은 절대적인 시간이 아니라 차이이기 때문에 Cycle Per Instruction(CPI)를 최대한 낮춘, 변수를 increment하는 방법을 통해 타이머를 구현하여 공격에 활용한다.[2]

V. 추후의 SGX 보안 강화 방안

보안성을 높이면 한편으로는 성능이 저하되는 overhead가 발생하는 것은 다소 불가피한 일이다. 앞서 살펴본 타이머를 지연시키기 위해 increment하는 명령어의 CPI를 의도적으로 늘린다는 등의 방법으로 정밀한 타이머의 구현을 막을 수 있을 것이다. 임의로 CPI를 증가시키는 만큼 성능의 저하가 약간 발생할 것으로 예상된다. SGX 안에서 활용이 불가능한 방법을 직접 구현하여 공격에 활용하는 것을 볼 수 있다. 이 방법들의 구현을 막는 쪽으로 추가적인 구현을 한다면 SGX의 보안성을 한 단계 높일 수 있을 것이다.

VI. 결론

공격자에 의한 보안 위협은 어느 환경에서든지 계속되어 왔으며 이에 따른 신뢰할만한 컴퓨팅 환경 또한 진화하고 있다. Intel에서 제시한 SGX는 CPU에 존재하는 하드웨어 feature로서 보안성을 높이는 데에 이바지했다고 할 수 있다. 하지만 이에 따른 보안 위협도 당연히 존재하며 앞으로 이는 더 개선되어 더 큰 위협이 될 것이다. 소프트웨어뿐만 아니라 하드웨어 기능에 대한 깊은 이해와 연구는 보안 분야를 연구함에 있어서 더 넓은 시야를 제공해주며 분야를 한 단계 더 발전시키는 데에 기여할 것이라고 확신한다.

VII. ACKNOWLEDGEMENT

이 논문은 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었으며 (IITP-2017-2015-0-00403), 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원과 (NRF-2017R1A2A1A17069478), 반도체공동연구소 지원의 결과물임을 밝힙니다.

[참고문헌]

[1] Schwarz, Michael, et al. "Malware guard extension: Using SGX to conceal cache attacks." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2017.

[2] Tsai, Chia-Che, Donald E. Porter, and Mona Vij. "Graphene-SGX: A practical library OS for unmodified applications on SGX." 2017 USENIX Annual Technical

Conference (USENIX ATC). 2017.

[3] Costan, Victor, and Srinivas Devadas. "Intel SGX Explained." IACR Cryptology ePrint Archive 2016.086 (2016): 1-118.

[4] Van Bulck, Jo, et al. "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution." Proceedings of the 27th USENIX Security Symposium. USENIX Association. 2018.

[5] JP Aumasson, Luis Merino. "SGX Secure Enclaves in Practive Security and Crypto Review." BlackHat 2016 https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf

마이크로아키텍처 부채널 공격기술 연구동향

김태현, 신영주
taehyun9203@gmail.com, yjshin@kw.ac.kr
광운대학교 컴퓨터공학과

Surveys on Microarchitectural Side Channel Attacks

Tae-Hyun Kim, Youngjoo Shin
Dept. of Computer Engineering, Kwangwoon University

요 약

마이크로아키텍처 부채널 공격은 CPU 캐시나 분기 예측기 등 마이크로아키텍처 요소에서 발생하는 시간차나 프로세서 동작에 의한 부작용을 부채널로 이용하는 공격을 말한다. 이중에서 캐시 부채널 공격은 목표 캐시의 라인 또는 캐시 집합의 이용을 관찰하여 암호체계에서 암호키를 추출하거나 은닉 채널에 사용되는 기법이다. 최근에는 CPU 캐시 뿐만 아니라 프로세서의 비순차 실행과 분기 예측기의 설계적 취약점에 의한 부채널을 활용하여 커널 주소공간을 접근하거나 다른 프로세스의 비밀정보를 획득할 수 있는 새로운 유형의 마이크로아키텍처 공격이 발견되었다. 본 논문에서는 여러 마이크로아키텍처 부채널 공격들에 대한 최신 연구 동향에 대해 살펴본다.

I. 서론

마이크로아키텍처 부채널 공격은 CPU 캐시나 분기 예측기 등 프로세서 공유자원에서 발생하는 시간차이를 부채널로 활용하여 공격대상자의 암호키와 같은 비밀정보를 획득하는 공격을 말한다.

이중 캐시 부채널 공격은 프로세서와 메인 메모리 사이에서 데이터의 전달속도를 높이기 위해 사용되는 하드웨어인 캐시를 부채널로 활용한 공격이다. AES, RSA 및 ECDSA 등 주요 암호 알고리즘들이 Flush+Reload와 Prime+Probe 와 같은 캐시 부채널 공격에 취약하다는 연구 결과가 이미 알려져 있다 [1,2].

최근에는 프로세서의 분기 예측과 비순차실행(out-of-order-execution)의 취약점을 이용한 Meltdown[3]과 Spectre[5] 등의 마이크로아키텍처 부채널 공격들이 발견되었다. 본 논문에서는 캐시 부채널을 비롯한 다양한 마이크로아키텍처 부채널 공격에 대한 최신 연구 동향에 대해 살펴보고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 캐시 부채널 공격인 Flush+Reload와 Prime+Probe 공격을 소개하며 3장에서는 비순차실행과 분기 예측 등 예측 실행의 취약점을 이용한 Meltdown과 Spectre 공격에 대해 소개를 한다. 마지막으로 4장에서는 결론 및 미래의 연구동향에 대해 서술한다.

II. 캐시 부채널 공격

2.1 Flush+Reload

Flush+Reload[1] 공격은 캐시 부채널 공격 중 하나로써 이 공격을 하기 위해서는 공격자와 희생자가 공유하는 메모리 라인이 필요하다. 공격의 대상은 L3 캐시에 있는 캐시 라인으로 지정한다. 그 이유는 대부분의 프로세서에서 L1와 L2 캐시는 각 코어에 존재하지만, L3 캐시 같은 경우는 서로 다른 코어에서

공유하므로, 공격자와 희생자가 서로 다른 코어에 있는 환경에서도 공격할 수 있다. 이것은 희생자가 공격자가 공격했는지를 알아차릴 수 없으므로 공격을 하는 데 있어 은신 특성이 있다. Flush+Reload 공격은 크게 세 단계로 나누어지게 된다.

첫 번째 단계는 공유된 메모리의 내용을 캐시 한 상태에서 시작한다. 이 상태에서 공격자는 공유된 메인 메모리의 내용이 존재하는 L3 캐시 라인을 x86 프로세서 명령어인 clflush 명령어를 이용하여 제거한다. 제거한 캐시 라인은 메인 메모리로 이동하게 되며, 인텔 프로세서의 캐시 일관성에 의해 L3 캐시 내에 있는 캐시 라인을 제거하면 L3 캐시 상위 레벨 캐시인 L1, L2 캐시도 L3에서 제거된 캐시 라인과 같은 내용은 제거되므로 상위 레벨 캐시에서 사라지게 된다.

두 번째 단계는 첫 번째 단계가 끝난 후 희생자가 공유된 메모리 라인을 실행한 뒤, 공격자는 희생자가 공유된 메모리 라인을 실행하는 시간 동안 아무것도 하지 않고 대기를 하고 있다.

세 번째 단계는 두 번째 단계가 끝난 후 공격자는 공유된 메모리 라인을 L3 캐시로 다시 로드 한 뒤, 특정 함수를 사용하여 L3 안에 있는 목표 캐시 라인에 대한 접근시간을 측정한다. 측정한 시간이 짧다면 희생자가 해당 캐시 라인을 사용했기 때문에, 공격자가 로드 할 때 해당 캐시 라인이 L3 캐시 안에 존재하였다는 것을 의미한다. 반대로 걸리는 시간이 길다면 희생자가 해당 캐시 라인을 쓰지 않았다는 것을 의미한다. 그래서 공격자가 로드 할 때, L3 캐시 안에 있는 캐시 라인이 존재하지 않으므로, 결국 메모리에 접근해서 해당하는 메모리 라인을 캐시로 가져와야 하므로 상당한 시간이 걸린다. 이 공격은 희생자가 해당 캐시 라인에 해당하는 내용을 공격자가 사용 유/무를 파악할 수 있다.

2.2 Prime+Probe

Prime+Probe[2] 공격은 공유된 메모리 라인이 없는 환경에서 사용하는 공격이다. 이 공격은 공유된 메모리 라인이 없으므로 특정 캐시 라인을 공격하지는 못하고, 캐시 셋을 이용해서 공격한다. 공격자는 희생자의 L3 캐시 내에서 공유된 캐시 셋을 만든 뒤, 세 단계로 이루어진 Prime+Probe 공격을 한다.

첫 번째 단계는 Prime 단계이다. 공격자는 공유된 캐시 셋에 자신의 데이터를 채운다. 두 번째 단계에서는 공격자는 기다리고, 희생자가 공유된 캐시 셋에 자신의 데이터로 채운다. 세 번째 단계는 Probe 단계이다. 공격자는 공유된 캐시 셋에서 자신의 데이터를 로드하면서, 캐시 셋 안에 있는 각 캐시 라인에 대한 접근 시간을 관측한다. 희생자가 사용한 캐시 라인이 있다면 해당 캐시 라인은 메모리에서 가져와야 하므로 캐시 라인에 접근 시간이 오래 걸린다. 반대인 경우에는 해당 캐시 라인은 캐시에 머무르므로 접근 시간이 짧다.

III. 예측 실행 공격

3.1 Meltdown

비순차실행은 가능한 CPU 코어가 모든 연산 유닛(Execution unit)을 사용하도록 하는 최적화 기술이다. 현재 명령어에 대한 연산 유닛을 사용하는 동안 다른 명령어들이 미리 연산 유닛을 이용하여 처리함으로써 마치 병렬적으로 명령어를 처리하는 효과를 보여준다. 그러나 비순차실행 도중 예외가 발생한다면 이미 실행된 명령어들의 실행을 취소하고 그 결과값은 제거되어야 한다. 이렇게 실행이 취소되는 명령어들을 transient instructions 이라 한다.

인텔 등 대부분의 프로세서는 실행이 취소된 명령어의 결과값들은 레지스터나 메모리와 같은 아키텍처 관점에서는 사라지지만 캐시 등 마이크로아키텍처 관점에서는 흔적이 여전히 남아 있다는 취약점을 가지고 있다. Meltdown[3]은 이러한 취약점을 이용하여 공격자가 접근할 수 없는 주소인 커널 주소에 접근하여 예외 상황에서 비순차실행 때문에 실행된 transient instruction에 커널 주소의 값을 저장

해서 커널 주소의 값을 추출하는 공격이다 (그림1 및 그림2 참고).

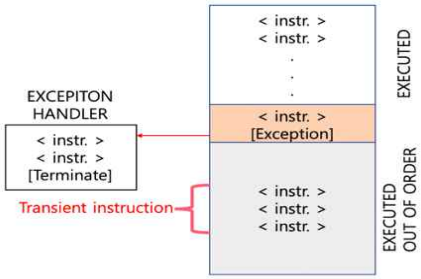


그림 1 Meltdown 공격 수행 코드

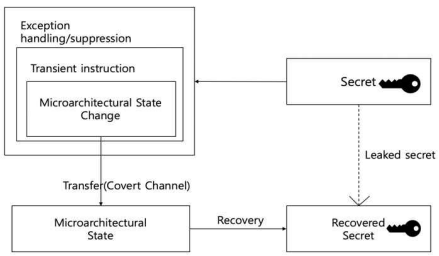


그림 2 Meltdown 공격 과정

```
1; rcx = kernel address
2; rbx = probe array
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

그림 3 비 순차 실행의 부작용을 이용한 코드

그림 3은 Meltdown 공격을 통해 커널 주소 값을 추출하는 코드이다. 각 코드 라인에 대한 설명은 다음과 같다. (1),(2) 커널 주소와 와 캐시 라인을 관찰하는 함수를 레지스터에 저장한다. (4) 커널의 주소에 접근하는 과정이며 여기에서 예외가 발생한다. (5)~(7) 예외가 발생하여 transient instruction 실행 결과는 버려지지만 캐시에 그 값은 남아 있게 된다. (7) Flush+Reload 공격에 사용되는 probe 함수에 커널 주소 값을 넣는다. 캐시 라인 안에 커널 주소 값이 들어 있다면 probe 함수가 반환하는 사이클은 짧을 것이며, 반대로 캐시 라인에 존재 않는다면 긴 사이클이 반환할 것이다. (3) ~ (7)까지의 과정을 반복하면 결국 전체의 커널 주소를 정확히 알 수 있게 되며, 결국 물리 메모리 전체를 얻게 된다.

3.2 Spectre

Spectre[4] 공격은 Meltdown 공격과 유사하게 transient instruction을 이용하여 비밀 정보를 획득한다. 그러나 Spectre 공격은 예외 상황을 이용하는 Meltdown 과 달리 분기 예측기를 사용하여 transient instruction을 실행하는 것이 특징이다.

분기 예측기는 투기적 실행(Speculative execution)을 이용하여 분기할 주소를 예측해서 분기 명령어가 분기할 주소를 다음 사이클에 바로 갈 수 있도록 도와준다. 비순차실행을 이용하여 예측 실행을 하게 된다면 크게 두 가지의 상황으로 나뉘게 된다. 만약 예측이 성공한 경우 다음 사이클에 해당 주소로 분기한다. 그런데 예측이 실패한 경우에는 명령어의 실행을 취소하고 그 결과 값들을 아키텍처에서 제거한다. 그러나 이 경우 캐시 메모리에는 여전히 값이 남게 되는 취약점을 가지고 있다.

Spectre는 비순차실행과 투기적 실행의 취약점을 이용해서 transient instruction에 분기 명령어를 넣은 뒤 투기적 실행을 통해 비밀 정보의 값을 추출한다.

```
1 if (x < bitstream_length)
2.   if (bitstream[x])
3.       flag = true
```

그림 4 Spectre 공격 실행 코드

그림 4는 Spectre에서 비밀 값을 유출하는 코드이다. x는 공격자가 제어하는 값이며, x = (비밀 키의 주소 값) - (bitstream의 기본 주소 값)이다. Spectre는 x의 값 이전에 bound의 조건을 충족하는(Taken) 값만 보내 Taken 값만 보낸다고 착각하게 만든 뒤, 공격자는 x에 경계 밖의 값(Not Taken)을 보내서 분기 예측이 명령어를 미리 가져와서 비 순차 실행으로 처리한다(2)~(3). 경계 밖의 값을 수행했으므로 프로세서는 수행된 값을 아키텍처 관점에서는 버려서 보이지는 않지만 마이크로아키텍처 관점에서는 값이 보이는 현상이다. 두 번째 줄의 실행으로 인해 캐시에는 세 번째 줄에 있는 flag의 값이 1(true)로 저장이 되어 있으므로 Flush+Reload 공격을 이용한

캐시 은닉 채널을 형성한 후 공격자에게 flag(비밀 키 주소 값) 값을 유출할 수 있다.

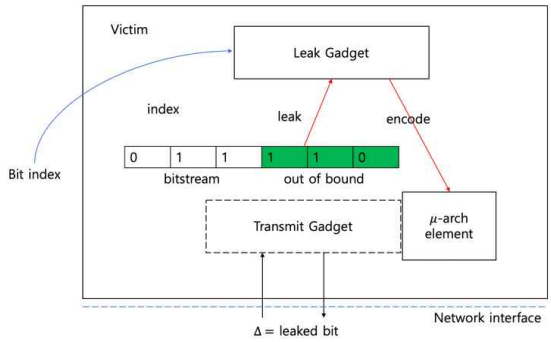


그림 5 Netspectre 공격 과정

한편 최근에는 네트워크를 통해 원격에서 Spectre 공격을 수행하는 Netspectre 공격이 공개되었다 [4]. 이 공격은 Spectre 공격 코드를 변형하여 네트워크 상에서 1비트씩 패킷 값을 유출시킨다 (그림 5 참고). 공격자는 유출된 패킷에 대한 응답시간이 마이크로초로 짧으므로 사이클 차이가 작은 캐시 은닉 채널에서는 수많은 측정을 통해 패킷의 값을 구별할 수 있다.

```
1 if (x < bitstream_length)
2.   if (bitstream[x])
3.       _mm256_instruction ();
```

그림 6 Netspectre 공격 실행 코드

캐시 은닉 채널보다 비밀 패킷 값을 측정하는 수를 줄이기 위해서 그림 6에서 256bit 명령어를 처리하는 AVX2 유닛을 이용하여 사이클 차이가 큰 은닉 채널을 사용한다. 측정 후 다음 패킷의 비트 값을 받고 싶다면 기존에 캐시 된 패킷의 비트 값을 없애야 하는데 공격자는 희생자 캐시에 대해 제어를 할 수 없으므로 공격자는 희생자로부터 파일을 다운로드를 이용한다. 다운로드를 통해 기존에 캐시 안에 있었던 패킷 값은 제거되므로 결국 캐시는 패킷의 값이 없는 상태로 리셋이 되고 처음부터 다시 공격을 시작하면 패킷 전체의 값을 네트워크를 통해 받을 수 있다.

IV. 결론

본 논문에서는 마이크로아키텍처 상에서의 부채널을 이용한 공격에 대해 조사하였다. 첫 번째는 CPU 캐시의 타이밍을 이용한 공격으로 Flush+Reload 공격과 Prime+Probe 공격으로 각각 캐시의 라인과 캐시 셋을 이용한 공격이다. 두 번째는 프로세서의 최적화 기술 중 하나인 비순차실행에서의 취약점을 이용한 공격으로 Spectre 과 Meltdown 두 가지에 대해 알아보았다.

마이크로아키텍처를 이용한 부채널 공격은 기존에는 캐시 타이밍을 이용한 공격에서 프로세서의 예측실행의 취약점을 이용한 공격 기술로 전환되었다. 앞으로는 프로세서의 새로운 취약점을 악용한 공격이 더욱더 많이 발견될 것으로 예상된다. 그러므로 프로세서 내부 구조에 관한 연구를 통해 공격에 대한 대비책을 세울 필요가 있다.

Acknowledgement

본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

[참고문헌]

[1] Y. Yarom and K. Falkner, "Flush + Reload : a High Resolution , Low Noise , L3 Cache Side-Channel Attack," in USENIX Security, 2014.

[2] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in IEEE Symposium on Security and Privacy, 2015.

[3] M. Lipp et al., "Meltdown: Reading Kernel Memory from User Space," in USENIX Security 2018.

[4] M. Schwarz, M. Lipp, and D. Gruss, "NetSpectre: Read Arbitrary Memory over Network," ePrint Archive, 2018.

[5] P. Kocher et al., "Spectre Attacks: Exploiting Speculative Execution," in IEEE Symposium on Security and Privacy, 2019.

국산 암호 알고리즘 부채널 분석에 대한 고찰

안규황*, 권용빈*, 권혁동*, 서화정*†

*한성대학교 대학원 정보시스템공학과

A Study on Side Channel Analysis of Domestic Cryptographic Algorithm

Kyuhwang An*, Yong-Been Kwon* Hyeokdong Kwon* Hwajeong Seo*†

*Division of information system engineering, Hansung University.

요 약

암호 알고리즘의 키(Key)를 찾아내는 방법으로 기존의 수학적 알고리즘을 분석하며 접근하는 방법과 다르게 암호 알고리즘이 하드웨어에서 동작할 때 생기는 전력 소모, 시간 등의 누수 정보들을 분석하는 방법으로 부채널 분석(Side-Channel Analysis)이 있다. 현재 AES 등 널리 이용되고 있는 국제 표준 알고리즘들이 부채널 분석으로부터의 취약점이 밝혀지고 있고 이와 동시에 대응 기법이 마련되는 등 연구가 활발하게 진행되고 있어, 국산 암호를 대상으로도 이러한 연구를 통해 취약점을 분석하고 보완할 필요하다. 본 논문에서는 부채널 분석 기법과 그 대응기법들을 소개한다.

I. 부채널 분석이란

부채널 분석이란 암호 알고리즘이 탑재된 전자기기를 작동시킬 경우 발생하는 전자파, 전력 소모량등의 누수정보들을 분석하여 비밀 키를 획득하는 분석법이다. 최초의 부채널 분석은 1996년에 P.Kocher에 의해 발표한 시차분석 공격[1]이며, 해당 논문을 기점으로 본격적인 부채널 관련 연구가 실행되었다.

부채널 분석이 발표되기 이전에는 다양한 암호 알고리즘의 비밀 키를 찾기 위해 다양한 평문을 넣는 방법인 전수조사, 비밀 키를 탈취하는 키 로깅 등이 시행되어 왔지만, 부채널 분석 발표 이후에는 암호 알고리즘이 탑재된 전자기기가 발생시키는 누수정보들을 분석하여 비밀 키를 획득할 수 있게 되었다. 따라서 이전과 달리 암호 알고리즘 또는 하드웨어 설계, 개발 시에 누수정보를 고려하여야 한다.

II. 부채널 분석 기법과 그 대응 기법

본장에서는 부채널 분석 중 연산의 전력소모량의 차이를 이용하여 분석하는 대표적인 기법들인 SPA(Simple Power Analysis), DPA(Differential Power Analysis), CPA(Correlation Power Analysis) 그리고 Guessing Entropy를 소개하고 이에 대한 대응 기법들을 소개한다.

2.1 분석 기법

2.1.1 SPA(Simple Power Analysis)

암호 알고리즘의 내부 함수들은 각기 다르게 구성되어 있어 서로 다른 전력 소모를 보인다. 이에 따라 각 함수별로 패턴화되어 파장으로 나타낼 수 있으며 이를 기반으로 어떤 함수들이 실행되는지를 알 수 있어 중간 값 등 키와 관계된 정보의 추출이 가능하다.

2.1.2 DPA(Differential Power Analysis)

평문을 넣어 암호를 구동하여 파형을 수집하고 수집된 파형에서 분류함수를 통해 정확한 비밀 키와 반응이 일어나는 파형을 얻는다. 두 개로 분류된 데이터를 각각 평균하여 차분을 구한다. 추측된 비밀 키 값이 올바를 경우 반응지점에서의 차이가 크게 발생해 차분 값이 큰 값을 가지고 맞지 않다면 0에 가까운 작은 값을 가진다. 이를 추측된 모든 비밀 키에 반복

하여 올바른 비밀 키 값을 찾을 수 있다.

2.1.3 CPA(Correlation Power Analysis)

평문 또는 비밀 키 값과 평문과 비밀 키를 연산(평문을 P, 비밀 키를 C라고 할 경우 P + C, P ⊕ C 등)하여 나오는 값을 선택한다. 공격자가 원하는 N번만큼의 평문 데이터를 암호 알고리즘에 입력시킨 후 소비되는 전력을 측정한다. 이때 소비되는 전력량을 I라고 할 때 해당 파형의 길이는 N * I가 된다. 예상 가능한 중간 값을 모든 키에 대입하여 계산한다. 계산 된 중간 값에 상응하는 전력 소비량을 계산하여 실제 전력 소비 값과 비교한다. 여기서 만약 임의로 측정한 값이 실제 전력 소비 값과 높은 상관관계를 갖고 있다면, 실제 비밀 키를 찾았다고 볼 수 있다.

2.1.4 Guessing Entropy

Guessing Entropy는 무작위로 추출한 파형 중 실제 비밀 키 값을 도출하기 위하여 공격자가 임의로 정의한 횟수만큼 돌려 실제 키와 상관관계가 높은 순서대로 나타내는 방법을 의미한다. 이때 추출하는 파형이 아무리 많다고 할지라도 상관관계가 가장 높은 비밀 키를 실제 비밀 키라 칭할 수 있다.

2.2 대응 기법

이에 대한 전력 분석의 대응 기법인 무작위성 기법, 마스킹 기법, 블라인딩 기법을 소개한다.

2.2.1 무작위성(Randomization) 기법

무작위성 기법을 적용하지 않은 암호 알고리즘의 경우 소비 전력 및 실행 시간 등 다양한 방법으로 부가적인 정보를 획득할 수 있게 되지만, 무작위성 기법을 적용한 암호 알고리즘의 경우 암호 알고리즘을 실행시키는데 발생하는 모든 값에 대해 난수처리 한다. 이렇게 되면 공격자는 최종 출력 값으로 난수를 획득하게 되고, 해당 난수에 대한 중간 값을 유추할 수 없기 때문에 SPA에 대하여 대응할 수 있다.

2.2.2 마스킹(Masking) 기법

마스킹 기법이란 암호 알고리즘이 연산을 수행할 때 발생하는 중간 값을 공격자가 유추할 수 없게끔 발생하는 전력 소모량을 조작하여 실제 비밀 키와의 상관관계를 제거하는 방법이다. 기본적인 마스킹 연산은 수식 1과 같다.

$$\text{불 연산} \ : x' = x \oplus r \quad (1)$$

$$\text{산술 연산} \ : A = x - r \bmod 2^k$$

2.2.3 블라인딩(Blinding) 기법

블라인딩 기법은 Chaum이 최초로 제안하였다. 블라인딩 기법은 암호 알고리즘에 적용되어 있는 함수를 이용하여 평문을 입력하여 비밀 키가 도출될 때 평문과 비밀 키를 숨기면서 값을 계산해주는 암호 함수를 설계해주는 기법이다.

III. 관련 연구 동향

3.1 SEED[2]

SEED에 대한 부채널 공격을 실험하기 위해 해당 논문에서는 XMEGA 보드를 사용했다. 해당 논문에서는 더미 연산 & 셔플링 및 1차 마스킹이 적용된 SEED에 대한 SOCPA 공격법에 대하여 설명한다. 이를 설명하기에 앞서 해당 논문에서 제안한 방법들에 대한 평가들을 위하여 몇 가지를 정의한다. 첫 번째로 부채널 공격 수행을 위한 비밀 키 추측을 위한 최소 경우의 수이다. 예를 들어 상관 전력 분석으로 S-Box를 추측하기 위해서는 2^8만큼의 공격 복잡도가 생긴다. 비밀 키 추측을 위해 최소 경우의 수만큼 수행해야 되기 때문이다. 두 번째로 셔플링이 적용되지 않은 비밀 키가 추출되는 최소 파형의 수를 ∂고 할 때, 특정 연산에서 셔플링이 $\frac{1}{n}$만큼 적용된다면, 필요한 최소 파형의 수는 $\partial \times \frac{n(n-1)}{2}$이다. SOCPA에 대한 공격지점은 더미 연산 및 셔플링 기법이 적용된 S-Box 출력과 G-함수 출력을 의미한다.

제안하는 공격 기법을 사용할 경우 위에서 언급한 2^8 만큼의 공격 복잡도가 아닌 2^{16}이 된다. 그 이유는 8비트 중 2비트 만 상관이 있기 때문이다. Noise-free 환경에서도 입력 가능한 값에 대한 상관도에 대하여 전수조사 하였을 때, 8비트의 값과 2비트의 상관도는 0.25이다.

따라서 필요파형 수는 상관도의 역수의 제곱에 비례하여 $\left(\frac{1}{0.25}\right)^2$배의 파형 수가 필요하다.

제안한 공격 방법을 수행하기 위해 G-함수 출력 값을 해밍웨이트 값으로 생성하였다. 현재 SEED에 적용되어 있는 셔플링 대응기법을 우회하기 위해 G-함수 출력 값을 로드하는 부분인 G-함수 덧셈 부분에서 파형을 수집하였으며, 기존 공격 방법들에 대한 공격을 수행하기 위해 G-함수 연산 부분을 따로 수집하였다. 그 결과 해당 논문에서 제안한 방법으로 약 8,000개 파형으로 정확히 Guessing Entropy가 1로 수렴함을 알 수 있었으며, 실제 장비에서 100,000개 파형에 대해 일반적인 방법으로 옳은 비밀 키를 추출하는데 실패했지만 해당 논문의 공격 방법에 의해 8,000개의 파형으로 옳은 비밀 키를 추출할 수 있음을 증명하였다.

3.2 HIGHT[3]

3.2.1 평문과 암호문을 이용한 HIGHT 부채널 분석

HIGHT 암호 알고리즘에 존재하는 취약점을 검증하고 이를 증명하기 위해 해당 논문에서는 ATmega128 보드를 사용했다. 해당 논문에서는 2개의 선형 연산을 분석하여 비밀 키를 복구하는 방법으로 취약점을 보여준다. 그림 1를 보면 빨간색 네모와 파란색 동그라미가 있다. 빨간색 네모의 경우 비선형 연산을 분석하는 지점이며 파란색 동그라미는 선형 연산을 분석하는 지점이다. 선형 연산보다 비선형 연산이 부채널 분석에 더 용이하게 사용되지만, WK_1, WK_3은 선형 연산으로 이루어져 있기 때문에 WK_1, WK_3를 구하기 위해선 선형 연산도 수행을 해줘야한다.

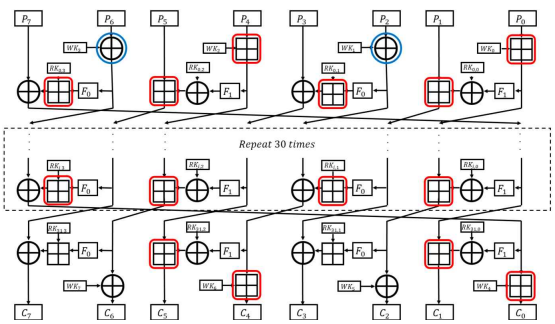


Fig. 1 Attack point of CPA

평문과 암호문 두 가지 모두 사용하는 부채널 분석의 경우 WK_1, WK_3가 k_{13}, k_{15}를 복구하는 것과 같기 때문에 비선형 연산만으로 원하는 암호문을 획득할 수 있다. 평문과 암호문을 모두 사용할 수 있는 환경에서는 라운드 함수 중 마지막 31번째 단계를 이용하는 것이 적절하다.
평문만 사용할 경우엔 선형 연산을 반드시 수행해줘야지만 원하는 암호문을 획득할 수 있다. 평문만 사용할 수 있는 환경이라면 라운드 함수 중 마지막 31번째 단계를 제외한 부분을 사용하는 것이 적절하다.

3.2.2 HIGHT 부채널 분석 실험

해당 논문에서는 ATmega128 보드에 8bits로 구현된 HIGHT 알고리즘을 사용하였으며, 공격 성공률 과 guessing entropy 각각 1,000번씩 분석하였다.
그림 2, 3[1]를 보면 각각 1,000번을 돌린 결과에 대해 확인할 수 있다. 그림 내부에 있는 F_0은 선형 함수만을 이용한 분석이며, F_1은 비선형 함수만을 이용한 분석이다.

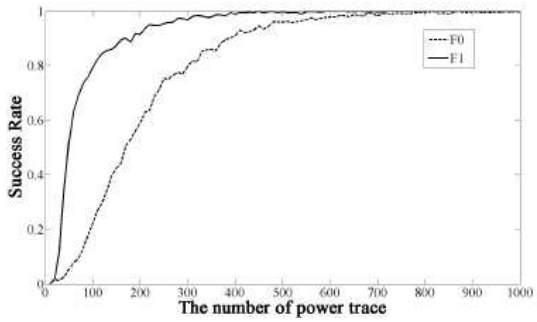


Fig. 2 Success rate of CPA for HIGHT

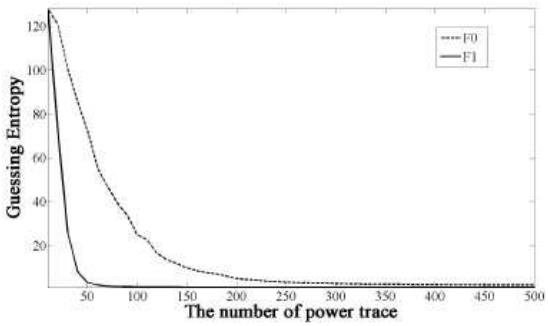


Fig. 3 Guessing entropy of CPA for HIGHT

1) 해당 그림은 원 논문에서 참조하였다.

그림 2는 공격 성공률을 나타내는 그림으로 공격 성공률이 99%이상이 되기 위해서 F_0는 730번 이상 F_1은 400번 이상인 것을 알 수 있다.

그림 3는 guessing entropy로 복잡도가 3이하가 되기 위해서 F_0는 270번 이상 F_1은 50번 이상인 것을 알 수 있다. 따라서 선형과 비선형 함수 모두 부채널 분석에 가능함을 증명하였으며 앞에서도 언급했듯이 부채널 분석에는 비선형 함수가 조금 더 용이한 것을 실험을 통해 다시 한 번 알 수 있었다.

3.3 LEA

2014년 CPA에 의해 LEA의 서브 키 두 쌍이 밝혀지면서[4], LEA 역시 전력 부채널 분석에 안전하지 않음이 알려졌다. 따라서 적어도 이러한 1차 분석에 대한 마스킹 기법을 적용하는 것이 필요하며, 이 과정에 있어 LEA의 설계 목적인 경량화, 고속화를 고려하는 것이 필요하다.

3.3.1 LEA에 대한 마스킹

LEA의 구조에서도 두 부분(XOR, Addition)에서 중간 값이 생성된다. 이 값을 통계적으로 분석하여 비밀 키를 탈취할 수 있다. 따라서 이 지점의 값에 난수를 취하여 비밀 키와의 상관관계를 제거하는 방식으로 마스킹 기법이 적용될 수 있다. 이러한 마스킹 기법의 적용에는 몇 가지 진행 중인 연구가 있다. 먼저, 더해지는 난수 자체의 안전성을 제기한 논문에서는 안전성을 검증한 TRNG(True Random Number Genorator)기반의 마스킹 기법을 제안했다.

다음으로 LEA가 가지는 ARX구조의 특징아래 두 종류의 연산 불 연산과 산술 연산간의 변환 기법이 필요하다. 이 변환 기법을 다른 논문에서는 이론상에서 효율적이라고 여겨졌던 룩업테이블 방식의 변환 기법이 실제 LEA가 목적으로 하는 플랫폼 위에서 비효율적임을 밝히고 마스킹 기법이 적용된 두 숫자를 기법을 해제하지 않고 연산하는 Secure Addition 기법을 적용한 실용적인 방법을 제안하고 구현한다. 이를 통하여 기존 AES 대비 마스킹 기법 적용에 의한 연산량 증가의 비효율성 문제를 완화했다.

IV. 결론

본 논문에서는 국산 암호 알고리즘에 적용된 부채널 분석에 대해 기존에 연구 된 사례들에 대하여 조사하였다. 그 결과로 국산 암호의 각 구조에 맞추어 부채널 분석이 시행될 수 있으며 부채널 분석 대응 기법 또한 국산 암호의 구조적 특징과 목적을 고려하여 효율적으로 적용되어야 함을 알 수 있다. 이후에는 또 다른 국산 암호 LSH, ARIA와 아직 표준화 작업이 진행되진 않았지만, 가장 최근에 나온 국산 블록암호인 CHAM에 대하여 본 논문에서 살펴본 바를 참고하여 부채널 분석을 해보고 그 대응 기법을 적용해 볼 예정이다.

[참고문헌]

[1] P. C. Kocher, "Timing Attacks on Implementation of Diffe-Hellman, RSA, DSS and Other Systems," Available: https://www.paulkocher.com/doc/TimingAttacks.pdf

[2] Y. S. Won, A. S. Park, D. G. Han, "Side Channel Analysis with Low Complexity in the Diffusion Layer of Block Cipher Algorithm SEED," *Journal of the Korea Institute of Information Security & Cryptology* 27(5), pp. 993-1000, Oct. 2017.

[3] T. J. Kim, Y. S. Won, J. H. Park, H. J. An, D. G. Han, "Side Channel Attacks on HIGHT and Its Countermeasures," *Journal of the Korea Institute of Information Security & Cryptology* 25(2), pp. 457-465, Apr. 2015

[4] Y. D. Kim, and H. S. Yoon, "First Experimental Result of Power Analysis Attacks on a FPGA Implementation of LEA," *IACR Cryptology ePrint Archive 2014*: 999. 2014

오토인코더 기반 딥러닝을 활용한 부채널 분석 노이즈 제거 기술 연구*

권동근*, 진성현*, 김희석**, 홍석희*
*고려대학교 정보보호대학원, **고려대학교 사이버보안학과

Study of Deep Learning Based Trace Preprocessing Method with Denoising Auto-Encoder

Donggeun Kwon*, Sunghyun Jin*, HeeSeok Kim**, Seokhie Hong*
*Graduate school of information security, Korea University.
**Department of cyber security, Korea University.

요 약

성공적인 부채널 분석이 이루어지기 위해서는 장비나 암호 알고리즘에 알맞은 부채널 분석 기법을 통한 취약점 분석뿐만 아니라, 그 이전에 수행해야 하는 단계인 측정 장비를 이용한 부채널 정보 수집 및 적절한 파형 전처리 과정 또한 중요하다. 분석 환경에 따라서는 전처리 단계 수행 여부에 따라 부채널 분석 성능이 크게 달라지거나 실패할 수 있다. 본 논문에서는 오토인코더를 이용한 파형 전처리를 통해 노이즈 제거로 부채널 분석 성능을 향상할 수 있는 기법에 대해 제안하고, 실험을 통하여 전처리 여부에 따른 상관전력분석 성능을 비교한다.

I. 서론

AES(Advanced Encryption Standard)[4]와 같이 이론적으로 안전성이 검증된 암호 알고리즘이 실제 장비에서 구현되어 동작할 때, 기존 암호 분석에서 이용되는 정채널로부터 나오는 암호문과 평문 정보뿐만 아니라 장비가 소모하는 전력, 전자기파, 연산 시간 등의 여러 부채널에서 나오는 정보 또한 공격자에게 유출될 수 있다. 이러한 부채널에서 획득한 정보들을 이용하여 암호 시스템을 분석하는 것을 부채널 분석이라고 한다.

여러 부채널 중에서 장비의 소비 전력을 이용하는 부채널 분석 수행 과정은 크게 3가지 단계로 나눌 수 있다. 첫 번째로는 암호 연산을 수행하는 공격 장비로부터 전력 파형을 수집하는 측정 단계이다. 다음으로는 수집한 전력 파형에서 유의미한 정보를 분석할 수 있도록 노이즈 제거, 시점 정렬 등을 수행하는 전처리 단계, 마지막으로 전력 파형으로부터 부채널 분석 기법을 이용한 취약점 분석을 통해 비밀 정보를 복원하는 단계로 구성된다.

첫 단계에서는 전력을 수집할 대상인 공격 장비와 해당 장비로부터 소비 전력을 측정하기 위한 오실로스코프와 같은 측정 도구가 필요하다. 이 단계에서는 공격자의 능력보다는 측정 도구의 성능에 따라 다른 결과가 나타난다. 마지막 단계는 수집한 파형에 부채널 분석 기법을 적용하여 비밀 정보를 복원하는 단계이다. 알고리즘의 명세에 따라 파형을 분석하여 비밀 정보를 획득하는 단계로, 알고리즘의 조건에 일치하는 파형을 명세에 따라 올바르게 처리한다면 충분히 수행할 수 있다.

하지만 실제 부채널 분석 환경에서 획득한 전력 파형은 시점의 흐트러짐, 노이즈 등, 알고리즘에서는 고려되지 않은 부분들이 존재하기

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

때문에 분석 기법을 올바르게 수행하더라도 암호 시스템 분석에 실패할 수 있다. 따라서 공격자는 이를 알맞은 방식으로 전처리해야 하지만, 알고리즘의 가정에 맞도록 파형을 전처리하는 것은 공격자의 능력에 따라 결과가 달라진다는 문제점이 존재한다.

따라서 본 논문에서는 공격자의 능력을 최소로 하는 파형 전처리를 위한 딥러닝 알고리즘 중 오토인코더를 이용한 노이즈 제거 기법을 제안하고, AES 알고리즘의 전력 파형에 대한 실험을 통해 전처리 기법 사용 여부에 따른 상관전력분석 성능을 비교한다.

이어지는 본 논문의 2장에서는 상관전력분석과 오토인코더에 대해 소개하고, 3장에서 오토인코더를 이용한 노이즈 제거 기법을 제안한 후, 제안기법을 통한 실험결과를 4장에서 보인다. 마지막 5장에서는 결론과 향후 연구 방향을 제시하면서 마친다.

II. 배경지식

2.1 상관전력분석

공격 장비로부터 전력 파형을 수집 가능한 환경에서의 부채널 분석 기법들에는 대표적으로 차분전력분석(DPA; Differential Power Analysis)[7]과 상관전력분석(CPA; Correlation Power Analysis)[2]이 존재한다. 그중에서 2004년에 Brier 등이 제안하였던 상관전력분석은 암호 연산을 수행하는 장비가 비밀키와 연관된 값을 연산할 때 발생하는 소비 전력과 연산하는 값 사이에 상관관계가 있다는 점을 이용하여 통계적인 방법을 통해 비밀키를 분석하는 기법이다. 이러한 기법을 활용한다면 평문 또는 암호문을 알고 있는 공격자가 키 추측에 따른 중간값들을 계산하고, 매칭되는 소비 전력량과 가장 상관관계가 높은 중간값을 계산한 추측키를 비밀키로 추정할 수 있다.

2.2 오토인코더 (Auto-Encoder)

오토인코더란 딥러닝 알고리즘 중 대표적인 비지도 학습으로 딥러닝 네트워크의 출력을 입력과 유사하도록 가중치들을 학습하여 입력 데이터를 압축하거나, 딥러닝 네트워크의 사전 학습, 데이터의 노이즈 제거 등의 목적을 위해 사용되는 모델이다.

초기의 오토인코더는 주로 딥러닝 네트워크에서 사용되는 가중치에 대한 초기화 및 사전 학습(Pre-training)을 위하여 사용되었다. 딥러닝 네트워크의 각 레이어 가중치에 대해서 입력 데이터를 가장 잘 표현할 수 있도록 오토인코더를 이용해 사전 학습하여 가중치를 초기화한 이후, Fine-tuning을 통하여 전체적인 가중치를 조정하는 방식으로 네트워크를 학습하였다. 사전 학습이 각 레이어에 대한 가중치 학습이라면, Fine-tuning은 모든 레이어들에 대한 가중치 학습으로 전체 네트워크가 잘 동작할 수 있도록 학습한다. 이러한 가중치 초기화 방식을 Stacked Auto-Encoder(SAE)[5]라고 부른다. 하지만 현재는 SAE보다 성능이 개선된 여러 가중치 초기화 방법들이 제안되면서 네트워크의 가중치 초기화를 위하여 오토인코더를 이용하는 방법은 잘 사용되지 않게 되었다.

다음으로 오토인코더는 데이터의 차원을 압축을 위한 기계학습 모델로 사용된다. 이러한 목적으로 사용되는 오토인코더는 기본적으로 DNN(또는 MLP; MultiLayer Perceptron)의 구조와 유사하며, 입력 데이터의 차원을 압축하는 인코더(Encoder) 부분과 오토인코더를 통하여 압축된 데이터를 원본의 입력 데이터로 다시 재구성하는 디코더(Decoder) 부분으로 이루어져 있다. Fig. 1은 오토인코더의 기본적인 네트워크 구조를 나타낸 그림이다.

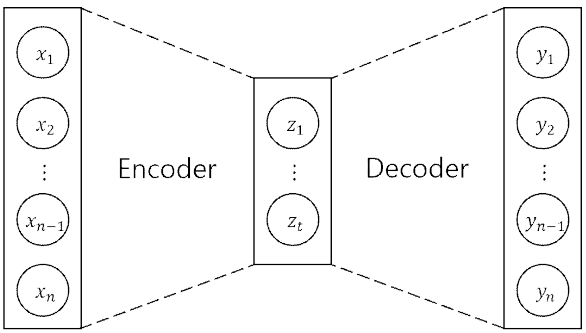


Fig. 1. Basic Auto-Encoder Architecture

Fig. 1에서의 x_1, x_2, ... x_{n-1}, x_n은 입력 데이터를 나타내고, z_1, ... z_t는 히든 레이어의 출력

데이터이면서 동시에 오토인코더를 통해 차원 압축된 데이터이다. y_1, y_2, ... y_{n-1}, y_n은 오토인코더의 출력을 의미한다. 이를 수식적으로 표현하면 다음과 같다.

$$z = a_{en}(\sum w_{en} x + b_{en})$$
$$y = a_{de}(\sum w_{de} z + b_{de})$$

위의 수식은 순서대로 오토인코더의 인코더, 디코더에 대한 수식이다. 여기서 w는 가중치(Weight), b는 바이어스(Bias), a는 활성 함수(Activation function)를 나타낸다.

오토인코더는 학습을 통하여 데이터의 특징을 가장 잘 표현하는 인코더 가중치들을 구성하고, 인코더 출력을 입력 데이터와 유사하도록 재구성해주는 디코더 가중치들을 구성한다. 즉, 오토인코더를 학습한다는 것은 출력 y_1, y_2, ... y_{n-1}, y_n과 입력 x_1, x_2, ... x_{n-1}, x_n의 차이가 작아지도록 네트워크의 가중치들을 수정하는 과정이다. 이를 수식으로 표현하면 아래와 같다.

$$\theta_{en}, \theta_{de} = \underset{\theta_{en},\theta_{de}}{argmin} \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i)$$
$$= \underset{\theta_{en},\theta_{de}}{argmin} \frac{1}{n}\sum_{i=1}^{n} L\left(x_i, f_{\theta_{de}}\left(f_{\theta_{en}}(x_i)\right)\right)$$

θ_{en}은 인코더에 사용되는 파라미터들을, θ_{de}은 디코더에서 사용되는 파라미터들을 나타내고, L은 손실 함수(Loss function)을 의미한다. 손실 함수로는 평균 제곱 오차(MSE; Mean Squared Error) 등을 사용한다.

앞서 설명한 오토인코더와 유사하게 노이즈 제거에 사용되는 오토인코더 또한 기본적인 오토인코더와 네트워크 구조는 일치하지만, 학습에 사용되는 데이터에서 차이가 존재한다[3]. 노이즈 제거 오토인코더(Denoising Auto-Encoder)는 학습 데이터에 노이즈를 추가하여 입력 데이터로 사용하며, 출력에 대한 레이블을 노이즈가 추가되기 이전의 데이터로 설정하여 학습한다. 이러한 방법을 통하여 노이즈가 제거된 데이터를 출력하도록 학습된 딥러닝 네트워크를 설계할 수 있다.

III. 제안기법

본 논문에서는 노이즈 제거 오토인코더의 원리를 부채널 분석 환경에 맞게 변형하여 파형의 노이즈를 제거하는 전처리 기법을 제안한다.

상관전력분석은 장비의 소비 전력과 연산 되는 값 사이의 상관관계가 존재하는 것을 이용한다. 따라서 두 값 사이의 상관성이 높을수록, 즉 노이즈가 작을수록 분석 성능이 올라간다.

변수 D는 데이터, δ는 고정된 상수인 오프셋, $H(\cdot)$는 해밍 웨이트 함수, B는 평균이 0인 표준 가우시안 랜덤일 때 공격 장비의 전력 모델이 $L = \delta + H(D) + B$ 라고 가정한다면, 같은 데이터(D')로부터 수집된 서로 다른 전력 파형들의 평균은, 충분히 많은 파형 수에 대해서, 노이즈가 제거된 $\delta + H(D')$가 된다[6]. 이와 같은 가정을 통해서 같은 중간값에 대해 수집한 파형들의 평균 파형을 노이즈가 제거된 파형이라고 충분히 가정할 수 있다.

따라서 같은 연산 중간값에 대한 여러 파형의 평균 전력 파형을 해당 집합의 파형들의 레이블로 하는 데이터베이스를 생성하여 오토인코더를 학습한다면 해당 파형들에 대한 노이즈 제거 오토인코더를 설계할 수 있다.

AES와 같은 블록 암호에서 비밀키가 고정일 때 같은 평문에 대해 같은 중간값을 가지므로, 동일한 평문에 대한 평균 파형을 레이블로 사용하여 제안기법을 수행할 수 있다.

IV. 실험결과

본 논문의 실험은 동작 주파수 7.37MHz의 Atmel XMEGA128 (8-bit 프로세서) 보드에서 임의의 평문에 대한 AES-128[4] 암호 알고리즘 1 Round 연산을 수행할 때, ChipWhisperer-Lite를 이용하여 클락당 샘플수 4(point/clock)로 총 3,514포인트의 100,000개 전력 파형을 수집하여 사용하였다. 본 논문의 제안기법인 오토인코더는 TensorFlow[1] 1.10.0 버전 내부의 Keras 라이브러리를 이용하여 구현하였다.

Fig. 2는 AES 1 Round의 SubBytes의 첫 번째 State에 대한 전력 파형과 256개의 키 추측에 따라 계산한 중간값의 해밍 웨이트(Hamming Weight) 사이의 상관전력분석 결과인 파형 256개에 대해, 시점별로 올바른 키에 대한 Correlation 값을 틀린 키에 대한 Correlation 값 중 가장 큰 값으로 나눈 값이다.

총 시점 3,514포인트 중에서 'CPA' 파형을 기준으로 올바른 키에 대한 Correlation이 절댓값 0.5 이상에 해당하는 60포인트 시점으로만 선별하여 Fig. 2에 나타내었다.

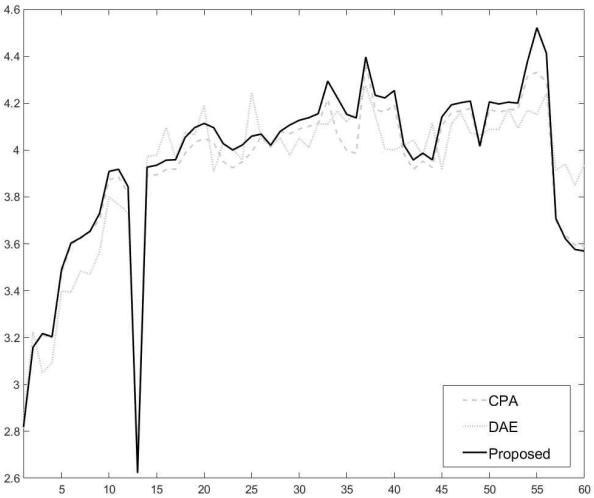


Fig 2. Right Key to Wrong Key Rate

Fig. 2의 'CPA'의 그래프는 전처리 없는 파형에 대해서, 'DAE'는 기존에 제안된 노이즈 제거 오토인코더를 적용하여 전처리한 파형에 대해서, 'Proposed'는 제안기법을 이용하여 전처리한 파형에 대해 분석한 파형이다. 그리고 X축은 Time Samples, Y축은 Rates를 나타낸다.

Fig. 2에서 X축의 55포인트인 시점을 보면 'Proposed'가 4.522로 분석 결과 중에서 가장 높은 수치인 것을 확인할 수 있다. 그리고 'Proposed'가 'CPA'보다 높으면서 동시에 'DAE' 보다도 높은 포인트는 Fig. 2에서 총 60포인트 중 40포인트로 전체적인 구간에서 제안기법의 결과가 높은 것을 볼 수 있다.

V. 결론

본 논문에서 제안하는 기법을 통하여 공격자에게 요구되는 파형 처리 능력을 최소로 하면서도 좋은 성능의 노이즈 제거 전처리를 수행할 수 있다. 또한 주성분분석(PCA), 선형판별분석(LDA), SSA와 같은 기법들을 제안기법과 함께 결합하여 적용한다면 더 다양한 알고리즘에 대한 적용 및 기존보다 좋은 노이즈 제거 성능을 기대할 수 있을 것이다.

본 논문의 실험에서는 고정된 하이퍼 파라미터에 대해서만 실험을 하였지만, 하이퍼 파라미터의 세부적인 조정을 통해 더욱 개선된 오토인코더를 설계할 수 있을 것으로 보인다.

[참고문헌]

[1] Mart´ın Abadi, et al, TensorFlow: Large-scale machine learning on heterogeneous systems, Preliminary White Paper, November, 2015.

[2] Eric Brier, Christophe Clavier, and Francis Olivier, Correlation power analysis with a leakage model, International workshop on cryptographic hardware and embedded systems, August, 2004.

[3] Pascal Vincent, et al, Extracting and Composing Robust Features with Denoising Autoencoders, Proceedings of the 25th international conference on Machine learning. ACM, July, 2008.

[4] Information Technology - Security Techniques - Encryption Algorithms - Part 3: Block Ciphers, ISO/IEC 18033-3:2005, 2005.

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, Greedy Layer-Wise Training of Deep Networks, Advances in neural information processing systems, December, 2007.

[6] Prouff Emmanuel, Matthieu Rivain, and Régis Bevan, Statistical analysis of second order differential power analysis, IEEE Transactions on computers 6, June, 2009.

[7] Paul Kocher, Joshua Jaffe, and Benjamin Jun, Differential power analysis, Annual International Cryptology Conference, August, 1999.

캐시 부채널 공격 방어 기법 연구

전성일, 김기열, 김현준, 이영한, 백윤흥*

*서울대학교 전기정보공학부

A Study on Cache side channel attack defense mechanisms

Seongil Jeon, Giyeol Kim, Hyunjun Kim, Younghan Lee, Yun heung Paek*

*Department of Electrical and Computer Engineering, Seoul National University.

요 약

최근 들어, 다양한 캐시 부채널 공격들이 시도되고 있으며, 이를 막기 위한 방어 기법들도 활발히 연구되고 있다. 본 논문에서는, 캐시 부채널 공격의 대표적인 예를 소개하고 이를 막기 위한 하드웨어 기반 방식 두 가지와 소프트웨어 기반 방식 한 가지를 소개한다. 하드웨어 기반 방식은 캐시를 수정하여 공격자로 하여금 보호되어야 하는 캐시 라인의 접근을 어렵게 만드는 방식이고, 소프트웨어 기반 방식은 캐시 접근에 따른 접근 시간 정보에 노이즈를 추가하도록 OS 커널을 수정하여 캐시 부채널 공격이 어렵도록 만드는 방식이다. 본 논문에서는 위에 소개한 방어 기법 세 가지를 다룬 연구들을 살펴본다.

I. 서론

최근 들어, 다양한 방법의 캐시 부채널 공격들의 가능성이 제시되고 있다. 캐시 부채널 공격을 통해 공격자는 AES나 RSA와 같은 암호 알고리즘의 키 값을 알아낼 수 있다.

1.1 캐시 부채널 공격 기법

캐시 부채널 공격이 일어나는 대표적인 상황은 공격자와 희생자(프로세스)가 서로 캐시 메모리를 공유하여 공격자가 희생자 프로세스가 접근하는 캐시 메모리를 접근할 수 있을 때 발생하는 상황이다. 이 상황에서는 공격자가 보다 적극적으로 공격을 할 수 있는데, 예를 들면, 공격자가 임의의 캐시 라인에 자신의 메모리 블록을 적재시켜 놓고, 희생자가 해당 캐시 라인을 접근하여 공격자의 캐시 라인을 밀어냄(Eviction)으로써 공격자가 자신의 메모리 블록을 다시 읽어 올 때, 캐시 미스가 발생함으로써 생기는 시간을 측정하여 희생자의 캐시 접근 정보를 알아낸다.

1.2 캐시 부채널 방어 기법

위에서 소개한 공격 기법은 근본적으로 프로세스 간 특정 캐시 라인을 가져가고자 서로 밀어내는 캐시 간섭 현상(Cache interference)에 의해서 발생되는 것이다. 이 근본적인 캐시 간섭 현상을 해결하기 위한 다양한 접근들이 존재하지만, 본 논문에서는 이 현상을 방지할 수 있는 하드웨어를 수정하는 두 가지 기법을 소개한다. 그리고, 캐시 간섭 현상 자체를 막는 것은 아니지만 공격자로 하여금 캐시 부채널 공격이 어렵게끔 노이즈를 삽입하는 소프트웨어(OS) 기반 방어 기법을 소개한다.

1.2.1 하드웨어 기반 방어 기법

[PLCache] 하드웨어 기반 방어 기법 중 첫 번째 기법은 다른 프로세스로 하여금 캐시 라인 접근 자체를 방지하는 방법이다.

[RPCache] 하드웨어 기반 방어 기법 중 두 번째는 다른 프로세스로 하여금 캐시 라인의

접근은 허용하지만 보호되어야 할 캐시 라인의 적재 위치를 계속해서 랜덤화 하여 공격자 프로세스로 하여금 유의미한 시간 측정 정보를 얻을 수 없도록 하는 방법이다.

1.2.2 소프트웨어 기반 방어 기법

[Düppel] 소개할 소프트웨어 기반 방어 기법은 소프트웨어 코드를 수정하여 캐시 접근 정보에 노이즈를 추가하여 공격자가 올바른 시간 정보를 획득할 수 없도록 하는 방법이다.

II. 캐시 부채널 공격 기법

[FLUSH+RELOAD 공격]

FLUSH+RELOAD 공격은 공격자가 희생자와 공유하는 LLC(Last-level-cache)에서의 공격을 가정한다. 이 공격에서 공격자는 x86 아키텍쳐가 가지고 있는 clflush라는 명령어를 사용하여 모든 캐시 레벨의 라인을 비울 수 있는 능력을 가진다. 이를 활용하여, 공격자는 모든 캐시 라인을 flush한 후, 대기시간을 가진 뒤, 그 사이에 희생자 프로세스의 캐시 접근이 일어나는 시간 동안 기다린 뒤(wait), 희생자가 접근했을 것이라고 예상되는 영역을 reload하여 시간을 측정해 희생자의 캐시 접근 정보를 유추해내는 공격이다.

이 공격이 가능할 수 있었던 큰 이유는 Last-level-cache를 서로 공유하기 때문에 공격자 프로세스와 희생자 프로세스가 특정 캐시 라인을 놓고 경쟁하는 캐시 간섭(Cache interference) 현상이 발생했기 때문이다.

III. 하드웨어 기반 캐시 부채널 방어 기법

소개할 하드웨어 기반 캐시 부채널 방어 기법 두 가지는 캐시 간섭 현상 자체를 방지하고자 하는 접근이 전제가 된다.

3.1 PLCache

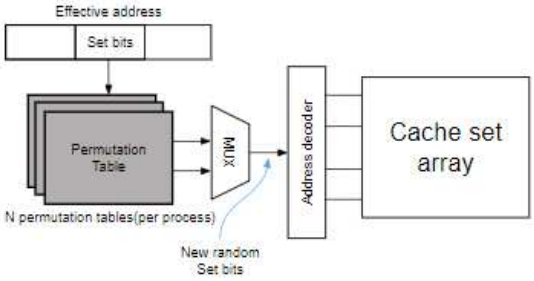
PLCache는 공격자로 하여금 허용되지 않은 캐시 영역에 대한 접근 자체를 차단함으로써, 캐시 간섭 현상을 방지하는 기법이다. 이를 구현하기 위해, 기존 Cache 라인에 L 비트와 ID 비트를 추가하여 구현하였는데([그림 1]), L 비트는 해당 캐시 라인이 잠김(Locked) 상태임을 의미하며, ID 비트는 잠김 되어있는 캐시 라인을 접근할 수 있는 권한을 의미한다. 만약 권한이 있는 프로세스가 캐시 라인을 방출하려 한다면, 캐시 라인은 자유롭게 방출되어 새 메모리 블록이 적재될 수 있다. 만약 권한이 없는 프로세스가 잠겨 있는 캐시 라인을 방출하고 자신의 메모리 블록을 적재하려 한다면, 그 경우는 캐시 적재가 불가능하고 Load 명령어의 경우 캐시를 우회하여 바로 core로 전달하고, Store의 경우 캐시를 우회하여 다음 단계 캐시에 적재한다.

위에 설명한 바와 같이 공격자 프로세스의 공격이 적극적으로 발생하는 1.1의 첫 문단의 공격을 막을 수 있고 1.1의 두 번째 문단에서 언급한 코드 내부에서 발생하는 캐시 간섭 현상 또한 코드 영역마다 따로 잠금을 걸어놓음으로써 막을 수 있다.

[그림 1] PLCache line

3.2 RPCache

RPCache는 공격자로 하여금 희생자의 캐시 영역으로의 접근은 허용하지만 아키텍쳐 내부적으로 캐시 라인의 적재를 랜덤하게 바꾸기 때문에 공격자가 유의미한 수행 시간 측정 정보를 알 수 없게 하는 것이 핵심이다. 이를 구현하기 위해([그림 2]), 캐시로 적재 될 때마다, 원래는 캐시로 적재되는 Set 비트가 고정되어 있었다면, RPCache에서는 Permutation Table을 통해 Set 비트가 매번 바뀌어서 캐시로 적재되도록 구현되었다.



[그림 2] RPCache structure

이 구조에서 만약 공격자가 특정 라인에 자신의 메모리 블록을 적재하여 캐시 간섭 현상을 일으키고자 할 때, 현재 캐시 라인에 있는 캐시 블록의 매핑이 Permutation table에 의해 정해진 특정한 랜덤 세트로 매핑되도록 두 라인을 뒤바꾸고 이 후의 올바른 접근을 위해 매핑 정보를 테이블 내에서도 뒤바꾼다. 또한, 기존에 캐시 라인에 값이 존재했다면(캐시의 dirty bit이 1이었다면), 이 값은 더 이상 쓸 수 없으므로 메인 메모리로 Write back 한다.

코드 내부에서 캐시 간섭이 발생하는 상황에 대한 방어도 RPCache를 통해 가능하다. 이를 막기 위해서 PLCache와 비슷하게 캐시 라인에 별도의 태그를 붙이는데[그림 3], 여기서 P 값을 통해 코드 내부에서 코드 간 발생하는 캐시 간섭 현상을 방지할 수 있다.

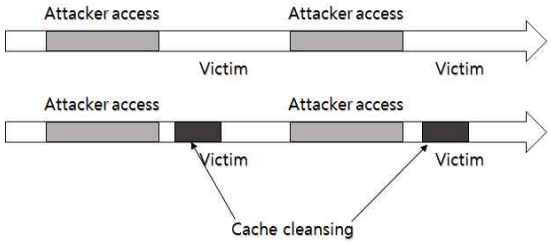
[그림 3] RPCache line

이를 막기 위해, 코드 영역별로 중요하다고 생각되는 구간(키가 사용되는 RSA 혹은 AES의 테이블)과 그렇지 않은 구간의 P 값을 다르게 할당하여 P 값을 비교하여 코드 영역이 다를 때 캐시 라인에 메모리 블록 적재가 랜덤하게 발생하도록 할 수 있다.

IV. 소프트웨어 기반 방어 기법

앞서 소개한 하드웨어 기반 방어 기법이 아닌 소프트웨어 코드를 수정하여 캐시 부채널 공격을 완화시키는 방법 또한 존재한다. 이 방식은 하드웨어 기반 방식과는 달리 캐시 간섭 현상은 허용하되 공격자로 하여금 의미 있는 공격을 할 수 없도록 하고자 하는 접근을 전제로 한다.

소개할 기법은 Düppel 이라는 기법인데, 이 기법에서는 희생자 프로세스의 수행 시간 동안 공격자가 탐지할만한 중요한 코드 영역에 대해서 주기적으로 OS가 개입하여(Interrupt) Cache cleansing[그림 4]을 수행하도록 하여 공격자가 캐시에 접근할 때 올바른 시간 정보를 획득할 수 없도록 한다. 이를 통해 공격자는 캐시 부채널 공격을 해도 의미있는 값을 유추해내기 힘들게 된다.



[그림 4] Düppel Cache cleansing

V. 각 솔루션별 개선 방안

5.1 하드웨어 솔루션 - 개선 방안

PLCache 방식은 캐시 간섭이 일어나면 안되는 캐시 라인으로의 접근 자체를 막는 방식이었고, RPCache 방식은 접근은 허용하되, 캐시 라인 적재가 캐시의 특정 위치로 고정되어 발생하는 것이 아니라 캐시 라인 적재 위치가 계속해서 랜덤하게 바뀌는 방식을 활용한다. 두 가지 방식을 다룬 논문에서는 각각의 방식이 하드웨어 수정을 최소화하였기 때문에 오버헤드가 크지 않다고 했지만 특정 메모리 블록이 캐시를 우회하여 바로 파이프라인으로 넘겨지거나 메인 메모리로 넘겨지는 상황들이 빈번하게 발생됨으로써 생각보다 큰 오버헤드가 발생할 것으로 생각하고, 특별히 RPCache에서는 캐시 라인을 서로 뒤바꾸는 과정에서 기존 캐시 라인에 있던 값이 무효화되고 메인 메모리로 Write back 되어야한다는 점에서 오버헤드가 발생할 것이다. 이와 같이 메인 메모리로 Write back되는 캐시 라인들이 최소화된다면 성능 오버헤드를 더 줄일 수 있을 것으로 예상된다. 또한, 이 솔루션이 더욱 실용적으로 쓰이기 위해서는 실제 프로세서 칩으로 구현되기 위한 많은 제약사항들을 만족해야만 한다. 따라서, 이 구현에 대한 STA(Static Timing Analysis)를 수행하여 PLCache와 RPCache 각각의 Critical path를 계산하고 적절한 버퍼를 Critical path에 삽입하여 최대 클럭 주파수를 최대한 끌어 올려야만 실제로 프로세서 칩으로써 사용이 가능하리라 본다.

5.2 소프트웨어 솔루션 - 개선 방안

Düppel에서 사용하는 Cache cleansing 기법은 주기적으로 OS가 개입하기 위한 타이머가 잘 구현되어야 하는 이슈가 존재한다. 또한, 이 솔루션에서는 동시 멀티스레딩 기법을 사용하지 않는다는 가정이 존재하는데, 이를 사용하지 못해서 발생되는 성능 오버헤드는 발생할 수밖에 없다. 이 두 가지가 잘 해결되어야만 조금 더 실용적인 솔루션이 될 듯 하다.

VI. 결론

본 논문에서 캐시 부채널 공격을 방어하기 위한 하드웨어 기반 솔루션 2가지(PLCache, RPCache)와 소프트웨어 기반 솔루션 1가지(Düppel)를 알아보았는데, 각 솔루션마다 다른 아이디어로 구현이 되어있으며 각 솔루션마다 최적의 효과를 내는 적용들이 각각 다를 것이라고 생각한다. 따라서 이 솔루션들을 더 잘 활용하기 위해서는 적용할 어플리케이션을 잘 고려하여 솔루션을 선택하는 것이 중요할 것이라고 생각한다.

[참고문헌]

[1] Yarom, Yuval, and Katrina Falkner. "FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack." USENIX Security Symposium. Vol. 1. 2014.

[2] Wang, Zhenghong, and Ruby B. Lee. "New cache designs for thwarting software cache-based side channel attacks." ACM SIGARCH Computer Architecture News. Vol. 35. No. 2. ACM, 2007

[3] Zhang, Yinqian, and Michael K. Reiter. "Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud." Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013.

암호 알고리즘의 연산기반 최적 전력 소모 모델*

- Case Study : AES

김주환*, 심보연*, 한동국**, ***

*국민대학교 수학과, **국민대학교 금융정보보안학과, ***국민대학교 정보보안암호수학과

Optimal Power Consumption Model based on the computation of cryptographic algorithms

- Case Study : AES

Ju-Hwan Kim*, Bo-Yeon Sim*, Dong-Guk Han**, ***

*Dept, of Mathematics, Kookmin University,
**Dept, of Financial Information Security, Kookmin University,
***Dept, of Information Security, Cryptology, and Mathematics, Kookmin University

요 약

부채널 분석이란 장비에서 암호 알고리즘이 동작할 때 발생하는 소비전력, 전자파, 소리 등을 이용하여 비밀 정보를 분석하는 것을 말한다. [2]에서 8비트 기반으로 구현된 AES-128 암호 알고리즘이 ChipWhisperer 전력 수집 보드에서 동작할 때의 소비전력을 측정하여 최상위 비트(Most Significant Bit, MSB) 정보기반 상관 전력 분석(Correlation Power Analysis, CPA)과 1 바이트 해밍 웨이트 정보기반 CPA를 수행한 결과 MSB 기반의 CPA가 바이트 기반의 CPA보다 성능이 높음을 보였다. 본 논문에서는 특정 비트에 의존한 CPA 성능이 우월한 원인이 갈루아 체 $GF(2^8)$에서의 모듈로 연산 때문임을 증명하였다. MixColumns을 모듈로 연산과 행렬 곱 연산을 혼합하여 수행하는 방법, 모듈로 연산과 행렬 곱 연산을 분리하여 수행하는 방법 두 가지로 구현하여 실험을 통해 갈루아 체 $GF(2^8)$에서의 모듈로 연산을 수행하는 부분에서 MSB 정보가 누출됨을 보인다.

I. 개요

부채널 분석(Side Channel Analysis, SCA)은 장비에서 암호 알고리즘이 동작할 때 발생하는 소비전력, 전자파, 소리 등을 이용하여 비밀 정보를 분석하는 것을 말한다. 소비전력을 이용하는 대표적인 부채널 분석 방법은 암호 알고리즘의 중간값과 소비전력의 상관관계를 이용하는 상관 전력 분석(Correlation Power Analysis, CPA)이다.

CPA는 레지스터에 데이터를 저장할 때나 읽을 때의 소비전력이 데이터와 선형관계에 있음을 이용하는 분석 방법이다. 소프트웨어로 구현된 암호 알고리즘의 경우 데이터의 해밍 웨이트(Hamming Weight)[1] 정보에 의존한 전력 소비가 발생한다는 가정을 기반으로 한다.

8비트 기반으로 구현된 AES-128 암호 알고리즘에 대하여 CPA를 수행하는 경우 한 번에 전체 128비트의 키를 추측하는 것은 전수조사 공격과 동일한 공격 복잡도를 갖는다. 따라서 분할 정복 기법을 적용, 알고리즘 연산 단위인 8비트씩 키를 추측하여 CPA를 반복 수행하는 방법으로 전체 128비트의 키를 찾는 것이 일반적이다. 데이터의 모든 비트는 소비전력에 영향을 미치므로 추측한 키로 계산한 8비트 중간값의 모든 정보를 이용하여 CPA를 수행하는 것이 일반적이다.

* 본 연구는 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원으로 수행되었습니다. (No.2017-0-00520, SCR-Friendly 대칭키 암호 및 응용모드 개발)

그러나 이지우 등은 ChipWhisperer 보드에서 동작하는 8비트 단위 AES-128 암호 알고리즘을 분석한 결과 8비트 중간 값의 최상위 비트(Most Significant Bit, MSB) 정보만을 이용한 CPA가 8비트 해밍 웨이트 정보를 이용한 CPA보다 분석 성능이 높음을 보였다 [2]. 그리고 이에 대한 원인은 ChipWhisperer 보드에서 MSB가 소비전력에 미치는 영향이 다른 비트에 비해 월등히 크기 때문이라고 주장하였다.

본 논문에서는 [2]의 실험 결과가 도출된 이유가 ChipWhisperer 보드에서 MSB가 소비전력에 미치는 영향이 월등히 크기 때문이라는 원 논문의 주장과 달리 특정 비트에 의존한 CPA 성능이 우월한 원인이 갈루아 체 $GF(2^8)$에서의 모듈로 연산 때문임을 증명한다. MixColumns을 모듈로 연산과 행렬 곱 연산을 혼합하여 수행하는 방법, 모듈로 연산과 행렬 곱 연산을 분리하여 수행하는 방법 두 가지로 구현하여 실험을 통해 갈루아 체 $GF(2^8)$에서의 모듈로 연산을 수행하는 부분에서 MSB 정보가 누출됨을 보인다. 따라서 암호 알고리즘의 연산에 따라 적합한 소비전력 모델을 선택해야 함을 보인다.

II. 관련 연구

S_1을 레지스터에 저장되어있는 데이터(입력 값), S_2를 레지스터에 저장할 데이터(출력 값)라 하면, 레지스터에 S_2을 저장할 때 소비하는 전력량 P_{total}은 아래 수식과 같다. 수식에서 ϵ은 상수, P_{noise}는 노이즈, f는 소비전력 모델이다.

$$P_{\text{total}} = \epsilon \cdot f(S_1,\ S_2) + P_{\text{noise}}$$

소프트웨어로 구현된 암호 알고리즘을 분석할 때에는 일반적으로 소비전력 모델을 해밍 웨이트 모델로 선택한다. 즉, 소비전력이 아래 수식과 같으며, 레지스터에 저장하는 데이터의 해밍 웨이트 값과 소비전력은 선형성을 띤다.

$$P_{\text{total}} = \epsilon \cdot \text{HW}(S_2) + P_{\text{noise}}$$

CPA는 알려진 다수의 평문 m을 입력으로 수집한 소비전력 파형 집합을 X, 평문 m과 추측한 비밀 키 k로 계산한 중간값의 해밍 웨이트 집합을 Y라 할 때 두 집합의 상관계수 값을 계산한다. 피어슨의 상관계수의 정의에서 두 집합 X, Y가 선형성을 띤다면 상관계수가 ± 1에 가까워진다.

$$\text{Corr}(X,\ Y) = \frac{\text{Cov}(X,\ Y)}{\sqrt{\text{Var}(X) \cdot \text{Var}(Y)}}$$

따라서 CPA는 소비전력 파형 집합 X와 높은 상관계수 값을 갖는 Y 집합을 계산할 때의 추측 키를 올바른 키라고 추정한다.

8비트 데이터 $S_2 = (a_1 a_2 a_3 \cdots a_8)_2$의 해밍 웨이트 정보를 모두 이용하는 바이트 기반의 CPA와 MSB 정보만을 이용하는 MSB 기반의 CPA가 있다. 각각의 경우 소비전력을 다음 수식처럼 추측한다.

$$P_{\text{total}} = \epsilon \cdot \sum_{i=1}^{8} a_i + P_{\text{noise}}$$
$$P_{\text{total}} = \epsilon \cdot a_1 + P_{\text{noise}}$$

III. MSB가 Byte 정보보다 우월한 이유

본 장에서는 [2]의 실험결과에 대한 원인을 두 가지 실험을 통하여 분석한다. 실험 환경은 [2]와 동일하다 8비트 단위 소프트웨어로 구현된 AES-128 암호 알고리즘이 ChipWhisperer 보드에서 1,000번 동작할 때의 소비전력을 수집한 뒤, 중간값은 S-Box 출력으로 하여 MSB 기반의 CPA와 바이트 기반의 CPA를 수행한다. 이때 분석 위치는 각 실험의 목적에 맞게 구성한다.

3.1 MSP CPA v.s. Byte CPA

전체 16바이트 중 첫 번째 바이트를 분석한 결과는 Fig. 1.과 같다. 첫 번째 그래프는 소비전력을 나타낸 그래프이다. 선을 이용하여 각 함수별 연산 위치를 표기하였는데 순서대로 AddRoundKey, SubBytes, ShiftRows, MixColumns 연산을 수행하는 위치이다. 두 번째와 세 번째 그래프는 첫 번째 바이트를 각각 바이트 기반 CPA, MSB 기반 CPA로 분석한 결과이다.

CPA 수행 결과 SubBytes, MixColumns 두 부분에서의 상관계수가 다른 부분에 비해 높음을 확인할 수 있다. 바이트 기반 CPA의 경우 두 부분에서의 상관계수가 약 0.8로 비슷하지만, MSB 기반 CPA는 MixColumns 부분의 상관계수가 0.9, SubBytes 부분의 상관계수가 0.3으로 MixColumns 부분의 상관계수가 약 3배 더 큰 것을 확인할 수 있다. 즉, MixColumns 연산이 수행되는 위치의 전력 파형에 MSB 정보가 더 많이 있음을 알 수 있다. Fig. 2.는 바이트 기반의 CPA는 두 부분에서의 상관계수가 유사하지만, MSB 기반의 CPA는 MixColumns 부분의 상관계수가 SubBytes 부분의 상관계수보다 약 3배 더 큼을 보인다.

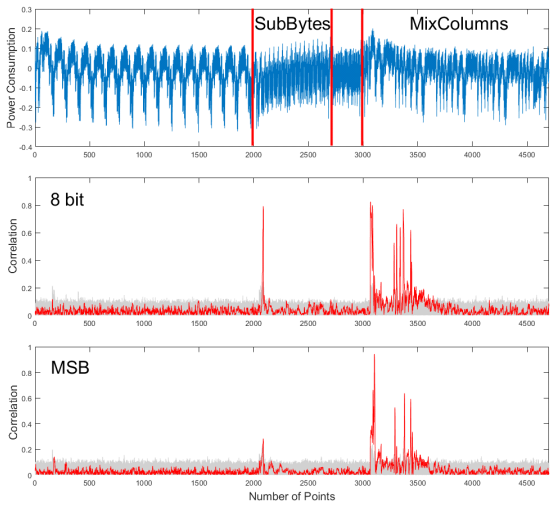


Fig. 1. ChipWhisperer 1 byte CPA result

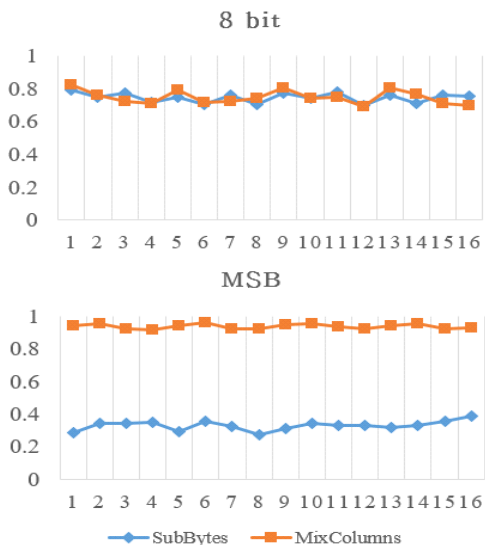


Fig. 2. ChipWhisperer 1 round CPA result
(X-axis : byte, Y-axis : correlation coefficient)

다음 절에서는 MixColumns 연산을 분석하여 MSB 정보가 누출되는 이유는 갈루아 체 $GF(2^8)$에서의 모듈로 연산 때문임을 보인다.

3.2 MSB 정보 누출 이유

AES의 MixColumns 연산은 다음과 같이 구성되어 있다. 이때, 바이트 단위 연산 $c \bullet s_{i,j}$ ($c \in \{1,2,3\}$, $0 \le i,j \le 3$)는 갈루아 체 $GF(2^8)$에서 이루어진다. 이 때, $GF(2^8)$에서의 기약다항식은 $x^8+x^4+x^3+x+1$ 이다.

$$\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

따라서 갈루아 체 $GF(2^8)$에서의 2배 연산(xtime)은 수식 (1)과 같이 $s_{i,j}$를 2배한 후, $s_{i,j}$ 값의 MSB 값에 따라 모듈로 연산을 추가로 수행해야 하는지를 확인해야 한다.

$$2 \bullet s_{i,j} = \begin{cases} s_{i,j} \ll 1, & \text{if } MSB(s_{i,j})=0 \\ (s_{i,j} \ll 1) \oplus 0x1b, & \text{if } MSB(s_{i,j})=1 \end{cases} \quad \cdots \quad (1)$$

즉, xtime 연산을 구현하기 위해서는 MSB 값을 확인하는 연산이 필요하다. 3배 연산은 고속화를 위해 일반적으로 $3 \bullet s_{i,j} = (2 \bullet s_{i,j}) + s_{i,j}$와 같이 계산한다.

3.3 MixColumns 코드 분석 및 실험 결과

MixColumns를 분석 위치로 하였을 때, MSB 기반의 CPA가 바이트 기반의 CPA보다 성능이 높은 원인이 갈루아 체 $GF(2^8)$에서의 모듈로 연산인지 확인하기 위하여 MixColumns를 두 가지 방법으로 구현하였다. Fig. 3.은 xtime 연산과 행렬 곱 연산을 혼합하여 수행하도록 구현된 MixColumns v1 코드이다. Fig. 4.는 xtime 연산과 행렬 곱 연산을 분리하여 수행하도록 구현된 MixColumns v2 코드이다. 그림의 Part 1이 xtime 연산을 수행하는 부분이고, Part 2가 행렬 곱 연산을 수행하는 부분이다.

```
void MixColumns(unsigned char STATE[16])
{
  unsigned char TEMP[4];

  TEMP[0] =   (STATE[0] << 1) ^ (0x1b * (STATE[0] >> 7))
            ^ (STATE[1] << 1) ^ (0x1b * (STATE[1] >> 7)) ^ STATE[1] ^ STATE[2] ^ STATE[3];
  TEMP[1] =   (STATE[1] << 1) ^ (0x1b * (STATE[1] >> 7))
            ^ (STATE[2] << 1) ^ (0x1b * (STATE[2] >> 7)) ^ STATE[2] ^ STATE[3] ^ STATE[0];
  TEMP[2] =   (STATE[2] << 1) ^ (0x1b * (STATE[2] >> 7))
            ^ (STATE[3] << 1) ^ (0x1b * (STATE[3] >> 7)) ^ STATE[3] ^ STATE[0] ^ STATE[1];
  TEMP[3] =   (STATE[3] << 1) ^ (0x1b * (STATE[3] >> 7))
            ^ (STATE[0] << 1) ^ (0x1b * (STATE[0] >> 7)) ^ STATE[0] ^ STATE[1] ^ STATE[2];
  STATE[0] = TEMP[0];
  STATE[1] = TEMP[1];
  STATE[2] = TEMP[2];
  STATE[3] = TEMP[3];
```

Fig. 3. MixColumns v1 code

```
void MixColumns(unsigned char STATE[16])
{
   int i = 0;
   unsigned char TEMP[16], Mc[16];

   for (i = 0; i < 16; i++)
      TEMP[i] = STATE[i];

   for (i = 0; i < 16; i++)                                          Part 1
      Mc[i] = (STATE[i] << 1) ^ (0x1b * (STATE[i] >> 7));

   for (i = 0; i < 16; i += 4)                                       Part 2
   {
      STATE[i]     = Mc[i]   ^ Mc[i + 1]   ^ TEMP[i + 1] ^ TEMP[i + 2] ^ TEMP[i + 3];
      STATE[i + 1] = TEMP[i] ^ Mc[i + 1]   ^ Mc[i + 2]   ^ TEMP[i + 2] ^ TEMP[i + 3];
      STATE[i + 2] = TEMP[i] ^ TEMP[i + 1] ^ Mc[i + 2]   ^ Mc[i + 3]   ^ TEMP[i + 3];
      STATE[i + 3] = Mc[i]   ^ TEMP[i]     ^ TEMP[i + 1] ^ TEMP[i + 2] ^ Mc[i + 3];
   }
}
```

Fig. 4. MixColumns v2 code

MixColumns v1에 대한 CPA를 수행한 결과가 Fig. 5.와 같다. 실험결과 MSB 기반의 CPA 수행 시 상관계수가 바이트 기반 CPA 수행 시 상관계수보다 높게 나왔다.

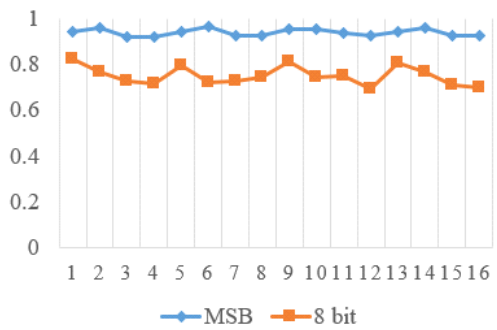


Fig. 5. MixColumns_v1 CPA result
(X-axis : byte, Y-axis : correlation coefficient)

MixColumns v2에 대한 실험 결과는 Fig. 6.과 같다. xtime 연산을 수행하는 Part 1에서는 MSB 기반 CPA의 성능이 좋고, 행렬 곱 연산을 수행하는 Part 2에서는 바이트 기반 CPA의 성능이 좋다. 이를 통해 MixColumns 부분에서 MSB 기반의 CPA가 바이트 기반의 CPA보다 성능이 높은 이유는 갈루아 체 $GF(2^8)$에서의 모듈로 연산 때문임을 알 수 있다.

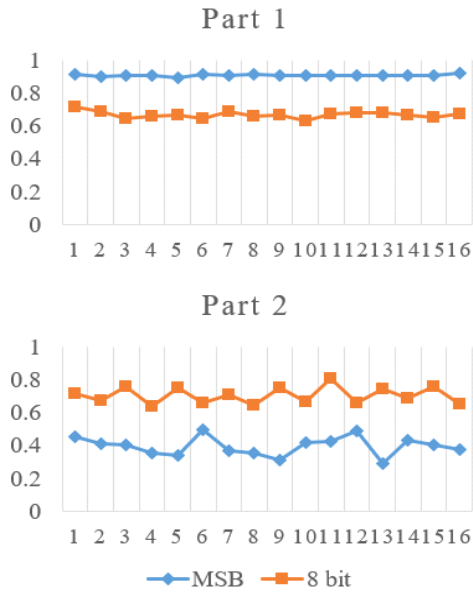


Fig. 6. MixColumns_v2 CPA result
(X-axis : byte, Y-axis : correlation coefficient)

IV. 결론

본 논문에서 MixColumns을 xtime 연산과 행렬 곱 연산 부분으로 분리하여 수행하도록 AES를 구현하여, MSB 기반의 CPA와 바이트 기반의 CPA를 수행하였다. 그 결과 xtime 연산 부분에서는 MSB 기반의 CPA의 성능이 높았고, 행렬 곱 연산 부분에서는 바이트 기반의 성능이 높았다. 따라서 갈루아 체에서의 모듈로 연산 부분에서 MSB 정보가 누출됨을 실험을 통해 검증하였다. 이는 암호 알고리즘 구현 방법에 따라 최적의 전력 소비 모델 선택의 필요성을 시사한다.

[참고문헌]

[1] Stefan Mangard, Elisabeth Oswald, Thomas Popp, "Power Analysis Attacks Revealing the Secrets of Smart Cards", Springer, 2007,

[2] 이지우, 김주환, 이종혁, 한동국 "Most Significant Bit가 Most Significant Power이다.", 정보보호학회 하계학술대회, 2018

부채널 분석 공격에 대응하는 초경량 블록 암호 CHAM 구현을 위한 축소 마스킹 기법 적용

권 홍 필[1], 배 대 현[2], 하 재 철[1†]

[1]호서대학교

Application of Reduced Masking Method to Implement Lightweight Block Cipher CHAM Against Side-Channel Analysis Attack

Hong-Pil Kwon[1], Dae-Hyeon Bae[2], Jae-Cheol Ha[1‡]

[1]Hoseo University

요 약

CHAM 암호는 국내에서 개발된 경량 블록 암호이다. 경량 블록 암호의 특성상 제한된 환경에 본 암호가 구현되었을 때 물리적 정보를 바탕으로 하는 부채널 분석 공격에 취약할 수 있다. 이러한 블록 암호에 대한 부채널 분석 공격의 대응으로 마스킹 기법이 주로 사용되고 있으나 마스킹을 설계하는 방식이나 알고리즘에 따라서도 연산 성능의 차이가 발생할 수 있다. 특히 CHAM 암호는 특성상 라운드 수가 다른 경량 암호 알고리즘들에 비해 많기 때문에 비교적 마스킹을 적용하였을 때 비효율적이다. 본 논문에서는 CHAM 암호에 대한 여러 가지 마스킹을 적용하여 구현함으로서 마스킹 기법 간의 효율성을 비교 분석하고 또한, 모든 라운드가 아닌 부분의 라운드에만 마스킹을 적용하는 축소 마스킹을 적용하였을 때 어느 정도 연산량이 감소하는지를 확인함으로서 효율성을 분석한다.

I. 서 론

최근에는 모바일 기기나 IoT 장비들과 같은 경량화 된 디바이스에 대한 소요가 증가하고 있다. 이러한 소요에 맞추어 경량화 된 디바이스에 적용될 암호 알고리즘에 대한 연구도 활발하게 진행되어지고 있다. 상기한 경량화 된 디바이스 환경에서는 전력, 메모리 등의 제한사항이 존재하기 때문에 기존에 개발된 암호 알고리즘을 적용하는 것이 현실적으로 불가능하다. 따라서 경량화 된 디바이스 환경에 맞게 경량 구현되어진 암호 모듈을 적용하여야 한다.

하지만 경량 블록 암호들은 저전력, 저사양의 환경에서 구현되어지고 경량 암호 모듈자체에도 기존의 암호 모듈만큼의 보안성을 갖기 어렵기 때문에 물리적 정보를 분석해 공격하는 부채널 분석 공격(Side-Channel Analysis Attack)에 취약할 수 있다[1]. 이러한 경량 블록 암호에 대한 부채널 공격은 공격자가 올바른 중간 값을 추출할 수 없도록 하여 대응할 수 있다. 올바른 중간 값을 추출하지 못하게 하는 방법에는 중간 값에 랜덤 한 값을 추가 연산함으로서 중간 값을 감추는 마스킹(masking) 기법이 대표적이다. 본 논문에서는 최근 국내에서 개발된 초경량 블록 암호 CHAM에 상기한 마스킹 기법을 적용하기 위한 설계 기법을 제시한다. 또한 여러 가지 마스킹 알고리즘들을 CHAM 암호에 적용하여 32비트 마이크로프로세서인 Cortex-M3 보드에 구현하고 나온 결과를 통해 CHAM 암호에 적용된 마스킹 기법간의 효율성을 비교 분석한다. 마지막으로 마스킹 기법을 적용한 CHAM 암호에 대한 효율성을 높이기 위해 부분의 라운드에만 마스킹을 적용하는 축소 마스킹(reduced masking)을 적용하여 그 효율성을 알아본다.

† 교신저자, jcha@hoseo.edu

II. 데이터 마스킹 기법

마스킹 기법은 블록 암호 알고리즘이 실행되는 동안 발생하는 중간 값에 랜덤 한 어떠한 값을 추가 연산하여 올바른 중간 값을 추출할 수 없게 하는 부채널 공격 대응 기법이다. 마스킹 기법은 일반적으로 암호 라운드의 연산 종류에 따라 부울 마스킹과 산술 마스킹으로 적용된다. 입력 x에 대한 부울 마스킹과 산술 마스킹은 다음과 같이 연산되어진다.

- 부울 마스킹 : $x' = x \oplus r_x$
- 산술 마스킹 : $A = x - r_x \bmod 2^m$

여기서 x는 마스킹이 적용되기 전의 원래의 중간 값이고, r_x는 랜덤 수로 마스킹을 위해 사용되는 마스크 값이다. 또한, m은 산술 마스킹 연산의 입력 데이터 비트 길이를 나타낸다.

대부분의 블록 암호의 라운드 연산에는 부울 연산과 산술 연산이 혼용되어 사용하기 때문에 마스킹 기법 또한 연산에 따라 상호 변환해주는 과정이 필요하다. 변환과정을 거치지 않으면 원래의 값을 얻기 위해 부울 마스킹 또는 산술 마스킹 된 값에 마스크 값을 상쇄 연산하여도 서로 다른 종류의 연산으로 원래의 값과 마스크 값이 섞여 제대로 된 연산 값을 얻을 수 없기 때문에 서로 다른 연산에 대한 마스킹 변환 과정이 꼭 필요하다. 변환 과정에서 주의할 점은 변환되는 과정에서 원래의 값 x 또는 마스크 값 r가 노출되면 안 된다는 것이다. 변환 알고리즘이 실행되는 가운데 x 또는 r값이 노출되면 이 또한 부채널 공격의 포인트가 될 수 있기 때문이다.

2.1 부울-산술 마스킹 변환 기법

부울-산술 마스킹 변환 기법은 부울 마스킹 된 중간 값을 산술 연산을 위해 산술 마스킹 된 중간 값으로 변환하는 기법이다. 지금까지 부울-산술 마스킹 변환 기법 가운데 Goubin이 제안한 부울-산술 마스킹 변환 알고리즘의 연산량이 매우 적고 효율적이기 때문에 많은 암호 알고리즘에서 사용되어지고 있다[2]. Goubin의 부울-산술 마스킹 변환 기법은 부울 마스킹 된 값 $x' = x \oplus r$과 난수 r을 입력받아 연산을 통해 산술 마스킹 된 $A = x - r$값을 출력한다.

2.2 산술-부울 마스킹 변환 기법

산술-부울 마스킹 변환 기법은 산술 마스킹 된 중간 값을 부울 연산을 위해 부울 마스킹 된 중간 값으로 변환하는 기법이다. 본 논문에서는 CHAM 암호에 대한 산술-부울 마스킹 변환 기법으로 Goubin 기법과 C-T 기법을 적용하였다.

2.2.1 Goubin 산술-부울 마스킹 변환 기법

Goubin이 제안한 산술-부울 마스킹 변환 기법은 부울-산술 마스킹 변환 기법과 함께 제안되었으며 산술 마스킹 된 중간 값 $A(=x-r)$와 마스크 값 r을 입력으로 하고 알고리즘 연산을 통해 부울 마스킹 된 값 $x' = x \oplus r$을 출력한다. Goubin 기법의 특징은 입력 비트 길이 m만큼의 반복 연산이 진행되어 이론상 많은 연산량을 필요로 한다는 것이다. 즉, m비트 연산을 처리하는 경우 $(2m+4)$번의 XOR 연산, $(2m+1)$번의 AND 연산, m번의 Shift 연산으로 총 $(5m+5)$번의 기본 연산이 소요된다. 다음 Fig. 1은 Goubin이 제안한 산술-부울 마스킹 변환 알고리즘을 나타낸 것이다.

```
Input : A = x - r , r
Output : x' = x ⊕ r
1. Γ ← γ
2. T ← 2Γ
3. x' ← Γ ⊕ r
4. Ω ← Γ ∧ x'
5. x' ← T ⊕ A
6. Γ ← Γ ⊕ x'
7. Γ ← Γ ∧ r
8. Ω ← Ω ⊕ Γ
9. Γ ← T ∧ A
10. Ω ← Ω ⊕ Γ
11. for i = 1 to m - 1 do
    Γ ← T ∧ r
    Γ ← Γ ⊕ Ω
    T ← T ∧ A
    Γ ← Γ ⊕ T
    T ← 2Γ
end for
12. x' ← x' ⊕ T
```

Fig. 1. Goubin's AtoB conversion method

2.2.2 C-T 산술-부울 마스킹 변환 기법

Coron 등은 Goubin이 제안한 산술-부울 변환 기법이 필요로 하는 연산량이 많아 비효율적이라는 것을 지적하며 룩업 테이블 기반의 새로운 산술-부울 마스킹 변환 기법(이하 C-T 기법)을

제안하였다[3]. 즉, $x' = (A + r_x) \oplus r_x$가 m비트 단위의 $(A + r_x)$값과 부울 연산할 때 덧셈에 의해 발생하는 캐리($carry$)를 미리 계산하여 테이블에 저장해 둔 후, 실제 변환 연산이 수행될 때 저장해 놓은 테이블 값을 이용하는 기법이다. 이러한 룩업 테이블을 이용하면 필요한 임시 메모리 크기가 늘어나지만 연산 속도를 높일 수 있다.

III. Secure Addition 마스킹 기법

기존의 마스킹 상호 변환 기법을 이용해 부울 마스킹 된 두 값을 더하기 위해서는 부울-산술 마스킹 변환, 산술 연산, 산술-부울 마스킹 변환의 3번의 과정을 거쳐야한다. 하지만 이러한 마스킹 상호 변환 기법의 비효율성과 많은 연산량을 개선하기 위해 SA기법이 제안되었다.

3.1. KRJ SA 마스킹 기법

Karroumi 등은 부울 마스킹 된 두 중간 값을 입력받아 새로운 값으로 부울 마스킹 된 덧셈결과를 출력하는 SA 마스킹 기법(이하 KRJ 기법)을 처음으로 제안하였다[4]. KRJ 기법은 논리적 AND, XOR, Shift 연산과 같은 기본적인 덧셈 알고리즘에 마스킹을 적용한 것이다. Fig. 2는 KRJ SA 알고리즘을 나타낸 것이다.

Input : x', y', r_x, r_y	
Output : z'	
1. $C = \gamma$	10. $B = \Omega \ll 1$
2. $T = x' \wedge y'$	11. $C = C \ll 1$
3. $\Omega = C \oplus T$	12. $z' = x' \oplus y'$
4. $T = x' \wedge r_y$	13. $r_z = r_x \oplus r_y$
5. $\Omega = \Omega \oplus T$	14. $T = C \wedge z'$
6. $T = y' \wedge r_x$	15. $\Omega = \Omega \oplus T$
7. $\Omega = \Omega \oplus T$	16. $T = C \wedge r_z$
8. $T = r_x \wedge r_y$	17. $\Omega = \Omega \oplus T$
9. $\Omega = \Omega \oplus T$	
18. for $i = 2$ to $k-1$ do	
18.1 $T = B \wedge z'$	18.4 $B = B \oplus T$
18.2 $B = B \wedge r_z$	18.5 $B = B \ll 1$
18.3 $B = B \oplus \Omega$	
19. $z' = z' \oplus B$	20. $z' = z' \oplus C$

Fig. 2. KRJ SA masking method

3.2. V-G SA 마스킹 기법

Vadnala와 Groβschdl는 KRJ 기법의 연산량을 개선하기 위해 사전 테이블을 이용한 SA 마스킹 기법(이하 V-G 기법)을 제안하였다[5]. VG 기법의 기본 개념은 m비트 단위의 $(x \oplus s_1) + (y \oplus s_2)$ 값을 계산할 때 발생하는 캐리를 미리 계산하여 실제 마스킹 연산 단계에서 이용하는 것이다.

IV. CHAM 알고리즘 및 마스킹 기법 적용

국내에서 개발된 CHAM 알고리즘은 입력된 평문을 4개의 블록으로 나누어 처리하는 Feistel 구조의 경량 블록 암호이다[6]. CHAM 암호는 키 스케줄 과정에서 키 상태를 업데이트하지 않는 특징이 있고, 라운드 키의 수를 라운드 수보다 적게 하여 라운드 키가 재사용되도록 설계함으로서 라운드 키를 저장하는데 필요한 메모리 크기를 줄였다. CHAM 암호의 라운드 연산은 좌측 순환 이동, 덧셈, XOR 연산으로 이루어져있으며, 입력 크기는 64~128비트, 키 길이는 128~256비트를 지원한다. 다음 Fig. 3은 CHAM 암호의 라운드 연산을 나타낸 것이다. Fig. 3과 같이 CHAM 암호의 라운드 연산은 라운드 i가 홀수일 때와 짝수일 때 좌측 순환 이동하는 비트 수의 차이가 있다.

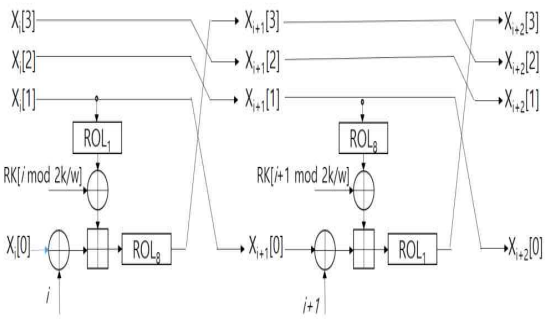


Fig. 3. Two round functions beginning with the even i-th round

CHAM 라운드 함수에는 부울 연산인 XOR 연산과 산술 연산인 덧셈 연산이 존재한다. 즉, CHAM 라운드 함수에 마스킹을 적용할 때 산술-부울 상호 마스킹 변환 과정 또는 SA 마스킹 기법을 적용하여야 한다. 산술-부울 상호 마스킹 변환 기법을 적용할 때에는 덧셈 연산 전에 부울 마스킹 되어있는 두 중간 값을 산술 마스킹으로 변환한 후 덧셈 연산을 수행한다. 그다음 덧셈 연산

을 수행한 산술 마스킹 된 중간 값은 다시 부울 마스킹으로 변환하는 과정을 거친다. 이렇게 산술-부울 상호 마스킹 변환기법을 CHAM 라운드에 적용한 것은 Fig. 4와 같으며 BtoA, AtoB로 표기되어 있는 것은 각각 부울-산술 마스킹 변환, 산술-부울 마스킹 변환을 뜻한다.

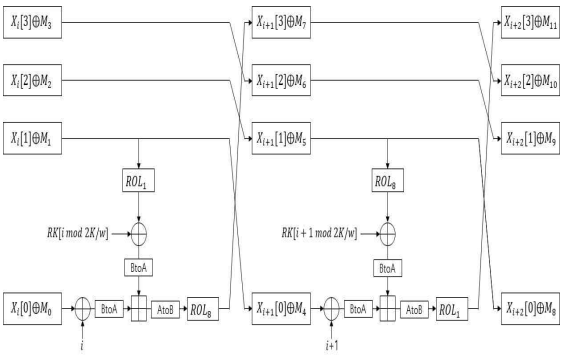


Fig. 4. CHAM round with AtoB and BtoA masking conversion method

SA 마스킹 기법은 부울 마스킹 된 중간 값을 입력으로 받아 새로운 부울 마스크 값을 생성하여 마스킹 함과 동시에 덧셈 연산을 수행하기 때문에 기존 라운드의 덧셈 연산하는 위치에 SA 마스킹 함수를 적용하면 된다. CHAM 라운드에 SA 마스킹 기법을 적용한 것은 Fig. 5과 같다.

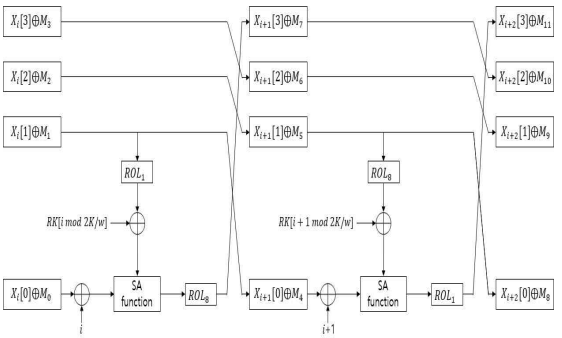


Fig. 5. CHAM round with SA masking method

V. 마스킹을 적용한 CHAM 구현 및 성능

본 논문에서는 기존에 존재하는 마스킹 기법을 CHAM 암호 알고리즘에 적용하여 32비트 Cortex-M3 프로세서 플랫폼에서 구현하였으며 컴파일러는 IAREW-ARM을 사용하였다.

마스킹이 정상적으로 설계되었는지 검증을 위해 블록 크기와 비밀 키가 128비트인 CHAM 암호 알고리즘에 마스킹을 구현하였다. 또한 IAREW-ARM 컴파일러에서 제공하는 클럭 사이클 수를 측정해 성능을 알아보았다. 다음 Table 1은 각 마스킹 알고리즘을 적용하여 구현한 CHAM-128/128의 수행 속도이다.

Table 1. Performance comparison of Masking methods applied on CHAM

구분	테이블 사전계산	계산	비율
No-Masking	-	1,408	1.00
Goubin	-	26,087	18.52
C-T	2,360	21,127	15.00
KRJ	-	26,784	19.02
V-G	7,089	28,144	19.99

CHAM-128/128에 마스킹을 적용하지 않았을 때의 연산량은 1,408번의 사이클로 다른 경량 블록 암호 알고리즘과 비교하였을 때 상당히 좋은 성능을 보인다. 하지만 마스킹 알고리즘을 적용하였을 때 5~10배 정도의 추가연산이 필요한 SPECK, LEA 등의 다른 경량 블록 암호에 비해 CHAM 암호는 15~20배로 좋지 않은 효율성을 보인다. 이는 SPECK, LEA 등과 같은 다른 경량 블록 암호의 라운드 수가 20~30정도인 것에 비해 CHAM 암호는 라운드 수가 80~96으로 많기 때문에 그에 따른 마스킹을 위한 연산이 늘어나 오버로드가 발생하는 것으로 분석된다. 마스킹을 적용하였을 때 CHAM 암호가 갖는 설계적인 문제를 고려하여 본 논문에서는 암호 라운드의 전체에 마스킹을 적용하는 것이 아닌 일부분의 라운드에만 마스킹을 적용하는 축소 마스킹을 적용해 구현하였다. 일반적으로 축소 마스킹은 암호 라운드의 처음 부분의 라운드와 마지막 부분의 라운드에 마스킹을 적용한다. 특히, CHAM 암호 라운드는 버전에 따라 8개 또는 16개의 라운드 키를 생성하여 반복사용하기 때문에 본 논문에서는 CHAM 암호 라운드의 처음과 마지막부분에서 라운드 키 전체가 한번 사용되는 라운드에 대한 축소 마스킹을 적용하였다. CHAM 암호에 대한 축소 마스킹 사양은 Table 2와 같으며 축소

마스킹을 적용하여 구현한 CHAM-128/128의 수행 속도는 Table 3과 같다.

Table 2. Reduced masking specification for CHAM

	r	생성 라운드 키	마스킹 적용 라운드 수	비율
64/128	80	16	32	40%
128/128	80	8	16	20%
128/256	96	16	32	33%

Table 3. Performance comparison of Reduced Masking methods applied on CHAM

구분	테이블 사전계산	계산	비율
No-Masking	-	1,408	1.00
Goubin	-	6,431	4.56
C-T	2,360	5,439	3.86
KRJ	-	6,566	4.66
V-G	7,089	6,838	4.85

Table 3과 같이 CHAM-128/128 암호 라운드에 대한 축소 마스킹을 적용한 경우, 전체 라운드에 마스킹을 적용했을 때 15~20배의 연산량이 추가되는 것에 비해 5배 이하의 추가 연산량을 필요로 하여 효율적으로 중간 값의 노출을 보호할 수 있다. 또한 전체 라운드에 대한 마스킹을 적용하였을 때와 축소 마스킹을 적용하였을 때 모두 21,127 사이클, 5,439 사이클로 C-T 기법에 대한 효율성이 가장 좋은 성능을 보였으나 사전 계산된 캐리 값을 저장하는 메모리 공간 256~512 바이트 정도가 필요하였다.

VI. 결 론

본 논문에서는 부채널 분석 공격에 대응하는 CHAM 경량 암호 알고리즘을 구현함에 있어 마스킹 설계 기법을 제시하고 32비트 프로세서 Cortex-M3에 구현하였다. 또한 CHAM 암호의 라운드 수가 많아 마스킹 기법을 적용하였을 때 다른 암호에 비해 많은 추가 연산을 필요로 하는 것을 고려하여 축소 마스킹을 적용하였으며 총 라운드의 20~40%의 라운드에 마스킹을 적용하였다. CHAM-128/128를 구현한 결과 20%만의 라운드에 마스킹을 적용하였을 때 전체 라운드에 마스킹을 적용했을 때와 비교하여 약 4배정도 추가 연산량이 감소하는 것을 확인하였으며, 여러 마스킹 알고리즘 중 C-T 마스킹 기법을 적용하였을 때 가장 효율적으로 중간 값을 보호할 수 있다는 것 또한 확인하였다.

참고문헌

〔1〕 P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO'99, LNCS 1666, pp. 388-397, 1999

〔2〕 L. Goubin, "A sound method for switching between Boolean and arithmetic masking," CHES'01, LNCS 2162, pp. 3-15, 2001.

〔3〕 J. Coron and A. Tchulkine, "A new algorithm for switching from arithmetic to Boolean Masking," CHES'03, LNCS 2779, pp. 89-97, 2003

〔4〕 M. Karroumi, B. Richard, and M. Joye, "Addition with blinded operands," Constructive Side-Channel Analysis and Secure Design, COSADE'14, LNCS 8622, pp. 41-55, 2014.

〔5〕 P. Vadnala and J. Groβschadl, "Faster mask conversion with lookup tables," Constructive Side-Channel Analysis and Secure Design, COSADE'15, LNCS 9064, pp. 207-221, 2015.

〔6〕 Bonwook Koo, Dongyoung Roh, Hyeonjin Kim, Younghoon Jung, Dong-Geon Lee, and Daesung Kwon, "CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices," International Conference on Information Security and Cryptology, ICISC'17, 2017

딥러닝을 활용한 부채널 이차 차분분석 성능 향상에 관한 연구

김 태 성*, 최 두 호*

요 약

부채널 분석의 마스킹 기법은 마스크를 랜덤하게 생성해 중간값을 변경시킴으로써 부채널 공격자가 중간값을 예측하지 못하게 하고 전력 소모량과 같은 부채널 정보와의 상관도를 낮춰 방어하는 기법이다. 이차 차분분석은 두 개의 중간값을 합쳐서 마스크를 제거하여 마스킹 기법을 무력화하는 공격 방법이다. 이차 차분분석에서 전처리는 전력소모량 두 지점을 절대차 또는 정규곱을 통해 하나로 합치는데 이는 가상 연산인 xor를 정확히 표현하지 못해 낮은 상관도를 가지고 되고, 결과적으로 이차차분 분석의 성능을 낮추는 요소로 작용한다. 본 논문은 딥러닝을 활용해 디바이스 전력 소모의 특성을 학습하고 이를 통해 상관도를 높이는 방법을 제안하고자 한다. 기존 정규곱 전처리에서 상관도의 최대치는 0.35이었으나, 본 논문의 딥러닝 기법을 적용하면 0.46까지 높일 수 있음을 실험을 통하여 보여준다.

Ⅰ. 서 론

부채널 분석 기술은 디바이스에 구현된 암호 알고리즘의 동작 과정에서 발생하는 부채널적 정보, 예를 들어 전력소모, 전자파, 동작속도 등을 사용해서 비밀키를 누출 시키는 공격 기법이다[1].

암호 알고리즘의 수학적 안전성이 확인되었다 하더라도 키 비트에 따라 전력소모량이나 전자파가 달라진다면 숨기고자 하는 비밀키가 외부로 누출되는 안전성에 문제가 생길 수 있다. 부채널 정보는 디바이스의 특성이나 암호 알고리즘의 구현에 의존적이다. 이러한 부채널 공격을 방어하고자 하는 노력도 계속되어 왔다. 소프트웨어적인 방어에는 마스킹 기법이 많이 활용되고 있다. 부채널 공격자는 가능한 모든 서브키 바이트에 대하여 암호 함수 내의 중간값을 예측한다. 그리고 예측된 중간값과 수집된 부채널 정보와의 상관성을 통계적 기법으로 조사해서 가장 높은 상관도를 가지는 서브키를 비밀키의 일부로 보는 것이다. 마스킹은 암호 함수의 중간값을 공격자가 예측할 수 없는 값으로 변경해서 상관성을 찾지 못하도록 하는 방어 방법이다. 마스크라는 랜덤값을 매 암호 수행마다 생성해서 중간값을 공격자가 전혀 예측할 수 없는 값으로 바꾸고 공격자의 상관도 추측을 무산시키는 것이다.

마스킹에 하나의 마스크를 사용하는 일차 마스크는 이차차분분석에 의해 무력화된다. 이때 공격자는 마스크를 제거하기 위해 암호 수행의 두 지점을 합쳐서 마스크를 제거하고 다시 상관도를 측정한다[4].

마스킹 기법을 높은 차수를 쓸수록 암호 구현이 복잡하고 무거워지며 수행 속도가 매우 느려지므로 현실적으로는 일차 마스킹 기법과 다른 방어 기법을 복합적으로 사용하고 있는 실정이다.

이차 차분 분석은 일차 마스킹을 공격해 낼 수 있는 방법이지만 수행 속도가 매우 느리다. 그 이유는 첫째, 서브키를 동시에 두 개를 추측해야 하므로 찾아야 하

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00230, (IoT 총괄+1세부) IoT 디바이스 자율 신뢰보장 기술 및 글로벌 표준기반IoT 통합보안 오픈 플랫폼 기술개발 [TrusThingz 프로젝트])

* 한국전자통신연구원 지능보안연구그룹 (taesung, dhchoi@etri.re.kr)

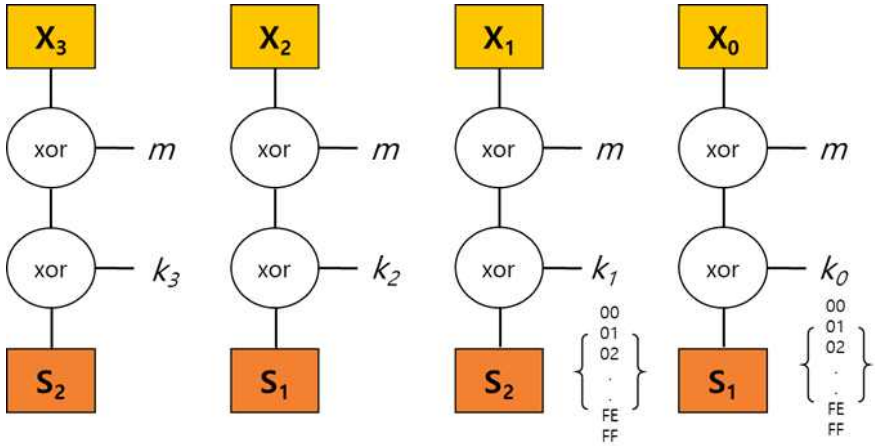
는 비트의 길이가 두 배로 길다. 둘째, 실제 암호 수행에서는 없는 연산을 가상으로 상정해 만들어 연관성을 찾음으로 인해 상관도가 매우 낮게 나오게 된다.

본 논문에서는 이차 차분분석에서 매우 낮게 나오는 상관도를 높이는 방법을 딥러닝을 활용해 해결하려고 한다. 이차 차분분석에서 상관도를 찾기 전에 전처리 과정을 거치는데 예측하는 암호 함수의 두 개의 중간값을 하나로 합치고, 전력소모 두 개의 값을 하나로 합치는 것이다. 기존의 이차분석기법에서는 두 개의 중간값을 XOR하고 이것의 해밍웨이트를 하는 전처리를 사용한다. 본 논문은 딥러닝을 활용해 디바이스의 파형 특성을 학습하고 중간값을 하나로 합치는 전처리를 학습된 딥러닝으로 대체하여 상관도를 높인다.

본 논문에서는 부채널 이차 차분분석에 대하여 좀 더 자세히 설명하고(Ⅱ장), 제 Ⅲ장에서는 이차차분분석의 전처리 과정을 딥러닝을 활용해 상관도를 향상시키는 방법은 제안한다. 제 Ⅳ장에서는 전력 소모량을 시뮬레이션하는 파형 생성과 딥러닝의 구현 및 실험 결과를 설명한다. 마지막으로 제 Ⅴ장에서 실제 부채널 분석의 활용 가능성과 다른 부분에서의 딥러닝 활용 가능성에 대해 논의하고 본 논문의 결론을 맺고자 한다.

Ⅱ. 부채널 이차 차분 분석

가장 흔히 사용되는 부채널 정보는 전력소모량과 전자파이다. 두 종류의 부채널 정보는 수집시에만 다를 뿐 비밀키를 누출하는 부채널 분석에서는 같은 방법을 사용하므로 본 논문에서는 전력소모로 통일하도록 하겠다. 부채널 공격자는 암호의 평문 또는 암호문은 획득 가능하다고 가정한다. 공격의 순서는 다음과 같다. 모든 가능한 서브키에 대하여 입력된 평문으로 암호 함수를 적용하고 공격하고자 하는 중간값을 계산한다. 이렇게 계산된 중간값과 수집된 전력 소모량과 상관도를 조사한다. 평문을 계속 바꿔가면서 중간값과 상관도를 조사한다. 올바른 서브키는 중간값 예측이 항상 맞을것이고 이에 따라 높은 상관도를 가지게 될 것이다. 이러한 과정을 모든 서브키에 대해 수행하면 전체 마스터키까지 누출 시킬 수 있다.



〔그림 1〕 마스킹 기법과 이차 차분 분석 방법

부채널 공격에 대한 소프트웨어적인 방어 기법은 마스킹이 가장 많이 사용된다. 마스킹은 암호 함수의 중간값을 공격자가 예측 불가능하게 만들어 상관도를 낮추려고 한다. 마스크라는 랜덥값을 매 암호 수행전에 생성해 평문과 서브키를 마스킹하고 중간값을 공격자가 예측할 수 없는 값으로 바꿔 놓는다.

마스킹 기법은 이차 차분분석에 의해 공격 할 수 있다. 이차 차분 분석의 핵심은 두 개의 중간값을 합쳐서 마스크를 제거하고 합친값을 예측 가능하도록 변경하는 것이다. 〔그림 1〕은 일차 마스킹 기법과 이차 차분 분석의 원리를 설명하고 있다.

$$MS[x_0 \oplus m] = S[x_0] \oplus m'$$
$$MS[x_1 \oplus m] = S[x_1] \oplus m'$$

$$MS[x_0 \oplus m] \oplus MS[x_1 \oplus m]$$
$$= S[x_0] \oplus m' \oplus S[x_1] \oplus m'$$
$$= S[x_0] \oplus S[x_1]$$

위에서 보여주는 수식처럼 마스크된 두 개의 SBOX 출력을 xor 하게 되면 마스크는 제거가 되고, 공격자가 예측 가능한 원래 SBOX의 출력의 xor로 변경이 된다. 이제 중간값을 예측했으므로 전력소모량에서 두 중간값의 위치에 해당하는 두 개의 값을 하나로 합쳐야 한다. 이렇게 합치는 과정을 이차 차분분석에서는 전처리 과정이라고 한다.

$$|t_v - t_u|,$$
$$(t_v - E(t_{vi}))*(t_u - E(t_{ui})),$$

위 의 두 수식은 파형 전처리에서 많이 사용되는 방법으로 절대차(absolute difference)나 정규곱(normalized product)라 부른다[5,6]. 중간값의 xor는 실제 암호 수행에 존재하지 않는 연산이므로 전처리는 전력소모량에서 이를 가상으로 만들어 본 것이다. 따라서 xor된 중간값과 전처리된 전력 소모량은 매우 낮은 상관도를 가지게 된다. 절대차의 상관도는 0.25, 정규곱의 상관도는 0.35가 최대치로 알려져 있다. 낮은 상관도는 이차차분 분석의 효율을 떨어뜨리는 중요한 요소로 작용된다.

Ⅲ. 이차 차분분석 딥러닝 전처리

이차 차분분석에서 두 개의 중간값은 xor를 통해 하나로 합쳐진다. 본 논문에서는 공격 대상 디바이스의 특성을 학습해 중간값 두 개를 합치는 방법을 딥러닝으로 대체하고 상관도를 높이고자 한다. 먼저 공격자는 해당 디바이스와 동일한 장비를 가지고 알고 있는 비밀키로 마스킹된 암호 알고리즘을 수행한다[2,3]. 얻어진 파형으로 두 개의 중간값과 정규곱 전력 소모량으로 딥러닝을 수행한다. 공격시에는 학습된 딥러닝을 활용해 두 개의 중간값을 입력해 정규곱 값을 예측한다. 이제 상관도 조사는 딥러닝 정규곱과 전처리 정규곱에 대하여 수행한다.

기존 전처리 기법의 절대차와 정규곱이 낮은 상관도를 가지는 것을 실험으로 파악하였다. [그림 2]와 [그림 3]은 조사한 상관도를 나타낸다. 파란색 라인은 올바른 키, 붉은 점선은 틀린 키를 나타낸다.

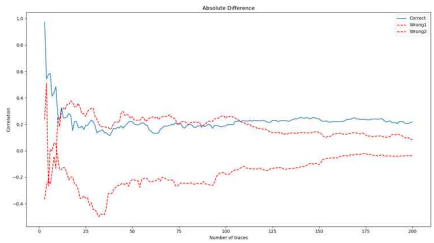
Ⅳ. 구현 및 실험

실제 디바이스에서 실험하기 전에 먼저 전력소모량을 시뮬레이션하는 파형을 생성해 본 논문의 아이디어를 검증하였다. 시뮬레이션 파형은 다음 수식에 의해 생성하였다. $y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + b$, y는 전력소모량, x는 중간값의 각 비트를 표시한다. w는 가중치 b는 편향이다. w는 아래 값으로 결정하였다. msb가 전력소모량에 가장 많은 영향을 미치는 설정으로 통상 저전력 장비가 이러한 특성을 보인다.

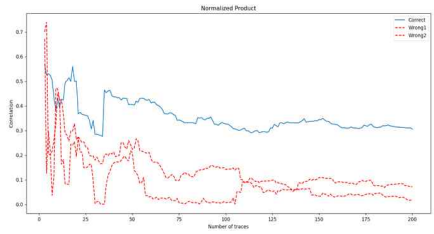
$$W = [0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]$$

마스킹 기법을 구현하고 딥러닝에 사용한 전력소모 파형을 생성하였다. 딥러닝의 입력 데이터는 두 개의 중간값의 비트와 xor된 중간값의 비트로 총 24개의 요소로 만들었다. 딥러닝의 출력은 전력 소모량의 정규곱으로 하였다. 딥러닝의 라이브러리는 텐서플로우를 사용하였으며, 2개의 히든레이어와 각 히든레이어에 50개의 뉴런을 사용하였다. 활성 함수는 Relu를 사용하였고 Aadm optimizer를 적용하였으며, 드랍아웃은 사용하지 않았다.

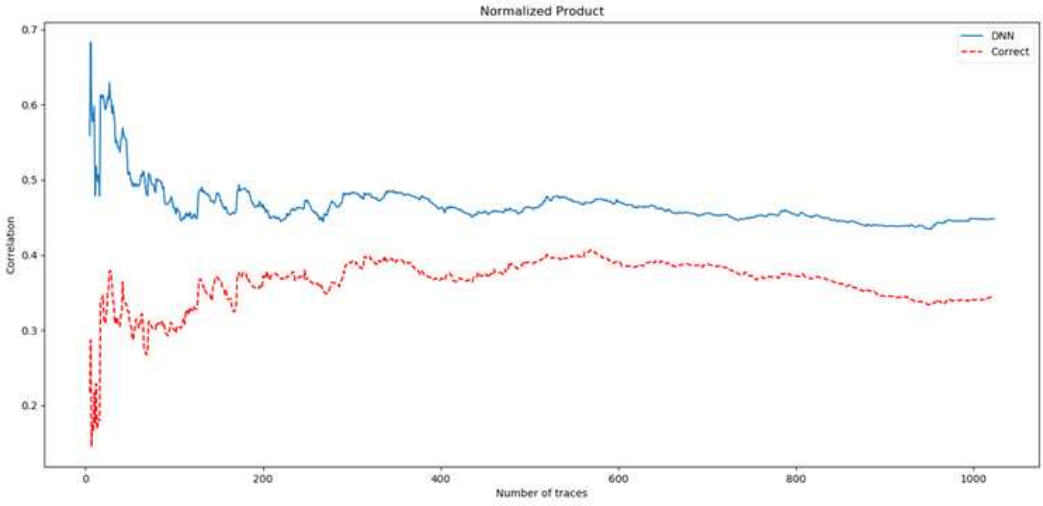
[그림 4]는 딥러닝 결과를 활용해 두 개의 중간값을 하나로 합쳐서 사용했을 경우에 이차 차분분석의 결과를 보여준다. 상관도가 0.46까지 높아지는 것을 볼 수 있다.



[그림 2] 절대차에 대한 상관도 조사



[그림 3] 절대차에 대한 상관도 조사



〔그림 4〕 딥러닝을 활용해 두 개의 중간값을 합쳤을 때 높은 상관도(0.46)

V. 결 론

부채널 이차차분 분석에서 전처리는 가상 연산(xor)을 절대차 또는 정규곱으로 표현한다. xor를 정확히 표현할 방법이 없으므로 이는 낮은 상관도를 가지게 된다. 낮은 상관도는 틀린 키와의 구별을 확실히 해주지 못하므로 이차 차분분석은 성능이 매우 떨어진다. 본 논문은 이러한 문제점을 해결하고자 딥러닝을 활용해 두 개의 중간값을 최대한 정규곱과 일치시키도록 하였다. 이 결과 상관도가 0.35에서 0.46까지 증가하였고 이차차분 분석의 성능을 높일 것으로 예상한다.

향후 연구 과제로는 실제 디바이스에서 구현과 실험을 수행해 높은 상관도가 이차차분분석의 성능에 미치는 영향을 파악하는 것이다.

참 고 문 헌

〔1〕 Kocher, Paul, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1999.

〔2〕 Oswald, Elisabeth, and Stefan Mangard. "Template attacks on masking-resistance is futile." Cryptographers' Track at the RSA Conference. Springer, Berlin, Heidelberg, 2007.

〔3〕 Chari, Suresh, Josyula R. Rao, and Pankaj Rohatgi. "Template attacks." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2002.

〔4〕 Waddle, Jason, and David Wagner. "Towards efficient second-order power analysis." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2004.

〔5〕 Prouff, Emmanuel, Matthieu Rivain, and Régis Bevan. "Statistical analysis of second order differential power analysis." IEEE Transactions on computers 6 (2009): 799-811.

〔6〕 Standaert, François-Xavier, et al. "The world is not enough: Another look on second-order DPA." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2010.

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

포스터발표 트랙 E

기계학습 기반의 작성자 분석 기술을 적용한 악성코드 분류 시스템

신건윤, 한명묵*

가천대학교 컴퓨터공학

Malware Classification System using Authorship Analysis based on Machine Learning

Gun-Yoon Shin, Myung-Mook Han*

Computer Engineering, Gachon University.

요 약

최근 해킹 기술과 공격 기법에 발전으로 인하여 대규모 피해를 발생시키는 공격이 지속적으로 발생하고 있으며, 공격 대상도 개인이 아닌 기업이나 국가를 목표로 하는 해킹공격이 늘어나고 있다. 이러한 사이버 공격은 악성코드에 의한 공격으로 행해지는 경우가 많으며 이처럼 악성코드는 사용자의 컴퓨터를 감염시켜 중요 정보 및 데이터 탈취, 시스템 마비 등의 행위를 수행한다. 따라서 본 논문에서는 작성자 분석과 기계 학습을 적용하여 악성코드를 분류하는 기술을 제안하며, 수집한 악성코드에서 어셈블리 코드와 문자열 코드를 추출하고 코드 안에 들어있는 opcode, 문자열, 실행파일 정보 등과 같은 다양한 데이터를 통해 특징 추출 및 선택을 수행하고 이를 통해 구축된 데이터 셋을 가지고 악성코드 분류를 수행하며 분류 시 다양한 기계학습 알고리즘을 적용하여 비교 분석 연구를 진행한다. 작성자 분석 이론과 기계 학습을 적용한 실험을 수행한 결과 91%의 악성코드 분류 정확도를 보였다.

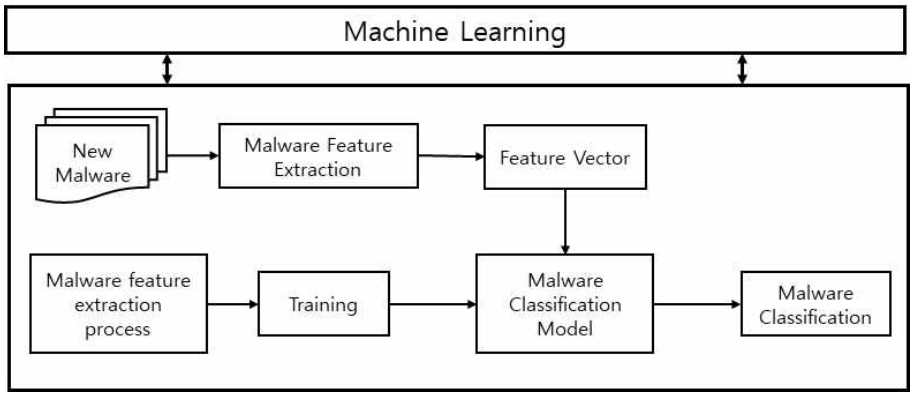
I. 서론

인터넷이 빠르게 발달함에 따라 산업, 기업, 국가 등에 인터넷이 보급되고 이를 활용한 다양한 기술들이 빠르게 증가하고 있다. 하지만 이러한 순기능과는 반대로 인터넷 보안을 위협하는 공격들도 빠르게 증가하고 있다. 대표적인 예로 지능형 지속 위협(Advanced Persistent Threat : APT) 공격을 들 수 있으며, APT 공격은 취약점을 통해 악성코드를 침입시키는 방식으로써, 침입을 통해 해당 산업 또는 기업의 인터넷 망을 망가트리거나 중요한 데이터를 훔쳐가기도 한다. 이러한 공격들을 정확히 탐지하고 식별하기 위해서는 주로 사용되는 악성코드 분석, 분류, 차단 및 탐지하는 시스템 구축이 필요하다. 악성코드 분석 및 분류에 사용되는 특징들은 대표적으로 N-gram, PE, Opcode, 패킹, 파일 구조, 함수, 레지스터, system call 등이 있으며, 작성자 분석을 통해서는 idiom, graphlet, super graphlet, call graphlet, N-gram, 단어 빈도, 어휘 표현력 등의 특징을 추출 할 수 있다. 작성자 분석에서 일반적으로 다루는 소스 코드, 어셈블리 코드 등은 악성코드 분석에서도 자주 다루는 부분으로 악성코드에 작성자 분석을 적용하면 기존과는 다른 정보를 파악할 수 있다[1].

따라서 본 논문에서는 악성코드 분류를 수행하기 위해서 작성자 분석 방법을 사용하여 악성코드를 분석하고 기계 학습을 통해 악성코드 분류를 수행하는 프레임워크를 제안한다. 악성코드 분석에 사용되는 일반적인 특징과 작성자 분석 기법을 적용한 특징을 추출하여 보다 다양한 특징을 추출하고 이를 통해 악성코드 분류에 적합한 특징들을 찾고, 분류 정확도를 향상시킬 것이다.

*Corresponding Author: mmhan@gachon.ac.kr

이 논문은 2018년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07050864)



[그림 1] 제안하는 악성코드 분류 시스템

II. 관련연구

2.1 악성코드 분류

악성코드를 분류하기 위해서는 우선 악성코드 분석을 수행하여 악성코드에 들어있는 정보를 추출해야한다. 이처럼 악성코드를 분석하는 방법에는 대표적으로 정적분석과 동적분석이 있으며, 정적분석은 디버거와 역공학을 이용하여 악성코드를 실행시키지 않은 상태에서 코드를 분석하는 방법을 말하며 동적분석은 가상머신 환경 내에서 악성코드를 직접 실행시켜 프로세스 생성, 레지스트 변경 등 악성코드의 행위를 분석하는 방법을 말한다. 정적분석에서는 opcode, PE구조, 파일 유형, 패킹 정보, 문자열 등의 정보를 수집할 수 있으며 동적분석에서는 프로세스, 레지스트 정보 등과 같은 행위와 관련된 정보를 수집할 수 있다[2].

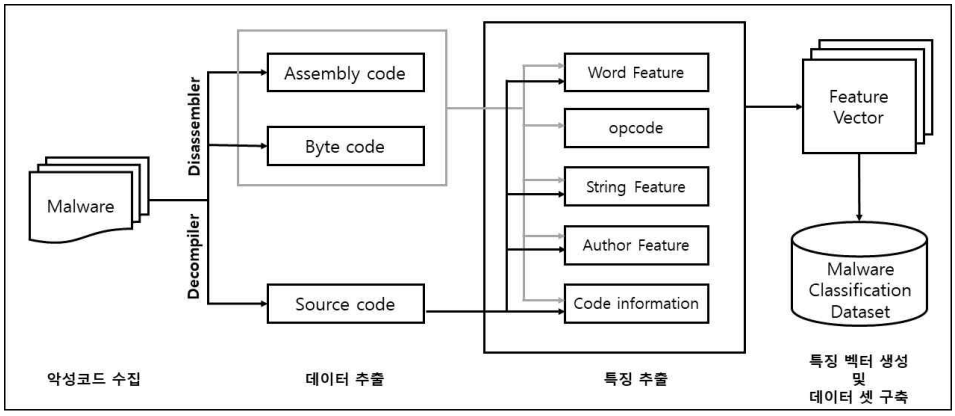
이와 관련된 연구로는 Y. Zhong[3]의 논문에서는 악성코드 분류를 수행하기 위하여 function structure 분석을 수행하였으며, API, 문자열, 블록 등을 통해 분류를 수행하였다. S. Nari[4]에서는 악성코드의 행위 분석을 통해 악성코드 그룹 분류를 수행하였으며, IP, port, 주소 등의 정보를 사용하였다. 또한 기계 학습을 적용하여 악성코드 분류를 수행한 연구들도 있는데 X. Hu[5]에서는 정적분석을 통해 특징을 추출하고 클러스터 앙상블 기법을 적용하여 악성코드 분류를 수행하였으며, R. Konrad[6]에서는 동적분석을 통해 추출한 특징에 군집화 알고리즘을 적용하여 악성코드 분류를 수행하였다.

2.2 작성자 분석

작성자 분석은 바이너리 코드, 어셈블리어 코드 등이나, 이메일과 같은 문서를 분석하여 작성자의 행위, 패턴, 특징 등을 찾아 식별하는 방법을 의미하며, 주로 분석을 통해 작성자의 프로필을 구축하여 식별한다. 작성자 분석은 다양한 특징을 가지고 있고 해당 특징들은 단순히 작성자를 구분하는 데만 쓰이는 것이 아닌 악성코드 분류에도 사용할 수 있다. 작성자를 구분하기위해 코드를 분석하는 것과 악성코드 분류를 위해 악성코드를 분석하는 것이 특정 코드의 동일한 부분을 분석하기 때문이다[7][8].

III. 악성코드 분류 시스템

본 논문에서는 [그림 1]과 같이 악성코드 분류를 수행하기 위하여 악성코드로부터 이와 관련된 데이터를 수집하고 특징 추출 및 선택을 통해 악성코드 분류를 위한 데이터 셋을 구축하고 여러 분류 알고리즘을 해당 데이터 셋에 적용하여 악성코드 분류를 하고, 기계 학습을 적용하여 해당 프로세스의 자동화를 수행한다. [그림 2]는 악성코드 분류를 위한 특징 추출 프로세스를 보여주는 그림으로, 먼저 악성코드를 수집하고 디컴파일러 및 디스어셈블러를 통해 어셈블리 코드, 바이트, 소스 코드 등과 같은 데이터를 추출한다. 특징을 추출할 때 작성자 분석 이론을 적용하여 특징을 추출하며 해당 방식은 악성코드 분석을 수행 시 보다 단순히 악성코드 식별을 위해 분석하는 것보다 더 많은 데이터를 분석하고 처리하기 때문에 다양한 특징 추출이 가능하고 이를 통해 보다 정확한



[그림 2] 악성코드 특징 추출 및 데이터 구축 프로세스

악성코드 분류가 가능하게 된다.

또한 각각의 데이터 마다 가지고 있는 정보가 다르기 때문에 이를 고려하여 특징을 추출해야 한다. 예를 들어, 어셈블리 코드에는 opcode, PE구조, 작성자 분석 및 문자적 정보 등이 들어있고 바이트 코드에는 코드, 문자적 정보 등이 들어있기 때문에 각각의 데이터에서 추출할 수 있는 특징을 선정한다. 특히 작성자 분석 또는 코드 정보 특징 등과 같은 경우에 데이터 추출을 통해 나온 모든 코드들에서 나오는 경향이 있기 때문에 이를 고려하여 데이터를 분석하고 특징을 추출해야한다. 특징 추출 후, 해당 특징들을 가지고 특징 벡터를 생성하고 이를 가지고 악성코드 분류를 위한 데이터 셋을 구축한다.

악성코드 분류 시, SVM, DT, RF 등과 같은 여러 분류 알고리즘을 가지고 수행하며 이를 통해 악성코드 분류가 정확하게 이루어지는 지와 결과 비교 분석을 통해 악성코드 분류에 적합한 알고리즘을 찾는 연구를 수행한다.

IV. 실험

본 논문에서는 악성코드에서 수집할 수 있는 다양한 코드들 중에 어셈블리 코드를 사용하여 악성코드 분류를 수행한다. 어셈블리 코드를 통해 opcode, 코드 정보, idioms 특징을 추출하고 이를 가지고 데이터 셋을 구축한다. opcode에서는 N-gram을 수행하고 코드 분석을 통해 코드에 들어있는 정보들(포멧, os 유형, 세그먼트 유형 등)을 얻으며 idioms를 통해 코드 명령어 시퀀스 분석을 수행한다. 데이터 셋을 구축하고 분류 알고리즘를 통해 악성코드 분류를 수행한다. 분류 성능을 측정하기 위해서 F1-measure와 분류 정확도를 사용하였다.

4.1 데이터 셋

사용하는 데이터 셋은 kaggle에 있는 Microsoft Malware Classification Challenge (BIG 2015)을 사용하였으며 해당 데이터 셋에 들어있는 4가지 종류의 악성코드와 어셈블리 코드를 가지고 실험을 진행하였다.

4.2 실험 결과

본 논문에서는 작성자 분석 기법을 적용한 악성코드 분류 정확도를 확인하기 위해 악성코드 분석에 주로 사용하는 특징들과 작성자 분석 기법을 적용하여 추출한 특징들을 가지고 분류 정확도 비교 분석을 수행한다.

[표 1]은 악성코드 분석에 주로 사용하는 특징들만을 가지고 악성코드 분류 실험을 수행한 결과이다. 결과값에 나와있는 것처럼 최대 85%의 분류 정확도를 보였으며, 분류 정확도가 가장 높은 알고리즘은 SVM과 RF로 나타났다. [표 2]는 작성자 분석 이론을 적용한 특징을 적용하여 악성코드 분류를 수행한 실험으로 가장 높은 분류 정확도는 KNN 적용시 91%의 정확도를 보였고 다른 알고리즘들의 정확도 작성자 분석 이론을 적용하기 전보다 향상된 것을 확인할 수 있었다. 이를 통해 작성자 분석기반 특징을 적용하는 것이 좀 더 향상된 악성코드 분류 정확도를 얻을 수 있다는 것을 확인하였으며, 본 연구에서는 작성자 분석기반 특징 중 일부분만을 추출하였기 때문에 지속적인 연구를

통해 다양한 특징을 추출하고 이를 적용하게 된다면 보다 향상된 분류 정확도를 얻을 수 있다는 것을 확인 할 수 있었다.

[표 1] 악성코드 분류 실험결과

	accuracy	precision	recall	F1-measure
SVM	0.85	0.71	0.84	0.77
KNN	0.84	0.70	0.83	0.76
DT	0.84	0.70	0.83	0.76
RF	0.85	0.71	0.84	0.77

[표 2] 작성자 분석을 적용한 악성코드 분류 실험결과

	accuracy	precision	recall	F1-measure
SVM	0.90	0.80	0.89	0.85
KNN	0.91	0.91	0.91	0.90
DT	0.89	0.79	0.88	0.83
RF	0.89	0.79	0.88	0.83

V. 결론

본 논문에서는 악성코드를 분류하기 위하여 작성자 분석 기법과 기계학습을 적용한 프레임워크를 제안하였다. 작성자 분석 기법을 적용하여 악성코드에서 추출 할 수 있는 특징들의 범위를 확장하였으며, 이를 통해 악성코드 분류에 특화된 특징들을 선정할 수 있으며 이는 분류 정확도 향상의 결과를 가져온다는 것을 확인할 수 있었다. 향후 연구를 진행하는데 있어 악성코드 그룹의 수를 늘려 보다 다양한 악성코드 그룹을 분류할 수 있는 연구를 수행하며, 작성자 분석을 적용한 다양한 특징을 추출하여 악성코드 분류에 보다 적합한 특징들을 찾고 이를 통해 악성코드 분류의 정확성을 높이는 연구를 수행할 것이다. 또한 악성코드 분류 프레임워크에 기계 학습을 적용함으로써 악성코드 분석에 대한 전문적인 지식이 없어도 악성코드 분석을 가능하게 하는 프레임워크 연구 및 개발을 수행 할 것이다. 더 나아가, 악성코드 분류를 통해 얻은 특징들과 추가로 악성코드 보유 이메일, 네트워크 분석 등을 연구하여 궁극적으로 악성코드 공격자를 식별하는 프레임워크를 제안하고 연구할 것이다.

[참고문헌]

[1] Y. Zhong, "A Malware Classification Nethod based on Similarity of Function Structure," IEEE/IPSJ, pp. 256-261, 2012.

[2] C. Wang, "A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm," Fuzzy systems and knowledge discovery, pp. 233-245, 2016.

[3] Y. Zhong, H. Yamaki, and H. Takakura, "A Malware Classification method based on Similarity of Function Structure", International Symposium on Applications and the Internet, pp. 256-261, 2012.

[4] S. Nari and A. Ghorbani, "Automated Malware Classification based on Network Behavior", International Conference on Computing, Networking and Communications, pp. 642-647, 2013.

[5] X. Hu and K. G. Shin, "DUET : Intergration of Dynamic and Static Analyses for malware clustering with Cluster Ensembles", Proceedings of the 29th Annual Computer Security Applications Conference, pp. 79-88, 2013.

[6] R. Konrad, T. Philipp, W. Carsten, and H. Thorsten, "Automatic Analysis of malware Behavior using Machine Learning", Journal of Computer Security, vol. 19, no. 4, pp. 639-668, 2011.

[7] E. Stamatatos. "A survey modern authorship attribution methods." ASIS&T, Vol.60, pp. 538-556, 2009.

[8] Alrabaee, Saed, et al. "On the feasibility of malware authorship attribution." International Symposium on Foundations and Practice of Security, pp. 256-272, 2016.

HDMI-CEC 프로토콜의 취약점 점검 기법에 대한 연구

김양우*, 문창현*, 정혜진*, 조수연*, 조혜원*, 양산휘**, 신정훈***, 김홍진****
*KITRI Best of the Best 7th
**세종대학교
***THEORI
****LG CNS

A Study on Method for Vulnerability Assessment of HDMI-CEC Protocol

Yang-U Kim*, Changhyeon Moon*, Hyejin Jeong*, Sooyeon Jo*, Hyewon Jo*, Sanhwi Yang**, Jeonghoon Shin***, Hongjin Kim****
*KITRI Best of the Best 7th
**Sejong University
***THEORI
****LG CNS

요 약

최근 HDMI를 필요로 하는 AV 전자제품이 증가하고 있다. HDMI는 영상 정보 전송뿐만 아니라 모니터나 TV와 같은 디스플레이 기기의 전원을 제어하거나 디스플레이 볼륨 조절 등의 기능을 위해 CEC 프로토콜이 존재한다. 본 논문에서는 HDMI CEC 프로토콜을 이용한 공격을 사전에 예방하기 위해 다양한 취약점 점검 기법을 알아본다. 그리고 CEC 프로토콜 분석 후 그에 따른 점검 기법을 설계 및 구현하고 취약점 점검 결과에 대해 기술한다.

I. 서론

HDMI(High Definition Multimedia Interface)는 고선명 멀티미디어 인터페이스로, 고화질 영상을 지원하는 다양한 AV 기기들의 HDMI 사용이 증가하였다[1]. HDMI에는 서로 연결된 두 기기 간에 영상 송•수신뿐만 아니라 소스 기기[1)]와 싱크 기기[2)]가 서로 연결된 상태에서 전원 on/off, 디스플레이 볼륨 조절 등을 할 수 있는 기능이 존재한다. 이를 지원하기 위해서 HDMI CEC(Consumer Electronics Control) 프로토콜이 사용된다.

CEC 프로토콜은 AV.link 기반으로 통신하며, 단선 양방향 통신 프로토콜이다. CEC는 전기적 사양, 신호 및 비트 타이밍, 프레임과 자세한 기능 분석 및 개별 메시지 전송 등에 활용이 된다.

HDMI를 지원하는 기기가 증가하는 만큼 이를 안전하게 사용하기 위해서 보안은 중요하다. 따라서 본 논문에서는 해당 프로토콜에 대한 취약점을 점검하기 위해 연구를 진행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 취약점 점검 기법과 관련된 연구를 소개하고 3장에서는 본 논문에서 이용한 취약점 점검 기법에 대해 기술한다. 끝으로 4장에서 결론을 맺는다.

II. 관련 연구

2.1 소스 코드 오디팅(Source Code Auditing)

1) 소스 기기(Source Device) : A Device with an HDMI Output
2) 싱크 기기(Sink Device) : A Device with an HDMI Input

소스 코드 오디팅은 실제 프로그램 소스 코드 내에서 버그나 프로그래밍 규칙 등을 위배하는 취약점을 발견하는 것을 목적으로 소스 코드를 전체적으로 분석하는 것이다. 그러나, 프로그램에 따라 소스 코드를 제공하지 않는 기기에서는 해당 기법을 적용하기 어렵다.

2.2 퍼징(Fuzzing)

퍼징은 대상 프로그램에 잘못된 값이나 임의 값을 입력하여 예상하지 못한 동작을 수행하도록 하고 보안과 관련된 결함 또는 DoS(Denial of Service), 서비스 저하 또는 의도하지 않은 행위로 이어지는 심각한 결함을 찾는 것이다[2].

퍼징 데이터를 생성하는 방법은 세 가지가 있다. 첫 번째, 대상 프로그램의 세부 정보를 가지지 않고 올바른 데이터를 수정하여 무작위로 생성하는 블랙박스 방식이 있다. 두 번째, 대상 프로그램의 소스 코드와 동작에 대한 지식을 가지고 있을 경우에 진행하는 화이트박스 방식이 있다. 마지막으로, 위에서 설명한 두 가지 방식의 장점을 모두 가지는 그레이박스 방식은 동작 대상에 대한 최소한의 지식만 사용한다[3].

2.3 리버스 엔지니어링(Reverse Engineering)

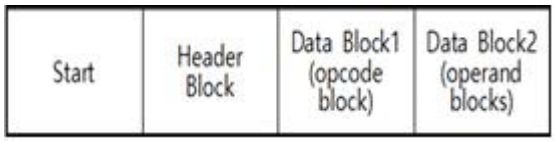
리버스 엔지니어링은 바이너리 분석이라고도 하며 소스 코드를 필요로 하지 않고 바이너리를 가지고 어셈블리를 분석하여 취약점을 점검하는 기술이다. 리버스 엔지니어링을 위해서는 높은 수준의 전문 지식이 필요하다[4].

취약점 점검 기법에는 크게 위에서 설명한 3가지 방법이 있다. 본 논문에서는 소스 코드나 펌웨어를 얻기 힘든 임베디드 기기 특성상 범용적으로 취약점을 분석하기 위해 퍼징 기법을 선택하였다. 따라서, HDMI CEC 프로토콜을 지원하는 임베디드 기기를 위한 퍼저(Fuzzer)를 설계하고 제작한다.

III. 취약점 점검 기법

3.1 CEC 프로토콜

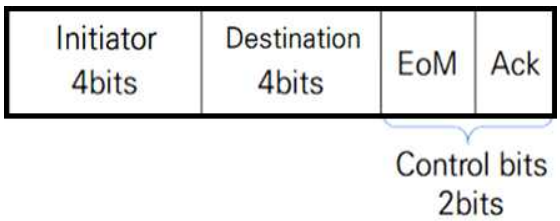
본 논문의 CEC 메시지 프레임은 [그림 1]의 형식을 가진다[5]. CEC 메시지는 start bit, header block, data block1, data block 2의 구조를 가지며 최대 크기는 16 block이다.



[그림 1] CEC Message Frame

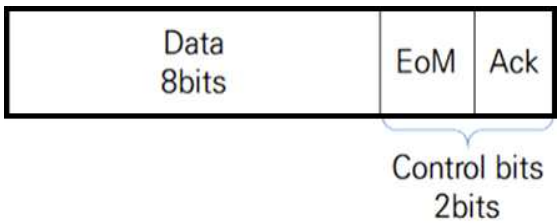
Start bit는 1 bit의 크기를 가지며 CEC 메시지의 시작을 나타낸다. 다음 header block부터 data block 1, data block 2는 모두 10 bits의 값을 가지며 하위 2 bits는 control bits로 EoM(End of Message) bit와 ACK (Acknowledge) bit가 각각 1 bit 씩 들어간다.

Header block은 [그림 2]와 같이 Initiator(이하 'source'로 칭한다.), Destination(이하 'dest'로 칭한다.)과 control bits로 구성되어있다. Source와 dest에는 각각 4 bits에 해당하는 logical 주솟값이 들어간다. TV에서 Tuner 기기로 CEC 메시지를 전송하는 경우 Source에는 TV(Root)의 logical 주소에 해당하는 0x00 값이 들어가며 dest에는 Tuner 기기의 logical 주소에 해당하는 0x03 값이 들어가게 된다.



[그림 2] Header Block

Data block 1과 Data block 2는 [그림 3]과 같은 구성을 가지며 상위 8 bits는 각각 opcode와 operand이다.



[그림 3] Data Block

EoM bit는 마지막 block 유무를 알 수 있는 값이다. 0의 값이 오면 다음에 block이 온다는 의미이며, 1의 값이 오면 현재 block이 마지막이라는 의미이다.

ACK bit는 소스 기기에 의해 항상 1로 설정된다. 0의 값이 오면 성공했다는 의미, 1의 값이 오면 실패했다는 의미이다.

3.2 퍼저 환경 구성

CEC 프로토콜로 데이터를 전송하기 위해선 HDMI pin map 에 맞게 직접 케이블을 구성해야 하는 번거로움이 존재한다. 이에 Pulse-Eight에서는 USB-serial로 CEC 메시지 전송을 쉽게 할 수 있도록 USB-CEC adapter를 제작하였다. 따라서 본 논문에서는 Pulse-Eight 사의 adapter를 이용하여 퍼저를 구현하였다.

3.3 퍼저 설계 및 구현

본 논문에서 제안하는 퍼저는 앞에서 설명한 HDMI의 CEC 메시지 프레임에 따라 CEC 메시지 길이, CEC opcode, CEC operand의 항목에 대하여 퍼징한다.

첫 번째, CEC 메시지 길이에 대한 퍼징으로는 CEC를 지원하는 임베디드 기기가 16 block 이상의 메시지를 입력받을 경우 Memory Corruption 등의 취약점이 발생할 수 있다. 따라서 CEC 프로토콜 구현 시 메시지 길이에 대해 mutation 한다.

두 번째, CEC opcode에 대한 퍼징이다. CEC는 기기의 전원 및 볼륨 제어 등의 다양한 기능을 지원한다. 이와 같은 기능은 CEC 메시지의 opcode 값으로 이를 구분한다. Opcode에는 Image View On(0x04), Set OSD String(0x64) 등이 있다. CEC는 제조사별로 지원하는 기능이 상이하며 특정 제조사가 새로운 기능을 만들어 지원하기도 한다. 이때, 해당 기기의 제조사가 지원하지 않는 opcode를 전송하였을 경우 이를 처리하는 부분에서 취약점이 발생할 수 있으므로 opcode에 대해 mutation 한다.

마지막으로 CEC operand에 대한 퍼징이다. Opcode 값에 따라 0~14개의 block이 위치할 수 있다. Operand 값은 opcode에 의존적이며 각 opcode 별로 입력받는 operand의 크기와 값이 다르다. 이 부분 역시 opcode와 유사한 취약점이 존재할 수 있으므로 operand에 대해 mutation 한다.

퍼징 결과에 따른 crash 여부 판단은 다음과 같다.

- CEC 메시지를 받은 기기의 ACK 메시지가 요구되나 받지 못한 경우 crash로 판단한다.

- 기기의 crash log를 확인하여 판단한다.

위와 같은 mutation과 crash 판정 모듈을 기반으로 퍼징을 수행한다.

IV. 결론

본 논문에서는 HDMI의 프로토콜 중 CEC 프로토콜에 대해 분석하고 CEC 프로토콜에서 발생할 수 있는 취약점 점검을 효과적으로 진행할 수 있는 기법에 대해 연구를 진행하였다. 본 논문에서 제안한 CEC 프로토콜의 취약점 점검을 통해 사전에 발생할 수 있는 악의적인 공격을 예방하고자 한다.

향후 CEC 프로토콜을 제외한 다른 HDMI의 프로토콜 분석 및 취약점 점검을 필요로 한다. 이를 효과적으로 수행하기 위해 취약점 점검 기법에 대한 지속적인 연구와 취약점 점검 도구의 구현을 통한 악의적인 공격의 예방이 요구된다.

[참고문헌]

[1] HDMI Licensing Administrator, "HDMI L A The Road to 8K and Next Generation Features", HDMI Licensing, 2018.

[2] Takanen, Ari, Jared D. Demott, and Charl es Miller, "Fuzzing for software security t esting and quality assurance", Artech Hou se, 2008.

[3] Bekrar, Sofia, et al, "Finding software vulnerabilities by smart fuzzing", 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. IEEE, 2011.

[4] Toby Clarke, "Fuzzing for software vulnerability discovery", Department of Mathematic, Royal Holloway, University of London, 2009.

[5] High-Definition Multimedia Interface (HDMI) Founders, "High-Definition Multimedia Interface Specification Version 1.4", HDMI Licensing, 2009.

프로세스 로그 기반의 악성코드 수집 및 탐지 시스템

김동현*, 양준호*, 김찬진*, 염흥열*
*순천향대학교 정보보호학과

Malware detection and collection system based on process log

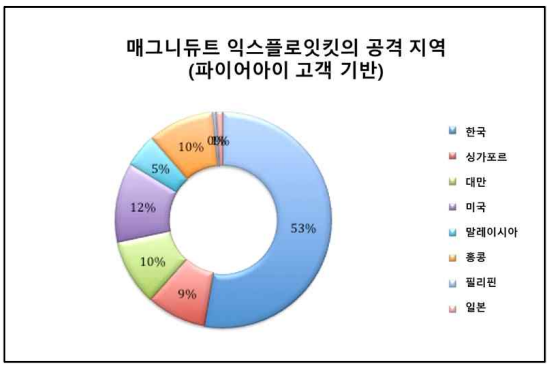
Dong-Hyun Kim*, Jun-Ho Yang*, Chan-Jin Kim*, Heung-Youl Youm*
*Department of information security, SoonChunHyang University.

요 약

현재 많은 악성코드가 Exploit-Kit(EK)를 활용하여 Drive-By-Download 기법으로 유포되고 있다. 하지만 기존의 크롤링 시스템으로는 EK를 기반으로 한 악성코드 유포를 탐지하기 어렵기 때문에 새로운 방식의 수집 시스템이 필요하다. 이에 본 논문에서는 프로세스 로그 기반으로 악성코드의 감염을 탐지하고 해당 악성코드를 자동으로 수집하는 시스템을 구현하여 피해 확산 전 대응할 수 있도록 한다.

I. 서론

현재까지 랜섬웨어를 비롯한 다양한 악성코드들이 드라이브 바이 다운로드(Drive-By-Download) 기법을 통해 유포되고 있다. 드라이브 바이 다운로드란 사용자가 웹사이트에 방문하는 자체만으로도 사용자 모르게 악성코드가 다운로드되고 실행되는 사이버 공격 방법이다[1]. 일반적으로 드라이브 바이 다운로드 기법은 익스플로잇 킷(Exploit Kit)이라는 공격 도구 모음을 사용한다. 최근에는 악성코드가 매그니튜드(Magnitude) 익스플로잇 킷을 통해 감염 시 별도의 파일이 생성되지 않는 파일리스(Fileless) 형태로 유포되고 있다[2].기존의 정적 크롤링 방식은 웹 페이지에서 다운받은 소스코드를 기반으로 동작하기 때문에 파일리스 형태로 유포되는 악성코드를 탐지할 수 없으며, 샘플을 자동적으로 수집할 수 없다는 한계점이 존재한다. 따라서 실제로 악성코드를 감염시킨 후 관련 로그를 분석하여 탐지하는 방식은 샘플까지 확보할 수 있을 뿐만 아니라 관련 로그를 확인하면 어떤 방식을 사용하여 사용자의 컴퓨터에 감염시키는지 한눈에 파악할 수 있다.



[그림 1] 매그니튜드 익스플로잇 킷 동향[3]

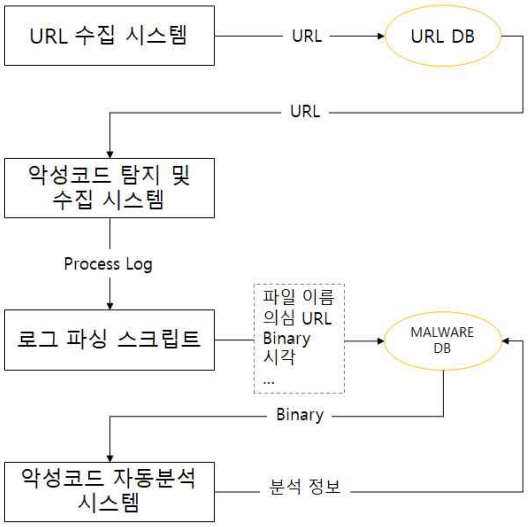
본 논문의 구성은 다음과 같다. 제2장에서는 프로세스 기반의 악성코드 수집 및 탐지 시스템의 개발 내용을 기술하고, 제3장에서는 해당 시스템의 이점에 대해서 논한다.

II. 본론

본 장에서는 악성코드 수집 및 탐지 시스템에 대해 기술하였다.

2.1 시스템 개요

본 시스템은 여러 시스템을 합쳐놓은 구조이다. 대략적인 도식은 다음과 같다.



[그림 2] 크롤링 시스템 도식도

URL 수집 시스템은 국내 도메인 URL을 수집하여 DB로 전송한다. 동적 크롤러는 DB에 저장되어있는 URL을 순서대로 크롤링 하여 악성코드 감염을 유도한다. 크롤링을 수행하는 각 가상 환경에는 프로세스에 대한 로그를 남기도록 제작된 DLL이 전역 인젝션 되어 실행되는 모든 프로세스에 대한 상세 정보가 저장된다. 저장된 로그는 로그수집 및 파싱 서버에 전송되고, 화이트리스트 기반으로 정상적인 프로세스가 아니라고 판단될 경우 악성코드로 의심되어 해당 샘플과 함께 관련 정보가 DB로 업로드된다. 기본적으로 취약한 시스템을 사용하여 악성코드의 감염을 기다리므로 일종의 허니팟 방식을 응용한 것이다.

2.2 URL 수집

사이트를 크롤링하기 위해서는 사전에 준비되어있는 URL이 필요하다. 본 시스템에서는 국내 URL을 대상으로 크롤링하기 때문에 국내 URL을 수집할 수 있는 시스템의 개발이 선행되어야 했다. URL 수집은 기본적으로 검색엔진에 사전 데이터를 검색하여 나온 결과를 바탕으로 URL을 수집하였다. 검색 결과에서 나온 URL은 GeoIP[4]로 국가를 판별하는 과정을 거친다. GeoIP는 IP 주소를 기반으로 대략적인 위치를 반환해주는 모듈이다. 이를 통해 해당 URL의 웹 서버의 위치가 한국일 경우에만 수집을 진행하여 국내 도메인에 대한 URL 데이터 셋을 구성할 수 있다.

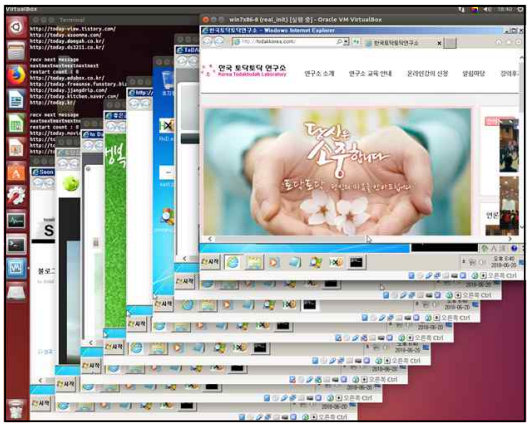
2.3 크롤링 환경 구축

기본적으로 크롤링 시스템은 서버와 클라이언트로 구성되어있다. 서버는 클라이언트를 관리하며 크롤링을 진행할 URL을 클라이언트로 전송하고 그 결과를 수신한다. 또한 악성코드에 감염된 클라이언트를 이전 상태로 초기화하여 다시 구동시키는 역할을 한다. 클라이언트는 VirtualBox[5] 가상 환경으로 구성되어 서버로부터 전송받은 URL을 브라우저를 통해 탐색하며 악성코드에 감염될 수 있도록 한다. 때문에 클라이언트 환경은 현재 익스플로잇 킷이 주로 사용하는 취약점이 잘 동작하도록 취약한 버전을 사용하여 구성하였다.

- Active Web Crawler
  - Windows 7 SP1 x86
  - IE : 10.0.9200.16438
  - Flash : 21.0.0.242
  - Java : 7.0.250
  - Silverlight : 5.1.20125.0

[그림 3] 크롤링 시스템 환경

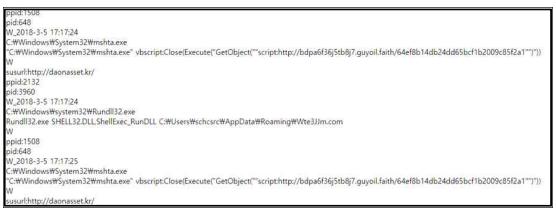
또한 각 클라이언트에는 프로세스 실행을 로깅하는 DLL가 전역 인젝션 되어 있기 때문에 실행되는 모든 프로세스의 로그가 저장되며 해당 로그는 상세 분석을 위해 서버로 전송된다.



[그림 4] 크롤링 시스템 화면

2.4 클라이언트 인젝션 DLL

각 클라이언트에는 프로세스 실행 관련 로그를 남기고 실행된 바이너리를 복제하여 백업하는 DLL이 전역인젝션 되어있다. 해당 DLL은 프로세스가 생성되고 메인스레드의 실행 직전에 호출되는 ZwResumeThread() API를 후킹하여 프로세스에 관련된 로그를 남긴다. 초기 버전에서는 CreateProcess() API를 후킹하였지만 프로세스 중 해당 API를 호출하지 않고 바로 하위 API를 호출하는 경우가 있어 로깅되지 않는 프로세스가 존재하였다. 때문에 메인스레드를 실행할 때 호출되는 API를 후킹함으로써 더 확실하게 프로세스 실행 정보를 로깅할 수 있다.



[그림 5] 로깅된 프로세스 정보

레지스트리에 AppInit_DLLs 값에 DLL의 절대 경로를 데이터로 설정하면 프로세스가 실행되며 자동으로 해당 DLL을 인젝션 시키는 과정이 수행되므로 간단하게 모든 프로세스에 DLL을 인젝션할 수 있다. 이 DLL로 인해 남겨진 로그와 바이너리는 다시 서버로 전송되어 분석된다.

2.5 로그 수집 및 파싱

2.4 절에서 서버로 전송된 로그와 바이너리는 로그 파싱 스크립트를 통해 상세 분석된다. 윈도우에서 기본적으로 발생되는 프로세스 로그를 화이트리스트로 구성한다. 만일 악성코드에 감염되면 해당 화이트리스트에 벗어나는 로그가 남겨질 것이므로 의도하지 않은 실행은 모두 악성코드로 간주할 수 있다. 의심 로그가 발생하면 로그에 기록된 PID와 PPID의 정보를 토대로 해당 로그를 실행한 IE 프로세스를 추적할 수 있다. 그리고 해당 IE의 PID로 로그를 검색하면 실행된 URL이 포함된 로그를 추적하여 어떤 URL이 악성코드에 감염된 URL인지 판단할 수 있다. 이렇게 모든 정보가 파싱되면 해당 정보(URL, 실행 날짜, 바이너리 등)를 DB에 업로드하여 연동된 웹 페이지를 통해 실시간으로 확인할 수 있다.

[그림 6] 수집된 악성코드

2.6 악성코드 자동 분석 시스템

쿠쿠박스는 바이너리를 가상 환경에서 실행하며 발생되는 이벤트들을 종합하여 사용자가 보기 쉽게 출력해주는 일종의 샌드박스이다. 2.4절의 로그 파싱 스크립트가 바이너리를 DB에 업로드하면 쿠쿠박스는 이를 자동으로 다운로드하여 샌드박스 내에서 실행시킨다. 이러한 과정을 거쳐 해당 악성코드의 주요 행위, C2서버 등을 한눈에 파악할 수 있다. 분석된 내용은 DB에 자동으로 업로드되어 연동된 웹 페이지를 통해 확인할 수 있다.

III. 결론

본 논문에서 제시한 시스템을 약 3달간 운용한 결과 국내 유명 웹하드 홈페이지에서 유포되는 파밍 악성코드를 탐지하여 빠르게 조치를 할 수 있었고, 다수의 CRBR 랜섬웨어를 탐지하였다. 또한 Myransom 랜섬웨어를 최초로 탐지하여 언론에 제보하였으며, 이후 3개월간 약 100여 개의 샘플을 수집할 수 있었다.

URL 수집 서버, 악성코드 수집 및 탐지 서버, 자동분석 서버를 연동시켜 자동화함으로써 드라이브 바이 다운로드로 유포되는 악성코드를 효과적으로 탐지하고 대응할 수 있음을 확인하였다.

제시한 악성코드 탐지 방식은 프로세스 실행에 대한 모든 정보를 기록하고 분석하므로 파일리스 악성코드 등 대부분의 악성코드 감염 기법 및 실행에 대해 탐지가 가능하다는 장점이 있다. 하지만 이것은 악성코드가 클라이언트에 감염된다는 전제가 바탕이 되는데 미래에 악성코드를 유포하는 익스플로잇 킷이 프로그램의 어떠한 버전에서 동작하도록 공격할지는 알 수 없으므로 최대한 다양한 환경을 구축해 놓는 것이 중요하다. 또한 최신 동향 분석을 통해 최근 유행하는 익스플로잇 킷이 어떠한 취약점을 사용하는지 파악하고 해당 취약점이 동작하는 버전으로 업데이트하는 것도 좋은 방안이 될 수 있다.

[참고문헌]

[1] ITWorld, "ITWorld 용어풀이 | 드라이브 바이 다운로드", http://www.itworld.co.kr/howto/93811

[2] Ahnlab, "국내 사용자 주변을 맴도는 랜섬웨어 셋", https://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?seq=27444

[3] FireEye, "FireEye reveals the latest malvertising trends", https://www.fireeye.com/blog/kr-threat-research/2017/07/fireeye-reveals-the-latest-malvertising-trends.html

[4] GeoIP, https://pypi.org/project/pygeoip/

[5] VirtualBox, https://www.virtualbox.org/

JSON 기반 원격 절차 호출 프로토콜 퍼징 자동화

이주선* 박주현 이시훈 이정민 이준오 이진우

*단국대학교 소프트웨어학과, 한세사이버보안고등학교 해킹보안과, 아주대학교 사이버보안학과, 세종대학교 정보보호학과, 고려대학교 사이버국방학과, 서울대학교 컴퓨터공학부

JSON Based RPC Protocol Automatic Fuzzing

Ju-Sun Lee* Ju-hyun Park, See-hoon Lee, Jeong-Min Lee, Jun-Oh Lee, Jin-Woo Lee

*Division of software engineering, DanKook University.

Division of Hacking Security, Hansei High School.

Division of Cyber Security, Ajou University.

Division of Information Security, Sejong University.

Division of Cyber Defense, Korea University.

Division of Computer Engineering, Seoul National University.

요 약

마이크로소프트에서 제공하는 ActiveX의 잦은 문제들로 인하여 국내에서는 이른바, Non-ActiveX라고 불리는 기술이 도입되어 많은 부분에서 사용되고 있다. 하지만, 이 기술도 기존의 ActiveX 못지않은 다수의 취약점을 가지고 있다. 해당 기술을 이용하는 많은 서비스들이 JSON 형태를 이용하여 서버와 클라이언트 사이에서 데이터를 송·수신 하는데, 해당 데이터를 다루는 부분에서 버퍼오버플로우와 같은 취약점이 발생하게 된다. 본 논문에서는 이러한 서비스 모델에서 발생할 수 있는 취약점을 자동으로 검출할 수 있는 퍼징 프레임워크에 대한 모델을 소개한다.

I. 서론

1.1 ActiveX background

마이크로소프트에서 제공하는 ActiveX는 소프트웨어 기반의 COM(Component Object Model)이다. ActiveX를 통해 사용자는 기존의 인터넷 익스플로러에서 사용할 수 있는 기능을 넘어서 Java Applets의 샌드박스 매커니즘과는 다르게 시스템 자원에 접근하는 등, 윈도우 상의 거의 모든 자원을 제한없이 사용할 수 있게 된다. 사용법 또한 간단하기에, 많은 윈도우 개발자들이 자신들이 개발한 서비스를 인터넷 익스플로러에서 사용할 수 있게 할 수 있었다.

하지만, 이 ActiveX가 제공하는 편의성이 해커에게 있어서는 다양한 공격벡터가 되었다. ActiveX 자체는 이른바 신뢰성을 기반으로 동작하기 때문에, 민감한 시스템 자원에 접근을 제한하는 샌드박스도 없으며, 한 번 설치되게 되면 운영체제의 일부분이 되며, 어떠한 행위도 원하는 때에 실행할 수 있게 된다. 이러한 특성 때문에, ActiveX가 웹 브라우저 상에서 만연하게 되었고, 많은 해커들의 표적이 되었으며, 수많은 취약점들이 공격당하였다.

이에 따라, ActiveX의 보안성을 검증하고 테스트하기 위한 ActiveX 퍼징 [1] 연구가 세계적인 보안 이슈가 되었었다. 이에 대한 결과로 Axfuzz [2], COMRaider [3], AxMan [4] 그리고 COMbust [5]와 같은 다양한 Fuzzing 도구들이

등장하였다.

국내의 경우에도 다양한 영역에서 이 Active X 기술을 사용하고 있었는데, 특히나 금융권이 보안 이슈와 밀접해 있었다. ActiveX 사용으로 인한 보안 이슈 뿐만 아니라, 인터넷 익스플로러 외에도 파이어폭스, 크롬 그리고 사파리 등과 같은 다양한 웹 브라우저를 사용하는 사람들이 늘어남에 따라, 인터넷 익스플로러에만 호환되던 ActiveX에 대한 불만이 늘어났고, 다양한 브라우저에서 사용할 수 있는 플러그인 개발을 요구하였다.

1.2 Non-ActiveX background

이러한 요구사항에 맞추고 기존 ActiveX의 보안 결함을 보완하여 나온 것이 이른바 Non-ActiveX라는 기술이다. ActiveX와 유사하게 시스템 리소스를 사용할 수 있게 해주는데, 기존의 방법과 달라진 점이라면, 사용자의 컴퓨터에서 일종의 서버·클라이언트 모델로 동작한다는 것이다. Non-ActiveX는 사용자의 컴퓨터에 .exe 확장자를 가지는 실행파일 형태로 저장되며, 설치와 동시에 항상 백그라운드에서 특정 포트를 열어둔 채로 동작한다. 전반적인 동작 개요는 [그림 1, 2, 3]과 같다. Non-ActiveX에 대한 기능을 사용할 필요가 있는 웹 페이지에서 사용자 컴퓨터의 로컬호스트로 Alive Check 패킷을 보낸다. 해당하는 서비스가 동작하고 있을 경우, 이를 받아서 응답을 돌려주지만, 그렇지 않을 경우, [그림 2]와 같이 해당 서비스 설치 페이지로 안내하게 된다. Non-ActiveX 서비스가 정상적으로 설치가 되고 해당 웹 페이지에서 웹 브라우저가 제공하는 기능 이상의 행위를 요청할 때는, [그림 3]과 같이 많은 서비스들이 JSON 형태로 데이터를 보내어 RPC(Remote Procedure Call)를 하게 된다.

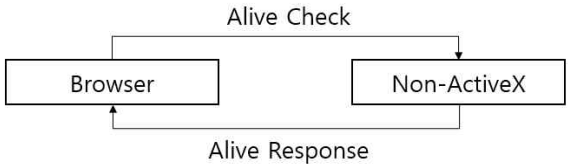


그림 1 Non-ActiveX 서비스 확인 개요도

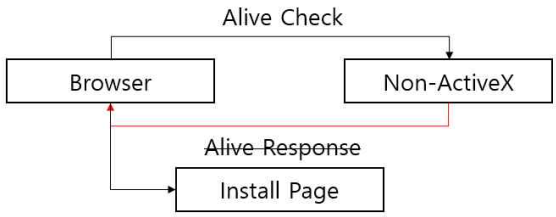


그림 2 Non-ActiveX 미설치 시 개요도

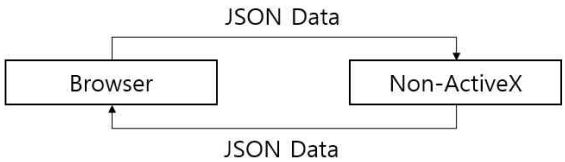


그림 3 Non-ActiveX RPC 개요도

II. Non-ActiveX Fuzzing

2.1 퍼징의 필요성

기존의 ActiveX가 가지고 있었던 문제를 보완하고 나온 Non-ActiveX도 여전히 많은 보안 문제점을 가지고 있다. 클라이언트의 역할을 하는 웹 페이지가 접속한 사용자의 서버 프로세스에 데이터를 전달하면, 서버는 해당 데이터를 읽어온 후에, 분석하고, 그에 맞는 핸들러를 호출해야한다. 이 부분에서 크게 2가지 취약점이 발생할 수 있다. 첫 번째는 메모리 커럽션류의 취약점이며, 다른 하나는 논리적인 취약점이다.

첫 번째 취약점의 경우의 예제는 간단하다. 특정 웹 페이지에 접근을 하게되면, 해당 웹 페이지에서는 임의로 데이터를 조작하여 Non-ActiveX 서비스에 요청할 수 있으며, 이 부분을 처리하는 구현이 아래의 [그림 4]와 같이 잘못되어 있으면, 치명적인 원격 코드 실행까지 이를 수 있다.

두 번째 취약점의 경우는, 검증에 대한 구현이 제대로 구현이 안되어 있을 경우 발생한다. 예를 들어, 특정 파일을 요청된 URL에서 다운로드하는 경우, 이 URL에 대한 검증이 제대로 이루어지지 않으면, 악의적인 파일을 강제로 다운로드하게끔 할 수 있으며, 경우에 따라서는 원하는 위치에 원하는 파일을 다운로드 받아 실행시킬 수도 있다.

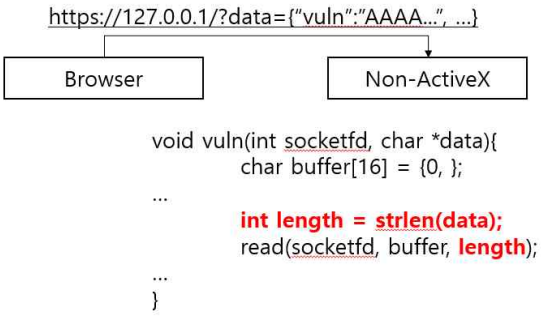


그림 4 메모리 커럽션 취약점 예제

이러한 케이스의 취약점 뿐만 아니라 다른 취약점들도 실제 Non-ActiveX 서비스에 많으며, JSON 기반으로 통신을 하는 다른 서버·클라이언트 모델에서도 잠재적인 많은 취약점이 존재한다. 서버 프로그램의 크기가 커질수록, 정적 분석만으로는 잠재적인 취약점을 많이 찾아내기 어렵다. 하지만, 현재 Non-ActiveX 취약점을 검출하기위한 퍼징 연구는 아직 국내·외로 진행된 바가 없으며, 이에 따라 구현된 퍼저 또한 전무하다. 본 논문에서는 JSON 기반으로 통신하는 Non-ActiveX 서비스 뿐만 아니라, 다른 서비스들에 대하여서도 자동으로 해당 서비스를 분석하여 퍼징을 진행하는 퍼징 프레임워크 모델을 소개한다.

2.2 Automatic Binary Analysis

JSON 기반으로 통신하는 많은 서비스들이 native c/c++로 구현되어 있다. 또한, JSON 데이터를 파싱하는데에는 많은 서비스들이 Jsoncpp 라이브러리를 사용한다. 이러한 특징에 기반하여 클라이언트와 서버가 서로 통신할 때, 주고받게되는 JSON 데이터 형식을 추론할 수 있는 모델을 만들 수 있다.

Jsoncpp 라이브러리를 사용하는 많은 서비스들에서 원하는 데이터를 가져오고, 해당하는 핸들러를 호출하는 부분에서 유사한 코드 패턴을 사용한다. 우선, JSON 데이터에서 임의로 원하는 멤버를 가져오는 코드패턴은 [그림 5]와 같다. 데이터를 처리하는 부분에서 서비스들은 공통적으로 root와 reader에 해당하는 객체를 생성하게 된다. 그리고 Reader 객체의 parse 메소드를 호출하는 시점에서 하나의 JSON 객체가 root 변수에 저장된다. 이후 Value 객체의 get 메소드를 통해 원하는 멤버의 값을 가져오게 된다. 첫 get 메소드를 통해 어떠한 멤버가 구성되는지 알 수 있으며, 이 멤버 값에 따라서 처리되는 코드 부분을 조건 분기문 분석을 통해 알 수 있다.

서비스에 따라서는 조건 분기이후, 다시 get 메소드를 통해 특정 멤버 값을 가져오고, 서브루틴에서 추가적인 데이터 처리를 하거나, 이 패턴을 다시 반복하는 경우도 있다. 정확하게 코드 패턴에 따라서 JSON 데이터를 추론하기 위해서는 서비스 바이너리를 기본 블록 단위로 나눔과 동시에, Taint Analysis가 필요하다.

```
...
Json::Value root;
Json::Reader reader;
reader.parse(JsonData, root);
std::string data = root.get(Target, "");
if(data == "..."){
        ...
}
else{
        ...
}
...
```

그림 5 Jsoncpp 코드 패턴 예시

기본 블록 단위로 나누는 기준은 그림 와 같이 모든 분기 명령어에서 나누어지게 된다. 기본 블록 단위로 나눔으로써 각 블록들은 연결되어 하나의 실행 흐름 그래프를 그리게 된다. 여기서 RootBlock을 시작 블록으로 지정하여, 이후 분기되는 모든 블록의 인자에 대하여 Taint bit를 설정한다. RootBlock으로부터 각 핸들러에 대하여 인자들을 Taint함으로써 이전 Block에서 어떠한 멤버를 가지고 있었는데 상태를 저장할 수 있다.

```
...
Json::Value root;                    RootBlock
Json::Reader reader;
reader.parse(JsonData, root);
std::string data = root.get(Target, "");
if(data == "..."){                   Block A
        ...
        dataHandler_1(root);
        ...
}
else{                                Block C
        ...
        dataHandler_2(root);
        ...
}
```

그림 6 기본 블록 단위로 나누기

2.3 Template Generate and Mutation

기본 블록 단위로 나누고, 각 핸들러 별로 taint하여 얻은 정보를 바탕으로 Non-ActiveX 서비스가 받아서 처리하게되는 JSON 데이터 템플릿을 만들어낼 수 있다. 각 JSON 데이터의 멤버들이 설정되면, 이에 대한 값을 생성해야 한다. 각 멤버들에 대하여 무작위로 값을 만들어내면 공격이 가능하거나, 유효한 샘플을 만들기가 어렵다. 본 논문에서 제안하는 퍼징 모델에서는 [그림 6]와 같이 root 객체가 사용되는 부분마다 taint하여 특정 멤버 값의 형태를 추론한다. 이를 통해, 정상적인 통신에서 사용하는 데이터 형태를 찾아낼 수 있으며, 이 형태를 바탕으로 Generation Fuzzing [6]을 진행할 수 있다.

2.4 Crash Monitoring

Linux 환경 기반에서 Crash를 탐지하는 방법은 2가지로 분류할 수 있다. 첫 번째로 GDB를 해당 프로세스에 연결하여 상태를 감시하고, crash가 발생하였을 때, callstack과 레지스터, 메모리 상태를 저장하여 주는 것이다. 하지만, 디버거를 프로세스에 붙여서 감시하는 과정에서 해당 프로세스의 실행 흐름에 영향을 줄 수도 있기 때문에 잘못된 결과를 얻을 가능성이 있다.

또 다른 방법으로는 dmesg를 확인하는 것이다. 메모리 커럽션이 발생한 경우, dmesg에 callstack과 레지스터 상태가 기록되므로, 이를 읽어들여서 상태 판단을 하는 방법이 있다.

III. 결론

ActiveX에 이어서 Non-ActiveX도 보안 취약점으로부터 극복하지 못하였다. 기존의 ActiveX에서 활발히 연구되었던 퍼징분야도 현재의 Non-ActiveX에서도 많이 연구될 필요가 있다. 본 논문에서 제시하는 퍼징 모델은 바이너리를 기본 블록 단위로 나누어, Jsoncpp 라이브러리의 코드 패턴을 따라 퍼징 템플릿을 생성한다. 데이터 타입까지 알아낼 수 있다는 장점은 있으나, 현재 모델 상 메모리 커럽션만 테스트할 수 있다는 한계점을 가지고 있으며, 보다 다양한 Json 관련된 라이브러리에 있어서는 퍼징이 불가능하다는 한계점이 있다. 메모리 커럽션 이외의 논리적인 취약점 또한 발견할 수 있도록 하며 보다 많은 라이브러리의 코드 패턴을 인식할 수 있도록 하는 추후 연구가 필요하다.

[참고문헌]

[1] Will Dormann and Dan Plakosh, Vulnerability Detection in ActiveX Controls through Automated Fuzz Testing, Carnegie Mellon University White paper, 2008.

[2] SourceForge.net: axfuzz, http://sourceforge.net/projects/axfuzz/

[3] Fuzzing Software Tools // iDefense Labs, http://labs.idefense.com/software/ fuzzing.php#more_comraider

[4] AxMan ActiveX Fuzzer, http://www.metasploit.com/users/hdm/tools/axman/

[5] COMbust, http://www.blackhat.com/presentations/bh-usa-03/ bh-us-03-bretmounet-combust.zip

[6] Charlie Miller and Zachary N. J. Peterson, Analysis of Mutation and Generation-Based Fuzzing, defcon 15, March, 2007.

AFL 기반 퍼징 기법에 대한 연구

김현준, 이영한, 양명훈, 김기열, 백윤흥*
*서울대학교 전기정보공학부

A Study on AFL-Based Fuzzing Technique

Hyun jun Kim, Younghan Lee, MyongHoon Yang, Giyeol Kim, Yun heung Paek*
*Department of Electrical and Computer Engineering, Seoul National University.

요 약

최근에 많은 연구가 이루어지고 있는 퍼징(Fuzzing)은 대상 프로그램의 버그를 발생시키는 입력값을 자동으로 찾아내는 기술로, 소프트웨어의 취약점을 효율적으로 발견해낼 수 있는 중요한 기술이다. 해당 분야에서 가장 획기적인 결과를 가져왔던 퍼저(Fuzzer)는 AFL이다. AFL 개발 이후 AFL을 기반으로 추가적인 퍼징 전략을 적용시킨 여러 퍼저들이 개발되었으며, 본 연구에서는 AFL을 기반으로 한 퍼저에서 사용되는 퍼징 기법들을 살펴본다.

I. 서론

점점 수와 복잡성에 있어 커져만 가는 보안 위협과 더불어 '퍼징(Fuzzing)'이라는 기술에 대한 관심이 점점 증가하고 있다. 퍼징이란 대상 프로그램에게 버그를 발생시키는 입력값을 찾아내는 기술로서, 사람이 직접 프로그램을 이해하고 찾아내는 것이 아닌 자동으로 프로그램을 분석하여 해당 입력값을 찾는 기술이다. 그리고 이 입력값을 통해 공격당할 가능성이 있는 취약점 어디에 있는지를 도출해낼 수 있고, 이 코드를 패칭(Patching)함으로서 해당 취약점을 제거할 수 있다.

과거의 퍼징 기법은 프로그램에 대한 정보 없이 랜덤으로 인풋을 생성하거나, (Black-box Fuzzing) 소스 코드에 대해 여러 가지 분석 기술을 사용하여 (White-box Fuzzing) 속도가 매우 느리거나, 사람의 도움을 받아 사용되었다. 하지만 퍼징 기법들이 연구되면서 바이너리에 대해서도 퍼징이 가능하며, 가벼운 분석만으로도 효율적인 퍼징이 가능한 기법들 (Grey-box Fuzzing)이 개발되기 시작했다.

그 중에서도 특히 AFL (American Fuzz Lop)이라는 퍼저가 개발되면서 획기적으로 많은 버그들을 자동으로 발견해 낼 수 있었다. 이 퍼저는 시드(Seed)라는 유효한 입력값을 특정 패턴에 따라 변형을 시킨다. 그리고 변형된 입력값으로 프로그램을 실행시켜. 해당 프로그램이 CFG 상에서 새로운 경로를 따라 실행되면, 해당 값을 시드로서 입력 큐에 다시 넣어 추후에 다른 입력값을 생성하는 데에 사용한다. 이 유전 알고리즘 (Genetic Algorithm)을 통해 다양한 경로를 시도해볼 수 있는 쪽으로 방향성을 줄 수 있고, 결과적으로 버그를 발생시키는 입력을 찾아낼 가능성이 높아지게 된다.

하지만 AFL은 프로그램의 특성에 특화되어 있지 않다. 따라서 프로그램 분석을 통해 얻은 정보를 활용하여 퍼징을 수행하면, 좀 더 효과적인 퍼징이 가능하다. 본 논문에서는 AFL을

기반으로 새로운 전략을 사용하여 성능 향상을 도모하며 발전해 온 메인스트림 퍼저 3가지를 소개한다.

II. AFL 기반 퍼징 기법

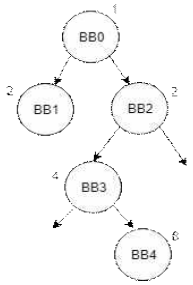
AFL은 Nested Branch가 있을 경우, 좀 더 깊이가 깊은 Branch에 도달하기가 매우 힘들다. AFL의 입력값 변형은 비트/바이트 뒤집기, 특정 값 더하기, 특정 값 넣기, 랜덤 값 넣기 등을 통해 이루어진다. 그리고 입력 값의 변형을 통해 Branch의 Constraint를 만족/불만족시켜서 가보지 못한 경로를 가게 된다. 하지만 Nested Branch의 경우는 여러 개의 Branch Constraint를 만족시켜야 하므로, 깊은 경로로 들어갈 수 있는 입력값이 매우 희소해져 많은 시도가 필요하며, 입력값의 변형한 부분이 바깥쪽 Branch에도 영향을 준 경우, 이미 해결된 바깥쪽 Branch Constraint를 다시 해결하게끔 만들 수도 있다. 그래서 이러한 Constraint를 효과적으로 처리하는 퍼징 기법들이 후에 기술될 것이다.

1.1 드릴러(Driller)

드릴러[1]는 AFL과 선택적인 콘콜릭 실행(Concolic Execution)을 결합하여 사용하였다. 콘콜릭 익스큐션은 프로그램이 실행될 수 있는 모든 경로를 분석하는 기술로, 특정 기호(Symbol)를 통해 변수를 표현하여, 경로에 따라 그 변수가 어떻게 변하는지를 기록한다. Branch에서는 이 기호로 나타내어진 변수를 솔버(Solver)를 통해 해당 조건을 만족시킬 수 있는 기호의 범위/조건을 역산할 수 있고, 이에 따라 이 Constraint를 만족시키는 입력값을 얻어낼 수 있다. 하지만 이 방식은 Branch Depth에 따라 그 경로의 수가 배로 증가하므로 경로 폭발 문제가 발생하고, 입력값을 찾아내는 데에 너무 많은 시간과 자원이 소모된다. 드릴러는 AFL을 사용하여 이 문제를 완화하였다. 처음에는 AFL을 통해 퍼징을 수행하고, 더 이상의 새로운 경로를 찾을 수 없다고 판단되면, 콘콜릭 실행을 수행한다. 콘콜릭 실행 엔진을 통해 해당 Branch에서 가보지 못한 경로를 통과하는 입력값을 찾으면, 그 입력값을 AFL에서 변형하여 퍼징을 다시 수행한다. 이 방식은 특정 Branch에 대해 한정된 경로만을 수행하므로 경로 폭발 문제를 최대한 완화시킬 수 있다. 하지만 프로그램 크기가 커지고, 깊은 Branch에 도달할수록 더 많은 기호들을 분석해야 하므로 여전히 경로 폭발 문제가 존재한다.

1.2 버저(VUzzer)

버저[2]는 AFL에 추가적인 방법론을 사용하여 깊은 분기문 도달 문제 해결을 시도했다. 버저는 프로그램에 대해 정적, 동적 분석을 수행한다.



[그림 1] CFG 가중치 산정

정적 분석 단계에서는 대상 바이너리의 cmp, lea 명령어에 사용되는 값을 전부 저장해둔다. 그리고 바이너리의 CFG를 분석해 좀 더 깊은 Branch 연결된 Basic Block에 더 높은 가중치를 준다. 이 값은 나중에 다른 경로를 도출해낸 특별한 입력값들을 다시 입력 큐에 넣을 때, 경로를 따라 이 가중치를 더한 값이 제일 높은 입력값만 우선시하여 입력 큐에 넣는다. 결과적으로 더 깊은 Branch를 지나게 만드는 입력값만 큐에 남게 되고, 깊은 Branch 지나가는 인풋을 더 빠르게 생성할 수 있다.

동적 분석 단계에서는 테인트 분석을 수행하여, 특정 Branch Constraint와 바이너리로 표현된 입력값 비트 중 관련성이 있는 특정 부분 간의 관계가 있다는 사실을 추적해낼 수 있다. 그리고 이 특정 부분에 정적 분석 단계에서 계산한 cmp, lea에서 사용된 값을 넣어줄 경우, Branch Constraint를 만족시킬 가능성이 존재한다.

하지만 Constraint와 테인트 분석으로 찾은 입력값의 특정 부분 간에 복잡한 관계가 성립할 경우는 통과를 시키지 못한다는 한계점이 존재한다.

1.3 앙고라(Angora)

앙고라[3]는 좀 더 세밀한 방식으로 분기문 조건 문제를 해결하였다. 앙고라는 초기 단계에 AFL과 동일하게 시드를 입력값으로서 넣어 프로그램을 실행시킨다. 이 과정에서 CFG 상에서 커버된 경로들을 트래킹하며, 좀 더 세밀한 테인트 분석을 통해 입력값의 특정 부분과 어떤 Branch Constraint의 관계를 알아내며, 입력값을 로드하는 시스템 콜을 분석하여 그 특정 부분이 실제로 프로그램에서 변수로서 로드되는 타입과 크기를 분석한다. 그리고 아직 한쪽 경로밖에 탐색 되지 않은 Branch 중 하나를 선택하여 알고리즘을 수행한다.

해당 Branch에 Constraint에 해당하는 두 변수 x, y에 대한 비교 조건을 특정 함수 f에 관한 비교 조건으로 변환할 수 있다.

x, y constraint	f(x, y)	f (x, y) constraint
x < y	f = x - y	f < 0
x <= y	f = x - y	f <= 0
x > y	f = y - x	f < 0
x >= y	f = y - x	f <= 0
x == y	f = abs(x-y)	f == 0
x != y	f = -abs(x-y)	f < 0

[표 1] Constraint의 변환

결과적으로 함수 f가 0 이하/미만의 값을 가지면 해당 Branch의 Constraint가 풀리게 된다. 그리고 해당 Constraint의 x, y는 테인트 분석에 의해 입력값이 로드된 특정 부분값 a_1 ~ a_n과 관련성이 있음을 알 수 있었다고 하자. 그렇다면 이 a_1 ~ a_n에 해당하는 부분을 변형시키면서 f의 값을 관찰해볼 수 있다. 프로그램이 잘못 짜여진게 아니라면 f의 값이 0보다 작아지는 순간이 반드시 존재할 것이며, 이를 그래디언트 디센트 알고리즘 (Gradient Descent Algorithm)을 통해서 알아낸다. 이는 a_i를 조금씩 변형시키면서 (a_i의 타입에 대해 최소 크기) $\nabla f_{a_i}(a_i)$를 계산하고, a_i를 $a_i - \varepsilon \nabla f_{a_i}(a_i)$로 업데이트시키며 f의 극솟값을 찾아가는 알고리즘이다. 만약 이 알고리즘이 극솟값에 도달해도 0이 되지 않았다면, a_i들을 랜덤으로 변형시키고 다시 그래디언트 디센트 알고리즘을 수행하여 더 작은 다른 극소값을 찾는다. 따라서 해당 분기문 Constraint를 푸는 입력값을 얻을 수 있게 된다.

1.4 앙고라의 개선 방향

앙고라는 현존하는 퍼저 중에 최고의 성능을 자랑한다. 하지만 여전히 깊은 Branch에는 잘 도달하지 못하는 문제가 있다. 그 이유는 여러 개의 조건이 겹치면서 서로 관련되는 변수들이 많아지고, 이에 따라 연산할 그래디언트 디센트 값도 많아지기 때문이다.

따라서 바깥쪽 Branch에 대한 정보를 더 많이 수집해서 더 깊은 Branch의 Constraint 해결에 도움이 되는 방식을 제안할 수 있겠다. 바깥쪽 Branch에 대한 그레디언트 디센트 알고리즘에서 0보다 작은 값을 만나면 바로 알고리즘을 종료시키지 않고, 로컬 미니멈을 찾을 때까지 수행한 다음, 로컬 맥시멈에 대한 그레디언트 어센트 알고리즘을 수행하여 f가 0 이하/미만을 가지는 구간의 범위를 구한다. (f가 0 이하/미만을 가지는 모든 구간을 구하는 것은 시간이 오래 걸리므로, 추후에 필요할 경우에 랜덤한 다른 값을 넣은 뒤, 그래디언트 디센트, 어센트 알고리즘 수행) 그리고 해당 브렌치보다 더 깊은 브렌치에 대해서 알고리즘을 수행할 때, 이전에 계산해놨던 바깥쪽 Branch Constraint 통과에 필요한 변수들의 범위를 알고 있으므로 공통되는 관련 변수들의 탐색 범위를 제한시킬 수 있으며, 이에 따라 해당 변수에 대한 연산을 최소화시킬 수 있을 것이다. 그리고 결과적으로 더 빠르게 깊은 Branch에 도달할 수 있을 것이다.

III. 결론

본 논문에서는 AFL을 기반으로 해서 다양한

방법론들을 사용해 AFL을 문제점을 보완하고 성능을 높인 퍼저들을 살펴보았다. 각 퍼저들마다 다양한 전략을 사용해 문제를 해결하려 시도하였고, 이러한 전략들을 조합 또는 응용하거나 다른 창의적인 방법을 사용하면 새로운 퍼저를 만들어 낼 수 있을 것이라 예상된다.

IV. ACKNOWLEDGEMENT

이 논문은 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었으며 (IITP-2017-2015-0-00403), 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원과 (NRF-2017R1A2A1A17069478), 반도체공동연구소 지원의 결과물임을 밝힙니다.

[참고문헌]

[1] Stephens, Nick, et al. "Driller: Augmenting Fuzzing Through Selective Symbolic Execution." NDSS. Vol. 16. 2016.

[2] Rawat, Sanjay, et al. "Vuzzer: Application-aware evolutionary fuzzing." Proceedings of the Network and Distributed System Security Symposium (NDSS). 2017.

[3] Chen, Peng, et al. "Angora: efficient fuzzing by principled search." IEEE Symposium on Security & Privacy. Vol. 14. Springer-Verlag New York, 2013.

모바일 포렌식 가이드라인 현황 연구*

김해니*, 권태경*
*연세대학교 정보대학원 정보보호 연구실

A Study on Mobile Forensics Guidelines

Haeni Kim*, Taekyoung Kwon*
*Information Security Lab, GSI, Yonsei University.

요 약

최근 많은 소송에서 모바일 기기의 디지털 증거 비중이 높아짐에 따라 모바일 포렌식의 중요성이 확대되고 있다. 국내외에서는 나라별 법률적, 기술적, 절차적인 실정을 반영하여 다양한 기관에서 모바일 포렌식 관련 가이드라인을 제정 및 개정함으로써, 모바일 포렌식을 수행하는 개인을 대상으로 절차 및 획득 방법과 관련된 기초적인 지식에 대하여 가이드하고 있다. 그러나 각 가이드라인이 정의하는 모바일 기기의 데이터 획득 관련 용어가 하나로 통일화되지 않는 문제점이 존재한다. 따라서, 본 논문에서는 국내외에서 제정 및 개정된 가이드라인 현황과 용어를 파악하고 모바일 포렌식의 데이터 획득에 관한 용어 표준화를 제시하였다.

I. 서론

전 세계적으로 정보 통신 기술이 발전함에 따라 다양한 모바일 기기가 보급되면서 사용자들의 모바일 기기 사용률이 증가하고, 이를 활용한 신종 범죄 또한 가속화되고 있다. 따라서, 최근 많은 소송에서 모바일 기기의 디지털 증거 비중에 높아짐에 따라 모바일 기기를 대상으로 한 디지털 증거 획득의 중요성이 확대되고 있다.

이에 맞춰 현재 다양한 기관에서는 나라별 법률적인 실정을 반영한 모바일 포렌식 관련 가이드라인을 제정 및 개정함으로써 모바일 포렌식을 수행하는 개인을 대상으로 한 기본적인 지식과 절차 및 기기 내 데이터 획득 방법에 관한 내용을 문서로 표준화하고 있다. 그러나, 이러한 기관들에서 정의하고 있는 모바일 기기를 대상으로 한 데이터 획득 관련 용어들은 현재 하나로 통일화되어 있지 않은 문제점으로 인하여 모바일 포렌식을 수행하는 관련 업종의 사람들에게 혼란을 주고 있다. 따라서, 본 논문에서는 국내외 표준 제정기관으로부터 모바일 포렌식 관련 가이드라인의 현황을 파악하고, 각 가이드라인에서 의미하는 데이터 획득 방식에 관한 용어를 재해석한 후, 안드로이드를 대상으로 한 모바일 포렌식 데이터 획득 방식 용어 표준화를 제시한다.

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00-304).

II. 모바일 포렌식 가이드라인 현황

디지털 포렌식은 적용 대상에 따라 컴퓨터 포렌식, 모바일 포렌식, 네트워크 포렌식, 시스템 포렌식 등으로 분류할 수 있으며 분석 대상에 따라 절차와 방법 등이 달라진다. 가장 널리 알려진 컴퓨터 포렌식의 경우, 국내외적으로 절차 및 방법에 관한 가이드라인이 제정 및 개정되었다. 이후 모바일 기기의 발전이 가속화되면서 모바일 포렌식의 중요성 또한 급증하게 되었다. 다른 기기들과의 특성과 기능이 매우 다른 모바일 기기는 컴퓨터 포렌식을 대상으로 한 가이드라인의 절차 및 방법을 적용하기에는 현실적으로 해결되지 않는 문제점이 많이 발생한다. 따라서, 모바일 포렌식에 부합하는 가이드라인을 국내 및 국외에서 국가별 현황에 따른 가이드라인을 발표하였다.

2.1 국내

국내 모바일 기기 사용자의 비율은 전 세계적으로 비교하여도 매우 높은 수준이다. 그러나 모바일 포렌식과 관련된 가이드라인을 표준화한 기관으로는 한국정보통신기술협회(Telecommunications Technology Association, TTA)가 유일하다. 이는 1998년에 설립된 국내 유일의 정보통신단체 표준 제정기관으로 2007년 12월 26일 'TTAS.KO-12.0058 컴퓨터 포렌식 가이드라인'과 'TTAS.KO-12.0059 이동전화 포렌식 가이드라인'을 동시에 제정하였다. 컴퓨터 포렌식 가이드라인의 경우 2017년 최신화되어 개

Table 1. Guidelines by Institution

Institution	Guidelines name (Establishment/Revision Date)
NIST (National Institute of Standards and Technology)	NIST Special Publication 800-101 Guidelines on Cell Phone Forensics (2007. 5. 30. Establishment), NIST Special Publication 800-101 R1 Guidelines on Mobile Device Forensics (2014. 5. 15. Revision)
SWGDE (Scientific Working Group on digital Evidence)	Best Practices for Mobile Phone Examinations (2009. 5. 21. Establishment), SWGDE Best Practices for Mobile Phone Forensics (2013. 2. 11 Revision), SWGDE Core Competencies for Mobile Phone Forensics (2013. 2. 11. Establishment), SWGDE Best Practices for Examining Mobile Phones Using JTAG, (2016. 2. 8. Establishment), SWGDE Best Practices for Chip-Off (2016. 2. 8. Establishment), SWGDE Best Practices for Collection of Damaged Mobile Devices (2016. 2. 8. Establishment), SWGDE Best Practices for Mobile Device Evidence Collection, Preservation and Acquisition (2018. 12. 1. Establishment)
ASTM (American Society for Testing and Materials)	3046-15 Standard Guide for Core Competencies for Mobile Phone Forensics (2015. 12. 1. Establishment)
ACPO (Association of Chief Police Oficers)	Good Practice Guide for Computer-Based Electronic Evidence ACPO Good Practice Guide for Digital Evidence, (2012. 3. Revision)

정되었으나, 모바일 포렌식 관련 가이드라인의 경우 다양한 모바일 기기가 보급되고 기술이 발전하면서

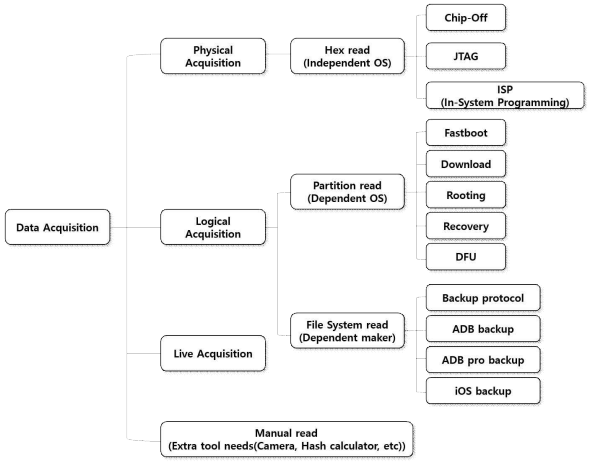


Fig 1. Mobile Forensics Data Acquisition Methods

기기별 데이터 획득 절차 및 방법이 크게 변화되었지만 최근 실정을 반영한 가이드라인이 개정되지 않는 상황이다.

2.2 국외

국외의 경우 국내와 달리 모바일 기기가 기술적으로 발전함에 따라 최근 실정을 반영하여 모바일 포렌식 가이드라인을 개정한 표준을 제공하고 있다. 미국에서는 현재 국립표준기술연구소(NIST), 범죄과학수사그룹(SWGDE), 미국재료시험협회(ASTM) 기관에서 모바일 포렌식 관련 가이드라인을 표준하고 있으며, 영국에는 경찰총장협회(ACPO)에서 디지털 증거와 관련된 가이드라인을 제정 및 개정하고 있다. 각 기관의 가이드라인 현황 대한 개요는 Table 1과 같다.

III. 기관별 모바일 포렌식 데이터 획득 방법 용어 해석

3.1 물리적 획득

위 기관들은 물리적 획득에 대하여 모바일 기기의 내부 메모리의 데이터의 전체 헥스 값을 읽어 비트 단위로 데이터를 획득하는 과정으로 정의하고 있다. 그러나 이와 같은 정의로 인하여 물리적 획득에 해당하는 방법에 관하여 기관별로 서로 다르게 구분하고 있다.

한국정보통신기술협회의 경우, 플래시메모리를 분리하여 직접 메모리를 읽는 방법 즉, Chip-Off와 JTAG와 같은 표준 하드웨어 인터페이스를 이용한 방법을 포함하지만, NIST의 경우 수정된 부트로더 또는 기타 소프트웨어를 모바일 기기의 메모리 보호 영역에 업로드하는 방식과 JTAG, Chip-Off 방법을 포함한다. 또한 SWGDE는 구체적으로 방법을 구분하지 않았으나 휴대전화를 분해하거나 분해하지 않고도 데이터를 획득할 수 있다고 정의하고 있다. 마지막으로 ASTM에서는 NIST와 유사하게 Hex Dumping/JTAG, Chip-Off를 물리적인 획득 방법에

Table 2. Definition of Data Acquisition Terminology

Terminology	Definitions
Physical Acquisition	운영체제 관계없이 모바일 기기 플래시메모리 전체 Hex 값을 읽어 데이터를 비트 단위로 획득하는 과정
Logical Acquisition	운영체제 혹은 기기 제조업체에 종속적인 방식으로 파티션 또는 파일시스템을 읽어 데이터의 전체 혹은 일부를 획득하는 과정
Live Acquisition	모바일 기기의 정상적인 부팅 이후 기기의 데이터로부터 접근 권한이 있는 파일에 대하여 획득을 수행하는 과정
Manual Read	모바일 기기의 키패드 혹은 핸드셋 디스플레이를 수동으로 조작하여 화면을 보고 확인한 데이터를 직접 기록하거나, 사진 혹은 동영상 촬영으로 산출된 데이터의 해시값을 획득하는 과정

포함한다.

3.2 논리적 획득

논리적 획득의 경우 대부분 용어에 대한 정의가 아닌 논리적 획득 방식에 포함되는 방법들로 구분하고 있다.

먼저, 한국정보통신기술협회에서는 이동 전화 내의 파일을 복사하는 방법으로 파일전송 프로토콜을 이용하여 이동 전화의 파일을 직접 받는 방법과 제조사의 PC 소프트웨어를 이용한 방법으로 논리적 획득을 구분한다. NIST의 경우, 모바일 기기로 설정된 인터페이스를 통해 일련의 명령을 전송함으로써 데이터를 획득하는 방법으로 정의하고 있으며, SWGDE와 ASTM에서는 파일시스템 일부를 추출하는 프로세스로 사용자가 액세스할 수 있는 파일에 액세스하여 수집하는 방법이라는 동일 의미로 논리적 획득에 대하여 정의하고 있다.

3.3 수동적 획득

수동적 획득은 한국정보통신기술협회를 제외한 네 기관에서 모바일 기기의 키패드 및 핸드셋 디스플레이를 수동적으로 작동하여 화면에 표시되는 저장된 데이터 내용을 기록하는 과정이라는 동일 의미로 정의하고 있다.

IV. 해석 표준화 용어 제시

본 항에서는 위 기관의 제·개정된 가이드라인 문서 내 데이터 획득 방법에 관한 용어를 해석한 후 용어 표준화를 제시한다. 데이터 획득 방법은 크게 물리적 획득, 논리적 획득, 라이브 획득 및 수동 읽기네 가지로 구분하며 해석 용어에 대한 정의는 Table 2와 같다. 물리적 및 논리적 방법을 구분하기 위하여 운영체제와의 관계 여부로 나누는 것으로 하여, 제시된 방법에 따라 각 데이터 획득 방식은 Fig 1과 같이 구분되며 다음과 같은 방법들을 포함할 수 있다.

먼저, 물리적 획득 방식은 운영체제와 독립적인 방식이므로 재부팅 과정이 필요하지 않은 Chip-Off, JTAG 및 ISP(In-System Programming) 방법이 포함된다. 이 방법들은 대상 기기의 케이스를 분해하여 메인보드로 접근하여야 하므로 모두 기기 분해적인 방식이라고 할 수 있다.

논리적 획득의 경우, 운영체제에 종속적인 방식으로 대상 기기의 우회 부팅을 통하여 수정된 부트로더 또는 기타 소프트웨어를 모바일 기기의 메모리 보호 영역에 업로드하는 방법과 운영체제 및 제조업체에 종속적인 기기의 백업 데이터를 획득하는 방법을 포함한다.

라이브 획득은 논리적 획득과 달리 대상 기기의 정상적인 부팅 이후 활성 데이터를 획득하는 방식이다. 따라서 모바일 기기의 플랫폼별로 획득 가능한 데이터 정보의 차이가 있다.

수동 읽기는 포렌식에 대한 지식이 없는 일선 수사관도 수행이 가능한 방법으로 대상 기기의 디스플레이에서 표시하는 데이터를 단순하게 확인하여 기록하는 과정을 수행하는 방법이다.

V. 결 론

본 연구에서는 국내외 기관별 모바일 포렌식 관련 가이드라인의 현황을 파악하고, 각 기관에서 정의하고 있는 데이터 획득 관련 용어를 해석하여 통일화되지 않은 용어에 대한 문제점을 지적하였다. 이에 따라, 데이터 획득 방식을 물리적 획득, 논리적 획득, 라이브 획득, 수동 읽기로 구분하여 해석 용어 표준화를 제시하고 해석에 따르는 각 데이터 획득에 포함되는 방법을 구분하여 제시하였다.

본 연구에서 제안한 모바일 포렌식에서의 데이터 획득 방법은 이후, 국내외적으로 확립되지 않은 모바일 포렌식에서의 용어를 하나로 확립하는데 활용될 수 있으며, 향후 국내 모바일 포렌식 관련 가이드라인 및 도구 요구사항에 관한 표준 가이드라인 개정

시 기여할 수 있을 것으로 예상된다.

[참고문헌]

[1] 이석희, 이상진, 길연희, 은성경, 홍도원, 인재형, "TTAS.KO-12.0059 이동 전화 포렌식 가이드라인(Guidelines on Cellular Phone Forensics)," 한국정보통신기술협회, Dec. 2007
[2] Wayne Jansen and Rick Ayers, "NIST Special Publication 800-101 Guidelines on Cell Phone Forensics," NIST, May. 2007
[3] Wayne Jansen, Rick Ayers and Sam Brothers, "NIST Special Publication 80-101 R1 Guidelines on Mobile Device Forensics," NIST, May. 2014
[4] SWGDE, "Best Practices for Mobile Phone Examinations," May. 2009
[5] SWGDE, "SWGDE Best Practices for Mobile Phone Forensics," Feb. 2013
[6] SWGDE, "SWGDE Core Competencies for Mobile Phone Forensics," Feb. 2013
[7] SWGDE, "SWGDE Best Practices for Examining Mobile Phones Using JTAG," Feb. 2016
[8] SWGDE, "SWGDE Best Practices for Chip-Off," Feb. 2016
[9] SWGDE, "SWGDE Best Practices for Collection of Damaged Mobile Devices," Feb. 2016
[10] SWGDE, "Best Practices for Mobile Device Evidence Collection, Preservation and Acquisition," Dec. 2018
[11] ASTM, "3046-15 Standard Guide for Core Competencies for Mobile Phone Forensics," Dec. 2015
[12] ACPO, "Good Practice Guide for Computer-Based Electronic Evidence,"
[13] ACPO, "Good Practice Guide for Digital Evidence," Mar. 2012

Two-Factor 암호화 통한 자료 유출방지 방안

김민창* 김지원*
*육군사관학교 컴퓨터과학과
alsckd7117@kma.ac.kr admin486@kma.ac.kr

A Study on the Prevention of Data leakage through Two-Factor Encryption

Min-hang Kim* Jee-won Kim*
*Department of Computer Science, Korea Military Academic.

요 약

우리나라 대부분의 인터넷 사용자는 공인인증서를 발급받았으며 이를 이용해서 은행 입·출금, 계좌이체, 온라인 쇼핑 등 다양한 업무에 사용하고 있다. 그렇기 때문에 공인인증기관 및 보안 관련 업체에서는 공인인증서를 사용자의 데스크탑(PC) 하드디스크에 저장하지 말고 보안을 위해서 USB메모리 및 휴대용 저장장치에 저장하고 사용하는 것을 권장하고 있다. 그럼에도 불구하고 공인인증서의 해킹은 지속적으로 발생하고 금전적 피해도 잇따르고 있다. 이에 본 연구는 사용자가 소유한 USB메모리와 PC의 정보를 이용한 이중 암호화(Two-factor)방식을 제안하여 개인키 파일의 안전한 관리방안을 제시하고자 한다.

I. 서론

우리나라의 인터넷 사용자들의 대부분은 온라인 뱅킹 서비스, 주식거래, 온라인 쇼핑 등의 전자거래 뿐 아니라 행정업무, 학사업무 등 다양한 분야에서 공인인증서를 활용하고 있다. 2017년 한국정보통신진흥원(KISA)의 대국민 전자서명 이용실태 조사에 따르면, 발급된 공인인증서는 총 3,388만 건에 이르고, 이 중 개인이 발급받은 건은 3,011만 건으로, 이 수치는 경제활동 인구를 능가하는 수치이다. 이처럼 공인인증서는 인터넷상에서 여러 활동을 하는 사람들이 실생활에 활용하고 있으며, 온라인상에서 자신의 신분을 증명하는 중요한 수단이 되어가고 있다. 공인인증서의 저장매체 수단 중 가장 높은 비율은 USB 메모리(62.2%)로 나타났고, 다음으로는 PC 하드디스크(56.4%)가 뒤를 따른다 [1]. 이것은 하드웨어를 통한 공인인증서의 복사 및 탈취의 가능성이 존재한다는 것을 의미한다. 이러한 공인인증서에 대한 위협에 대응하기 위해 기존의 보안토큰이나 USIM 스마트 인증과 같은 서비스가 제공되고 있지만 별도의 금전적 지불이 필요하다. 또한, 공인인증서 자체의 분실 및 개인키 노출에 대한 보호방안도 강구되지 않은 상태이다. 따라서 본 연구에서는 대부분의 공인인증서 사용자들이 공인인증서 저장매체로 사용하고 있는 USB메모리의 Container ID 및 PC 고유의 MAC 주소 값을 조합하여 활용하는 Two Factor 인증방식을 도입하여 공인인증서의 보안을 강화하는 방안을 제시하고자 한다.

II. 이론 및 문제점

공개키 기반구조(Public Key Infrastructure, PKI)는 공개키 암호 방식을 바탕으로 한 디지털 인증서를 활용하는 소프트웨어, 하드웨어, 사용자, 정책 및 제도 등을 총칭하여 일컫는 말이다. 공인인증서는 공개키 증명서, 디지털 증명서, 전자 증명서 등으로 불리며, 개인키와 한

쌍으로 존재하며, 이는 안전하게 나누어주는 역할을 담당하는 신뢰된 제3자(인증기관)의 존재를 전제로 하고 있다. 공인인증서는 사용자가 인증서 발급을 요청함에 따라 생성된 공개키 파일, 개인키 파일, 인증서 체인 파일로 구성되어 있으며 이러한 파일들은 현재 특정 PC의 하드디스크나 USB메모리에 저장되어 이용되어진다[2]. 우리나라 공인인증서의 특징은 파일들이 보관, 저장되는 위치와 방법이 독특하여 웹브라우저로는 사용이 불가능하므로 공인인증서를 이용하려면 이용자가 반드시 추가프로그램을 설치해야만 한다.

Table 1. 공인인증서 유출현황[3]

연도	12년	13년	14년	15년	16년	합 계
건수	8	8,710	41,733	22,796	6,850	80,097

이런 특징은 특정 저장매체에 저장된 공인인증서의 정보들은 공격자에게 탈취당해 사용자가 피해를 입을 수 있는 취약점에 항상 노출되어 있고 공인인증서 유출 위협이 지속적으로 야기되고 있음을 확인할 수 있다. 실제 한국인터넷진흥원에서 공개한 자료를[3] 보면 2012년 8건에 불과했던 공인인증서 유출 건수는 2013년 8,710건으로 전년 대비 천 배 이상 증가했으며, 2014년에는 4만 1,733건으로 폭증했고, 2016년에는 6,850건이 유출됐다.

Table 1.과 같은 공인인증서의 지속적인 유출을 막기 위해서 악성코드 및 스미싱(Smishing) 유포지를 탐지해 사전 차단하거나 백신 및 방화벽 설치 및 업데이트 등의 예방방법을 사용하거나 나아가 생체인식을 병행한 이중인증(Two-factor)방식을 사용하기도 한다. 이중 인증방식이란 인증 프로세스에 별도의 보호계층을 추가하는 방법으로 사용자는 비밀번호 이외에 정보를 식별하는 두 번째 부분을 제공하는 방식의 인증방법이다. 현재는 편리성을 강조하여 생체정보인 지문이나 홍채를 이용하는 방법이[4] 대중화 되어 있는데 이러한 생체인식 정보는 생체인식의 정보 자체가 탈취당하면 재발급을 할 수가 없다는 치명적인 단점이 존재한다.

또한, 현재 공인인증서의 안전한 관리 및 보관을 위해 PC의 CPU칩에 공인인증서를 보관하는 방법(Identity Protection Technology)과 TrueCrypt 등으로 암호화시킨 볼륨에 공인인증서를 넣어놓으면 컴퓨터 해킹이나 USB 메모리 분실 등으로 타인에게 유출되는 것을 방지할 수 있는 방법은 현재 개발이 중단된 상태이다.

III. 제안방식

본 연구에서 제안하고자 하는 방식은 공인인증서 파일이 존재하는 PC의 MAC 주소와, USB의 Container ID를 통해 개인키 파일을 암

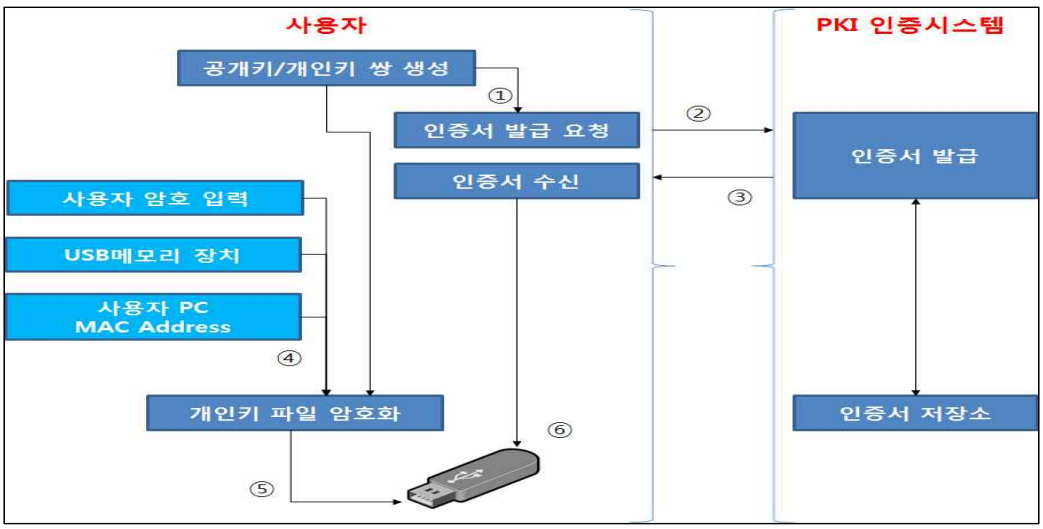


Fig 1. 사용자 인증서, 개인키 등록 절차

호화하는 기존의 연구[5] 방식과는 다르게 지정된 PC를 추가하여 안전하게 보관하는 Two-factor 공인인증서 보관방법을 제안하고자 하며 그에 따른 절차는 Fig 1.과 같다.

먼저, 사용자 PC에서 공개키와 개인키 한 쌍을 생성한다. 그 후, 생성된 공개키와 사용자 정보를 인증서 요청 양식에 따라 작성하여 PKI 인증시스템에 사용자의 인증서 발급을 요청하고, 사용자는 이 인증서(User_Cert)를 수신한다.

그 후, 개인키 암호용 패스워드와 USB 메모리의 Container ID값, PC의 MAC 주소 값을 조합하여 해시 값(User_Key)을 생성한다.

$$User_Key = H(UserPW \parallel ContainerID \parallel MAC\ Address)$$

위와 같은 식으로 생성된 해시 값과 인증서에서 수신한 개인키를 이용하여 USB 메모리에 저장하여 활용한다.

이와 같은 방식은 개인키를 생성할 때 단순히 사용자의 암호만을 통해서 암호화하는 방식이 아니라 USB와 PC의 고유 값을 이용한 다음 다시 해시 값으로 저장하는 인증방식이므로 해킹으로 인해 개인키 유출 등이 발생해도 해시 함수의 특성으로 인해 개인키의 복호화가 불가능하다. 또한, 두 가지의 다른 저장매체의 고유한 값을 요구하기 때문에 USB를 분실하거나 PC의 하드디스크가 위협 당하더라도 개인키의 안정성을 보장받을 수 있게 된다. 실제 인증서를 사용하는 사용자도 등록된 USB와 등록된 PC에서만이 정상적으로 개인키를 복호화할 수 있게 된다.

또한, 본 연구에서 제안한 방식의 객관적인 안전성 평가를 위해 기존에 안전하다고 평가받거나 널리 활용되고 있는 USB 저장방식[6], USIM 스마트 인증방식, OTP 키 관리방안 방식을[7] 본 연구에서 제안하는 시스템과 비교하여 평가한 결과는 Table 2.와 같다.

본 연구에서 제안하는 인증방식은 편리성을 생각하여 USB에 저장하는 방식은 동일하나 등록된 USB에 한정되며, 추가 이용요금이 발생하지 않는다. 또한, 해시값을 이용하여 탈취된 데이터를 재사용할 수 없으며 개인키 노출에 따른 재사용도 방지 할 수 있어 최상의 보안성을 제공할 수 있다.

Table 2. 저장 방식에 따른 비교 평가

구분 \ 방안	기존 방식	USIM 스마트 인증	OTP기반 키 관리	**제안방식**
저장매체	USB	스마트폰의 USIM	USB	**등록된 USB**
이용요금	무료	유료	무료	**무료**
인증요소	PW	PW + USIM 카드 암호	PW + OTP	**PW + USB 컨테이너 ID + PC MAC Address**
탈취된 데이터 재사용 방지	X	O	O	**O**
개인키 PW노출에 따른 재사용 방지	X	X	O	**O**
보안성	중	상	상	**상**

IV. 결론 및 향후연구

공개키 인증방식(PKI)은 지금까지도 훌륭한 알고리즘으로 평가받는다. 그럼에도 불구하고, 악의적인 공격자들에 의해 공인인증서와 비밀번호에 대한 보안사고가 지속적으로 발생하고 지속적으로 공인인증서의 개인키와 복호화 패스워드의 보관과 관리의 중요성은 지속적으로 강조되고 있다. 본 연구에서는 USB 메모리와 PC의 고유 값을 활용한 Two-factor 인증방식을 제안하여 개인키 파일의 안전한 관리방안을 제안하였다. 지금은 PC에 적용하는 인증방법으로 제안하였지만 추후 연구에는 스마트폰에 적용할 수 있는 방안도 모색할 필요가 있을 것이다.

[참고문헌]

[1] 한국인터넷진흥원, 17년도 대국민 전자서명 이용실태조사, 2017.12

[2] B. Kaliski, PKCS #8: Private-Key Information Syntax Standard V1.2, RSA Laboratories, 2008.

[3] 보안뉴스, 공인인증서 유출현황, 검색일 : 2018.11.1.

[4] 김선종, 스마트폰 환경에서 공인인증서 사용시 소유 및 생체인증 연동 방법, 정보보호학회지 제25권 6호 pp13-17, 2015.

[5] 김선주 조인준, USB 메모리의 컨테이너ID를 이용한 PKI 기반의 개인키 파일의 안전한 관리 방안, 한국콘텐츠학회논문지 15권 제10호 pp.607-615, 2015.

[6] 김선영 김선주 조인준, 이동저장매체를 활용한 패스워드기반 사용자인증 강화방안, 한국콘텐츠학회논문지 제4권 제1호 pp.533-540, 2014(11).

[7] 김미혜 서세영, 모바일PKI 기반한인증구조, 한국콘텐츠학회논문지 제권 제호 pp.67-75, 2004(3).

[8] B. Kaliski, PKCS #5, Password Based Cryptography Standard V2.1, RSA Laboratories, 2000.

인텔 PT를 활용한 효율적인 수행 흐름 무결성에 대한 연구

서지원*, 백세현*, 방인영*, 김기열* 백윤흥*
*서울대학교 정보보호학과

A Study on Efficient Control-Flow Integrity Using Intel PT

Ji-Won Seo*, Se-Hyun Baek*, In-Young Bang*, In-Young Bang*, Gi-Yeol Kim*, Yun-Heung Paek*
*Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University

요 약

운영체제 커널은 전체 시스템에서 가장 중요한 시스템 소프트웨어이자 전체 시스템의 Trusted Computing Base(TCB) 이기 때문에 시스템 전체의 안전성 확보를 위해 반드시 보호되어야 한다. 그러나 커널은 여전히 다양한 공격에 노출되어 있고 지속적으로 취약점들이 보고 되고 있다. 본 논문에서는 수행 흐름 무결성 연구에 대한 기존 연구들을 소개한 다음 커널에 수행 흐름 무결성 기법을 적용시킬 수 있는 방안에 대해 제시할 것이다.

I. 서론

운영체제 커널은 전체 시스템에서 가장 중요한 시스템 소프트웨어 이자 전체 시스템의 Trusted Computing Base 이기 때문에 시스템 전체의 안전성 확보를 위해 반드시 보호되어야 한다. 그러나, 컴퓨터 시스템의 안전성에 대한 연구가 계속하여 지속 되어 왔음에도 불구하고 커널은 여전히 다양한 코드 재사용 공격 문제(return-into-libc, JIT-ROP, JOP..등) 에 노출되어 지속적으로 취약점들이 보고되고 있고, 이는 앞으로도 계속하여 주요 사이버 위협으로 남아 있을 것이다. 공격자들은 항시 커널을 손상되게 함으로써 각 응용프로그램 간 고립을 무력화하고 동시에 커널 내 존재하는 중요한 정보를 탈취하고자 한다. 커널은 크기가 큰 코드이므로 취약점을 완전히 제거하는 것은 불가능하다. 그렇기에 현재까지 알려진 취약점들이 모두 제거되었다고 하더라도 공격자가 비슷한 형태의 취약점 공격을 수행할 수 없다는 보장을 하는 것도 불가능하다.

기존에 커널에서 이러한 코드 재사용 공격에 대응하기 위해서 잘 알려진 기법으로 수행 흐름 무결성 기법이 있다. 수행 흐름 무결성은 응용프로그램이 정해진 수행 흐름 그래프만을 따라서 흐름이 진행되도록 보장하는 기법이다. 이 기법은 소프트웨어 방식으로 구현되는 경우 성능문제를 무시할 수 없기 때문에 최근 하드웨어의 도움을 받아 수행 흐름 무결성을 보장하는 연구들이 나오고 있다.

본 연구에서는 Intel에서 제안한 Processor Trace[1] 라는 하드웨어를 활용하여 수행 흐름 무결성을 적용 시킨 기존 논문들에 대해 살펴볼 것이다. 기존 연구들은 보통 응용프로그램에 대한 수행 흐름 무결성 검증을 하였기 때문에 본 논문에서는 기존 연구들을 소개한 다음 커널에 수행 흐름 무결성 기법을 적용 시킬 수 있는 방안에 대해 제시할 것이다. 이를 위하여 먼저 2장 Background에서 관련 개념들에 서술

한 후 기존 연구에서 제시한 방어법들을 소개한 뒤 결론을 제시하도록 하겠다.

Packet Name	Packet Description
Paging Information Packet (PIP)	현재 사용 중인 Paping 정보를 담고 있는 CR3 register의 변화가 있을 때마다 발생하는 packet이다. 특히 가상머신 커널의 수행과 호스트 커널의 수행을 구분할 수 있는 NR(non-root) bit을 포함하고 있다.
Target IP Packet(TIP)	indirect branch, exceptions, interrupts 가 발생할 경우의 목적지 주소(target address)를 기록하는 패킷이다.
Flow Update Packets(FUP)	interrupt와 exception이 발생한 경우의 source address를 기록하는 패킷이다.
Packet Stream Boundary (PSB)	메모리 영역에 일정량의 패킷들이 쌓이면 발생하는 패킷이다.

표 1. Intel PT packet들과 이에 대한 설명

II. Background

2.1 Intel Processor Trace

Intel PT는 최근 Intel 프로세서의 수행 흐름 정보를 적은 오버헤드로 추출할 수 있도록 하기 위해 제공된 하드웨어 기능이다. Intel PT는 수행 흐름 정보를 여러 패킷의 형태로 메모리 버퍼에 저장하게 된다. 표 1은 이러한 패킷 가운데 관련 연구들이 수행 흐름 분석을 위해 사용하는 패킷을 정리한 것이다. 한편 Intel PT는 사용 목적에 따라 패킷 생성 관련 설정이나 필터 관련 기능도 제공한다. 이와 관련한 자세한 내용은 관련 매뉴얼에 정리되어 있다[7].

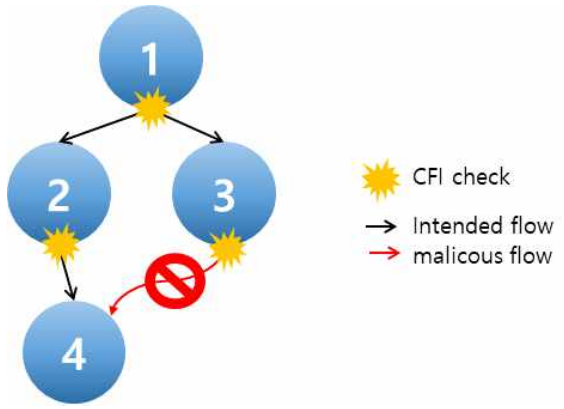


그림 1 수행 흐름 무결성

2.2 수행 흐름 무결성[6]

수행 흐름 무결성이란 감시하는 프로그램이 적법한 수행 흐름으로 실행되고 있는지를 검사하는 것이다. 응용프로그램에 대한 기존 수행 흐름 분석 연구는 대상 프로그램의 모든 간접 분기문에 코드 수정을 통해 무결성 검증을 하는데 이 기술은 상당한 오버헤드를 초래한다. 커널에 대한 기존 수행 흐름 분석 연구는 보통 코드 혹은 바이너리 수정을 통해 특정 코드를 수행할 때의 정보를 추출하여 응용프로그램과 달리 커널의 일부 메모리에 대한 무결성을 검증하였다.

III. 수행 흐름 분석에 관한 연구

3.1 Intel Processor Trace를 활용한 흐름 분석에 관한 연구

3.1.1 PT-CFI [2]

backward-edge 에 대한 수행 흐름 검증으로 Intel Processor Trace를 활용하여 적은 오버헤드로 수행 흐름을 추출하였다. PT 는 본래 오프라인에서 소프트웨어 디버깅 혹은 퍼포먼스 분석이 목적인데 PT-CFI 는 이를 온라인 보호 목적으로 사용하였다. 해당 연구에서 제시하는 방안은 CFI policy를 enforce 하기 위해 T

IP 패킷에 대한 sequence graph를 그렸다. 즉 모든 PT 패킷을 보고 동적으로 CFG를 그리는 것은 오버헤드가 많이 드는 접근 방식이므로 TIP 패킷만 봄으로써 오버헤드를 줄일 수 있고 indirect control flow transitions을 정확하게 capture 할 수 있다. TIP 그래프는 프로세서 트레이스의 패킷을 파싱하여 만들고 이후 알려지지 않는 경로에 대해서는 심층 검사를 거친 다음 TIP 그래프에 해당 경로를 추가하거나, 만일 잘못된 경로로 판단이 된 경우 실행 중인 프로세스를 강제 종료한다.

3.1.2 GRIFFIN [3]

GRIFFIN 은 Intel PT를 활용하여 수정하지 않은 바이너리에 대해 수행 흐름 무결성을 enforce 한 연구다. 이 논문에서 제안하는 challenge 는 Intel PT 가 기록해주지 않는 call sites 정보를 알아야 하는 것인데, 해당 연구는 바이너리를 디스어셈블하여 trace buffer를 해석하였다. 디자인을 최적화 하기 위해 Basic Block 정보를 빠르게 조사하기 위한 자료구조 디자인을 제안하였고 Idle 코어로 디스어셈블, 트레이스 처리, CFI enforcement를 병렬적으로 처리하였다. GRIFFIN은 3.1.1 PT-CFI 와 달리 Forward/backward edge 에 대해 Coarse-grained CFI, Fine-grained CFI, Stateful CFI를 적용하였다. Coarse-grained CFI 는 TIP 패킷에 있는 목적지 주소를 통해 코드 위치가 합법적인지 저장하는 정책 페이지에 합법적인 경우 1, 합법적이지 않은 경우 0을 기록하였다. Fine-grained CFI 는 출발지를 행, 목적지를 열로 하는 행렬을 구성한다. 이때 출발지 주소를 알기 위해 바이너리를 디스어셈블하여 패킷을 추적한다. Stateful CFI 는 수행 흐름을 연속적으로 처리하기 위해 콜고 리턴을 실행 순서대로 처리하는 방식이다.

3.1.3 PITTYPAT [4]

PITTYPAT은 적은 오버헤드 하드웨어 기반 모니터링으로 points-to 분석을 실시간에 한다. 해당 연구에서는 주요기술로 크게 두 가지를 제안한다. 첫 번째로 Intel PT 로부터 모든 선택된 수행 흐름 타겟을 수집하는 event-driven 커널 모듈을 이용한다. 두 번째로는 abstract-interpretation-based incremental points-to 분석으로 최적화를 위해 수행 흐름 히스토리가 아니라 어떠한 Basic Block 이 실행되었는지만 알면 되고 이는 CFG 를 통해 해결하였다. 두 번째로 기존 연구는 정적으로 points-to-analysis를 하여 대기시간이 크다는 단점이 있는데 PITTYPAT 은 코드 포인터의 points-to 정보만 알면 되기 때문에 이전 연구의 문제를 해결하였다.

3.2 커널에 대한 수행 흐름 분석에 관한 연구

3.2.1 FINE-CFI

커널에 context-sensitive 및 field-sensitive 포인터 분석을 통해 Fine-grained CFG를 construct 한 연구다. 본 기법은 ret2usr 와 커널 코드 재사용 공격을 방어하는 것을 목적으로 하고, Fine-grained CFG를 얻기 위해 모든 indirect call 과 jmp 에 대해 point-to set를 얻어야 한다. struct location vector를 통해 indirect function calls의 타겟을 추론하여 indirect call 및 jmp 명령어의 target 수를 굉장히 감소시키고 결과의 정확도도 향상시킨다. 또한 하이퍼바이저 기반 접근 방식으로, 하이퍼바이저에 런타임 문맥 정보를 백업하여 현재 커널 스택과 하이퍼바이저에 있는 값을 비교하여 값을 검증한다.

IV. 인텔 PT를 활용한 커널 무결성 검증

기존에 커널에서 이러한 코드 재사용 공격에 대응하기 위해, CFI가 가장 저명한 기법이고 이를 제안하는 논문으로 Fine-CFI, KCoFI 가 있다. 이들 논문들은 수행 흐림 변화에 대한 모니터링을 위해 소프트웨어 기반 기술을 제안한다. 그러나 커널 수행에 끊임없이 개입해야 하기 때문에 오버헤드가 크다는 단점이 있다.

따라서, 본 논문에서는 이러한 성능문제를 해결하여 보다 효율적으로 커널에 CFI를 보장할 수 있는 방안을 제안하는데 이때 솔루션 키는

인텔에서 제안한 PT 라는 하드웨어를 이용하는 것이다. PT를 이용하면 5% 미만의 성능 오버헤드로 코드 수정 없이 수행 흐름을 추출할 수 있다. 따라서, 많은 논문들은 이를 이용하여 사용자 레벨에서 CFI를 적용하려는 시도들을 했다. 하지만 이들 연구들은 커널에 그대로 솔루션을 적용하는 것은 쉬운 문제가 아니다.

첫 번째로 커널은 런타임 프로세스 실행 수에 비례하여 호출되기 때문에 trace 정보가 폭발적으로 증가하여 이에 따른 오버헤드가 커질 수 있다. 두 번째로는 커널의 방어를 위해 PT를 이용할 경우, PT가 커널 위에서 동작하기 때문에 공격자에 의해 PT 관련 값들이 변조 될 수 있다.

첫 번째 문제는 우리의 관찰에 의하면 커널은 제한된 인터페이스인 시스템 콜을 통하여 호출되기 때문에 반복적인 CFI 검증을 할 필요가 없다. 두 번째 문제를 해결하기 위해서는 커널 내에서 PT를 독점적으로 control 하고 trace 하기 위에 커널 내부에 isolate 된 영역을 만들어 내부에서만 PT를 돌게 하면 공격자로부터의 접근을 막을 수 있다.

V. 결론

커널은 전체 시스템에서 가장 중요한 시스템 소프트웨어이자 전체 시스템의 TCB이다. 그러나 커널은 여전히 다양한 코드 재사용 공격 문제에 노출되어 있고 앞으로 계속하여 주요 사이버 위협으로 남아 있을 것이다. 기존 논문들은 control-flow 변화에 대한 모니터링을 위해 소프트웨어 기반 기술로 커널 수행에 끊임없이 개입해야 하기 때문에 오버헤드가 크다는 단점이 있다. 본 논문에서는 이러한 성능 문제를 해결하여 보다 효율적으로 커널에 CFI를 enforce 할 수 있는 방안을 제안한다.

ACKNOWLEGEMENT

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2016-0-00078, 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발)

[참고문헌]

[1] Guide, Part. "Intel® 64 and IA-32 Architectures Software Developer's Manual." Volume 3B: System programming Guide, Part 2 (2011).

[2] Gu, Yufei, et al. "PT-CFI: Transparent backward-edge control flow violation detection using intel processor trace." Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. ACM, 2017.

[3] Ge, Xinyang, Weidong Cui, and Trent Jaeger. "Griffin: Guarding control flows using intel processor trace." ACM SIGOPS Operating Systems Review 51.2 (2017): 585-598.

[4] Ding, Ren, et al. "Efficient protection of path-sensitive control security." 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC: USENIX Association. 2017.

[5] Li, Jinku, et al. "Fine-CFI: Fine-Grained Control-Flow Integrity for Operating System Kernels." IEEE Transactions on Information Forensics and Security 13.6 (2018): 1535-1550.

[6] Abadi, Martín, et al. "Control-flow integrity." Proceedings of the 12th ACM conference on Computer and communications security. ACM, 2005.

코드 재사용 공격에 대한 조망

방인영, 김현준, 김기열, 백윤흥*
*서울대학교 전기정보공학부

A Survey on Code Reuse Attack

In-Young Bang, Hyun-Jun Kim, Ki-Yeol Kim, Yun-Heung Paek*
*Electrical and Computer Engineering, Seoul National University.

요 약

본 논문에서는 코드 재사용 공격과 방어 기법의 역사적 흐름을 살펴보고 향후 연구 방향에 대해 논한다. 코드 재사용 공격을 막기 위해 제안된 기법들은 상당한 효과를 거두었는데, 그럼에도 코드 재사용 공격이 아직 위협적임을 Newton이라는 공격 프레임워크를 통해 알아본다. Newton이 어떻게 방어기법을 파훼하여 공격을 성공시키는지를 살펴보고, 그 성취를 바탕으로 코드 재사용 공격의 향후 연구 방향을 조망하고자 한다.

1. 서론

2007년 Shacham[1]이 반환 지향 프로그래밍 공격을 공식화한 것이 코드 재사용 공격에 대한 최초의 체계적인 공식화였다. Victim binary code의 구조를 분석하고 가젯의 위치를 locate한 뒤 가젯을 chain하여 공격을 craft한다는 매우 간단한 원리다. 이 논문은 굉장히 영향력이 있었으며, 2018년 11월 기준 1000회 이상의 인용을 기록하고 있다.

10년이 지난 지금에도 공격 모델 자체는 바뀌지 않았다. Shacham 이후 코드 재사용 공격과 방어에 대해 수많은 논문이 발표되었고, 이러한 노력으로 인해 프로그램에서 수 백만개씩 발견되었던 가젯의 수가 수 십개로 줄게 되었다. 자연스럽게 코드 재사용 공격이 더 이상은 어렵다고 생각하기 쉬우나, Veen[2]은 런타임 가젯 탐색 프레임워크인 Newton을 제안하면서 여전히 코드 재사용 공격이 현실적으로 막기 어려움을 제시하고 있다.

방어하는 입장과는 달리 공격자로서는 가젯의 수나 튜링 불완전성이 공격의 불가능을 담보하지 않으며, execve나 mprotect와 같은 주요한 함수를 조작 가능한 인자를 이용해 호출할 수만 있다면 공격자가 원하는 목표를 달성할 수 있다고 지적한다. Veen은 Newton을 통해 '가젯을 어떻게 연결하여 방어를 무력화하고 자유롭게 공격 코드를 작성하는가?'를 고민하는 대신, '어떤 메모리 값을 수정하면 프로그램을 제어할 수 있는가?'라는 근원적인 물음에 답하는 것이 코드 재사용 공격 성공의 열쇠임을 말하고 있다. 본 논문에서는 코드 재사용 공격과 방어 기법의 역사적 흐름을 살펴보고, Newton이 어떻게 방어기법을 파훼하여 공격을 성공시키는지를 살펴보고자 한다. 그리고 Newton의 성취를 바탕으로 코드 재사용 공격의 향후 연구 방향을 조망하고자 한다.

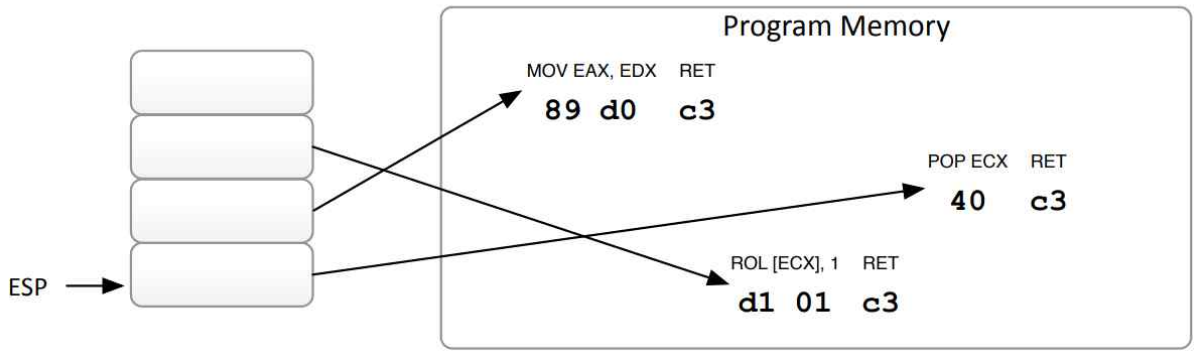


그림 1. 반환 지향 프로그래밍을 위한 스택 조작

2. 본론

코드 재사용 공격은 4개 이상의 명령어로 이루어진 코드 조각인 가젯을 공격자가 임의적으로 연결하여 원하는 동작을 수행한다. 이를 가능케 하기위해 공격자는 대상 프로그램의 코드를 정적 분석하여 가용 가젯을 확보한 뒤, 오버플로우 공격을 이용하여 스택이나 힙에 가젯의 주소를 순서대로 저장한다. 마지막으로 제어 흐름을 탈취하여 가젯을 순차적으로 실행시킴으로써 공격자는 목적을 달성하게 된다.

2.1 Ret2libc

ROP 이전에 가장 쉽고 강력했던 공격은 코드 삽입 공격이었는데, 공격자가 원하는 코드를 공격 대상에 주입하여 실행시키는 공격이다. 이 공격은 데이터 영역에는 쓰기 권한만을, 코드 영역에는 실행 권한 만을 부여하는 W xor X 기법에 의해 막히게 된다. ROP는 코드를 삽입하지 않고, 대상 프로그램의 코드를 사용하므로 W xor X와 같은 데이터 실행 보호를 간단히 무력화한다. ret2libc 공격을 생각해보면 ROP 공격이 지닌 파괴력을 알 수 있다. Libc는 유닉스 운영체제의 system이나 execve와 같은 보안적으로 중요한 시스템 콜을 비롯하여 대부분의 커널 함수를 포함한다. ret2libc 내의 가젯들을 이용하면 공격자는 Turing complete하게 원하는 공격을 수행할 수 있다.

2.2 반환 지향 프로그래밍

코드 재사용 공격에서 가젯을 이를 때 분기 명령어중 return 명령어를 사용하는 공격을 반환 지향 프로그래밍이라고 하며 앞서 언급한 ret2libc는 ROP의 일종이다. 상술하자면, 공격자가 return으로 끝나는 가젯을 찾아 순서를 정하고, 해당하는 순서대로 스택에 쌓는다. 스택 포인터를 첫 가젯에 놓게 되면 호출 규약에 의해 return이 될 때마다 순서대로 가젯을 실행하게 된다. 이 과정을 나타내면 [그림 1]과 같다.

2.3 Shadow Stack

이를 막기 위해 등장한 대표적인 방어 기법으로 shadow stack이 있다. Shadow stack에 대한 정보를 공격자는 알 수 없으며, 함수가 호출되어 스택에 반환 주소가 스택에 저장될 때, 같은 값이 shadow stack에도 push되게 된다. return 명령어에 의해 스택에서 반환 주소를 꺼낼 때, shadow stack에서도 pop하여 값을 비교하면, 원래 돌아가야 할 주소가 맞는 지를 확인할 수 있다. 만약 공격자가 스택을 조작하여 임의의 주소로 return을 하고자 할 경우 shadow stack을 통해 이를 확인할 수 있으며, 따라서 반환 지향 프로그래밍 공격에 대한 내성을 가지게 된다.

2.4 점프 지향 프로그래밍

Return 대신 jump 명령어를 이용한 코드 재사용 공격을 점프 지향 프로그래밍이라고 한다. 반환 지향 프로그래밍에서는 가젯을 스택에 순서대로 넣으면 호출 규약에 의해 순차적으로 실행이 되었지만, 점프 지향 프로그래밍에서는 이를 위해 디스패처라고 불리는 가젯이 필요하다. 디스패처 가젯은 아키텍처에 따라 다르게

구성되며, 점프에 사용할 주소가 담긴 레지스터 값을 word 크기만큼 증가시키면서 해당 주소로 점프를 하게 된다. 공격은 원하는 가젯의 주소를 순차적으로 메모리에 저장한 뒤, 디스패처 가젯에서 사용하는 레지스터에 첫 가젯의 주소를 넣고 프로그램을 수행하는 방식으로 이루어진다.

2.5 Control Flow Integrity

이를 막기 위해 분기 명령어의 적법한 분기 대상 주소를 규정하고, 이 주소들에 대해서만 분기하도록 제한하는 branch regulation이라는 방어 기법이 제안되었다. 적법한 분기에 대한 몇 가지 규칙을 정하고, 해당 규칙이 프로그램 내에서 준수되는지를 감시한다. 유사하게, function pointer(forward indirect branch), return(backward indirect branch)과 같은 모든 간접 분기 명령어에 대해서 적법한 분기 대상 주소를 제한하여 코드 재사용 공격을 방어하는 기법인 Control Flow Integrity(CFI) 방어 기법에서는 정적 분석을 이용하여 control flow graph를 작성하고, 이에 기반하여 branch instruction의 대상을 제한하여 반환 지향 프로그래밍과 점프 지향 프로그래밍에 대한 방어력을 가진다[3].

2.6 Informaition Hiding

코드 재사용 공격이 프로그램에 대한 정적 분석에 기반한다는 통찰을 바탕으로, 프로그램 분석을 어렵게 만들어 공격에 필요한 비용을 극대화함으로써 공격의 효용을 떨어트리고자 하는 경제적 시도가 있었다. 먼저 주소 공간 배치 랜덤화(Address Space Layout Randomization)는 코드와 데이터의 시작 주소를 섞어서 공격자가 메모리를 오염시킬 수 있더라도, 가젯을 파악하는 것을 방해한다. 최초의 ASLR은 로드 타임에 주소 공간 배치를 랜덤화하였지만, 공격자가 런타임에 랜덤화에 대한 정보를 파악할 수 있기 때문에, 런타임에 가젯을 찾고 반환 지향 프로그래밍을 시도하는 JIT 반환 지향 프로그래밍이 등장하였다. 이에 대응하기 위해 런타임 ASLR 기법이 등장하게 되었고, 주기적으로 주소를 섞는 Re-randomization 기법도 제시되었다.

2.7 Code Pointer Integrity

코드 포인터 무결성은 공격자가 코드 포인터를 오염시켜 contrl flow를 하이재킹하는 것을 막기 위해 모든 코드 포인터를 보호하는 기술이다. 런타임 분석을 통해 공격자가 포인터의 값을 변경하고 잘못된 위치로 점프하기 위해 메모리를 오염하는 것을 방지하는데, 오버헤드가 크기 때문에 완화된 기법인 코드 포인터 분리(Code Pointer Separtion)이 제시되었다.[4]

2.8 Newton

Veen이 제시한 런타임 가젯 탐색 프레임워크인 Newton은 지금까지 언급한 모든 방어기법을 고려하여 가젯을 도출해내고, 동적 오염 분석을 통해 알아낸 취약한 메모리로부터 코드 재사용 공격을 성공시킨다. Control Flow Integrity, Information Hiding, Re-randomization, Pointer Integrity의 네 가지 분류의 방어기법을 제약조건으로 모델링하여 state-of-the-art 방어기법을 파훼할 수 있다.

Newton의 개괄은 그림 2와 같다. 이진 파일과 라이브러리를 입력으로 받아 동적 오염 분석을 수행하여 조작가능한 메모리 영역을 확인하고, 방어 기법의 모델링을 바탕으로 정적 분석과 동적 분석을 수행하여 활용 가능한 가젯을 자동으로 도출해낸다.

제약조건은 쓰기 제약조건과 대상 제약조건으로 나뉜다. 쓰기 제약조건은 공격자가 오염 가능한 메모리 영역을 명시하며, 함수 포인터, 데이터 포인터, int나 str과 같은 비포인터 값을 대상으로 한다. 대상 제약조건은 공격자가 조작 가능한 호출 사이트와 호출 가능한 피호출자(callee)에 대한 가정을 나타낸다. 쓰기 제약조건과 대상 제약조건을 조합하면 현존하는 거의 모든 방어기법을 모델링할 수 있으며, 방어를 고려하여 가젯을 탐색하는 것이 가능하다.

가젯은 단순히 코드 파편이 아닌, 어떤 주소나 인자에 의해 호출 사이트(callsite)를 조작 가능한지, 그리고 해당 호출 사이트에서 어떤 함수를 호출할 수 있는지를 나타내는 형식을 가진다. 동적 오염 분석을 통해 바이트 단위로 메모

리 종속성을 추적하고, 제약 조건을 고려하면 가젯에 대한 풍부한 정보를 얻는 것이 가능해진다.

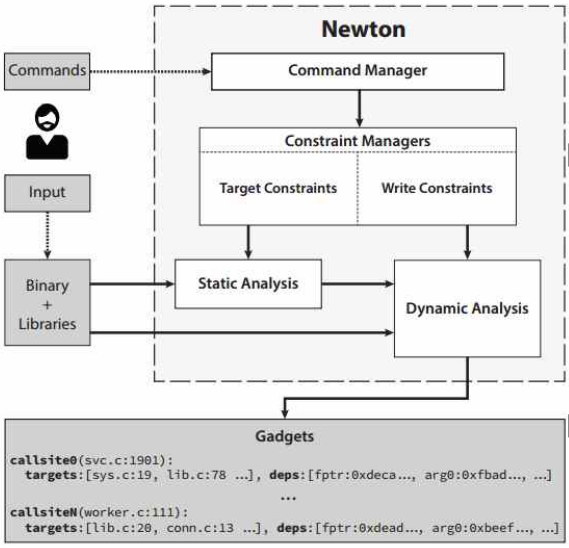


그림 2. Newton의 개괄

가젯의 수가 줄어 공격이 어려워졌다는 기존의 관념과는 동적 분석을 공격에 이용하면 코드 재사용 공격을 성공적으로 수행할 수 있다는 사실과, 단 하나의 가젯으로도 공격자가 원하는 행동을 할 수 있다는 것을 실험적으로 도출한다.

3. 결론

본 논문에서는 방어 기법의 역사적 흐름을 살펴보고, Newton이 어떻게 방어기법을 파훼하여 공격을 성공시키는지를 살펴보았다. Newton의 실험 결과를 참고하면, 각각의 방어기법이 가젯의 수를 크게 줄일 수 있지만, 이것이 공격이 불가능함을 담보하지는 않는다는 사실을 알 수 있었다. 개별적인 방어기법으로는 결국 코드 재사용 공격을 완전히 차단하기 어려우므로 기존의 방어기법들을 조합할 필요성이 있으며, 이를 위해서는 방어기법의 성능 오버헤드 문제가 매우 중요하다는 것을 알 수 있다. 코드 재사용 공격은 여전히 유망한 분야이며, 결론적으로 방어기법을 어떻게 조합하고, 성능 오버헤드를 어떻게 줄이느냐가 연구의 관건이 될 것이다.

4. Acknowledgement

[참고문헌]

[1] Hovav Shacham, The Geometry of Innocent Flesh on the Bone: Returninto-libc Without Function Calls (on the x86), CCS, October, 2007.

[2] Victor van der Veen, Dennis Andriesse, Manolis Stamatogiannakis, Xi Chen, Herbert Bos, Cristiano Giuffrdia, The Dynamics of Innocent Flesh on the Bone: Code Reuse Ten Years Later, CCS, October, 2017.

[3] Martín Abadi, Mihai Budiu, Ulfar Erlingsson, Jay Ligatti, Control-flow integrity, CCS, November, 2005.

[4] Volodymyr Kuznetsov, László Szekeres, Mathias Payer, George Candea, R. Sekar, Dawn Song, OSDI, October, 2014.

정적 경로를 이용한 바이너리 프로그램의 함수 식별 방법*

김영철, 조은선
충남대학교 컴퓨터공학과

Identification Method for Function of Binary Program Using Static Path

Yeongcheol Kim*, Eun-Sun Cho*
Dept. of Computer Science & Engineering, Chungnam National University.

요 약

악성 행위가 포함된 바이너리 프로그램을 빠르게 탐지하는 것은 중요하다. 이러한 프로그램의 소스코드를 얻을 수 없기 때문에 바이너리를 분석하는 능력을 갖는 것이 중요하다. 본 논문에서는 윈도우 실행 파일 바이너리를 대상으로 구성하고 있는 함수 정보를 추출하여 비슷한 함수를 식별하는 방법을 연구한다. 함수의 제어 흐름 그래프에서 탐색 가능한 정적인 실행 경로를 이용하여 비슷하지만 다른 함수를 식별할 수 있도록 한다.

I. 서론

악성 행위가 포함된 바이너리 프로그램을 빠르게 탐지하는 것은 중요하다. 실제로 악성 행위를 수행하는 코드는 프로그램의 일부분이기 때문에, 분석 대상이 되는 코드를 형식적으로 변형하고 특징을 추출하여 악성 행위를 수행하는 코드를 탐지할 수 있다. 기본적으로 분석 대상의 소스코드를 얻을 수 없기 때문에 바이너리를 분석하는 능력을 갖는 것이 중요하다.

바이너리 프로그램을 소스코드 없이 완전하게 분석하는 것은 어렵다. 다행히, 실행 파일 바이너리의 code 영역은 함수로 구성되어 있으므로, 프로그램을 함수 단위로 분해하고 구성하고 있는 함수를 분석하여 비슷한 다른 함수를 탐지할 수 있다면 프로그램을 분석하는 것이 용이해진다.

본 논문에서는 함수를 식별할 때 윈도우 실행 파일 바이너리를 구성하는 함수를 대상으로 실행 흐름을 반영할 수 있도록 함수의 시작 지점부터 끝 지점까지 도달 가능한 정적인 실행 경로와 경로를 결정하는 분기 조건을 이용한 함수 탐지 방법을 연구한다.

II. 관련 연구

바이너리 코드를 분석하여 식별하는 방법에 대한 연구는 다양한 방향으로 진행되고 있다.

먼저, 바이너리 프로그램의 함수를 구성하는 명령어의 연산자 서열 비교를 통해 같은 함수를 탐지하는 방법이 있다[1][2]. 단순히 연산자 서열 비교이기 때문에 함수의 조건 분기에 의한 실행 흐름을 반영하기 어렵다. 의미가 상반된 두 함수의 경우, 연산자 서열이 매우 비슷하기 때문에 [1][2]의 방법으로는 결국 다른 함수

* "본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2018-2016-0-00304)

라는 것을 인지하지 못한다. Fig. 1.은 2개의 인자를 넘겨받아 상반된 처리를 하는 두 함수의 구성 명령어이다. 왼쪽은 더 작은 값을 반환하는 행동을, 오른쪽은 더 큰 값을 반환하는 행동을 하는데, 두 함수를 비교하면 5번 라인의 미묘한 차이를 탐지할 수 없게 된다.

다른 방식으로는 바이너리 함수 내에 존재하는 strand 집합을 이용한 식별 방법이 있다[3]. strand란 특정 value를 처리하는 명령어의 집합이다. Fig. 2.는 Fig. 1.의 코드를 기본 블록 단위로 구분하고 조건문과 분기문을 제거한 것이다. [3]에서는 이와 같은 처리를 한 후, strand를 추출하는데 육안으로도 완전히 같기 때문에 마찬가지로 두 함수를 식별할 수 없게 된다. 또한, strand와 유사한 접근 방법인 tracelet[4]과 통계적인 방법[5]도 식별 과정에서 조건식을 배제하여 상반된 결과의 두 함수가 같게 되는 경우를 올바르게 다루지 못한다.

따라서, 비교 대상이 되는 두 함수가 완전히 반대로 작동하는 코드를 가질 때 식별하기 위한 연구가 필요한 것으로 보인다.

```
 1.    push   ebp
 2.    mov    ebp, esp
 3.    mov    eax, [ebp+0x8]
 4.    cmp    eax, [ebp+0xc]
 5.    jge    L0
 6.    mov    eax, [ebp+0x8]
 7.    jmp    L1
 8.    jmp    L1
L0:
 9.    mov    eax, [ebp+0xc]
L1:
 10.   pop    ebp
 11.   ret
```

```
 1.    push   ebp
 2.    mov    ebp, esp
 3.    mov    eax, [ebp+0x8]
 4.    cmp    eax, [ebp+0xc]
 5.    jle    L0
 6.    mov    eax, [ebp+0x8]
 7.    jmp    L1
 8.    jmp    L1
L0:
 9.    mov    eax, [ebp+0xc]
L1:
 10.   pop    ebp
 11.   ret
```

Fig. 1. Instructions of two functions requiring tow parameters.

본 논문에서는 비슷한 함수를 식별하기 위해 함수를 기본 블록 단위로 분리하여 형성된 제어 흐름 그래프를 이용한 방법을 연구한다. 특히, 실제로 수행하지 않고도 정적으로 얻을 수 있는 실행 경로와 경로를 결정하는 분기 조건을 이용하여 실행 흐름과 의미를 둘 다 반영하여 함수를 식별할 수 있는 방법을 연구한다.

III. 정적 경로를 이용한 함수 식별

3.1. 함수 정보 추출 방법

```
bb1:
 1.    push   ebp
 2.    mov    ebp, esp
 3.    mov    eax, [ebp+0x8]
 4.    cmp    eax, [ebp+0xc]
 5.    jge    bb4
bb2:
 6.    mov    eax, [ebp+0x8]
 7.    jmp    bb5
bb3:
 8.    jmp    bb5
bb4:
 9.    mov    eax, [ebp+0xc]
bb5:
 10.   pop    ebp
 11.   ret
```

```
bb1:
 1.    push   ebp
 2.    mov    ebp, esp
 3.    mov    eax, [ebp+0x8]
 4.    cmp    eax, [ebp+0xc]
 5.    jle    bb4
bb2:
 6.    mov    eax, [ebp+0x8]
 7.    jmp    bb5
bb3:
 8.    jmp    bb5
bb4:
 9.    mov    eax, [ebp+0xc]
bb5:
 10.   pop    ebp
 11.   ret
```

Fig. 2. Result of removing cmp and jump instructions in Fig. 1.

두 함수를 식별하기 위해서 먼저 함수에 대한 정보가 필요하다. Fig. 3.은 파라미터 2개를 필요로 하고 두 파라미터 중에서 작은 값을 반환하는 *min* 함수의 제어 흐름 그래프이다. 함수의 시작과 끝이 정해져있고 조건에 따른 분기 지점을 알 수가 있다. 앞서 언급했듯이 함수의 실행 흐름을 반영하기 위해서 시작 지점부터 끝 지점까지 도달하는 모든 경로를 탐색한다. 루프 발생 시 최악의 경우에는 경로를 무한정 얻을 수 있기 때문에 경로 탐색 결과로 루프는 존재하지 않도록 한다. 이러한 방식으로 Fig. 3.의 *min* 함수에서 얻을 수 있는 경로는 1-2-5, 1-4-5 두 가지이다. 3번 블록은 의미 없는 블록으로 경로를 탐색하면서 자연스럽게 걸러진다.

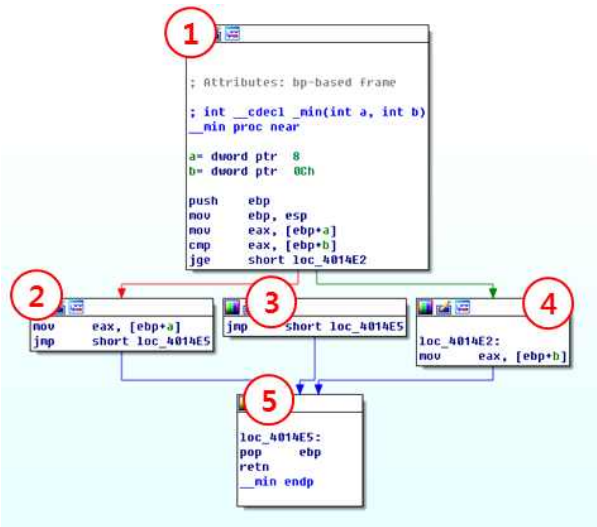


Fig. 3. Control flow graph of minimum function.

경로 탐색 결과에는 경로가 생성된 조건과

분기 여부도 포함된다. 분기 조건은 분기가 발생하는 블록에서 즉, Fig. 3.의 1번 블록에서 정보를 얻을 수 있다. 경로 탐색 결과로 경로가 정해지면 경로에 포함된 비교 명령어(cmp)와 분기 명령어(jmp or conditional jmp)는 제거해도 그 의미가 유지된다.

Fig. 4.는 Fig. 3.에서 1-2-5로 결정되는 경로에서 비교 명령어와 분기 명령어를 제거한 후의 명령어 순서, 분기 조건과 분기 여부를 보여준다. 이 정보를 이용하여 함수를 식별하면 실행 흐름도 반영하면서 세밀한 분석이 가능하다.

```
branch condition : False
cmp     eax, [ebp+0xc]
jge     block 4
```

```
min path 1-2-5
push    ebp
mov     ebp, esp
mov     eax, [ebp+8]
cmp     eax, [ebp+0xc]
jge     block 4
mov     eax, [ebp+8]
jmp     block 5
pop     ebp
ret
```

Fig. 4. Instructions of 1-2-5 path in Fig. 3.

3.2. 함수 식별 방법

이 절에서는 경로 간 비교를 통해 두 함수를 식별하는 방법을 설명한다. 앞서 언급했듯이 Fig 1.과 같은 두 함수를 다른 것으로 인지하기 위해서는 지금까지와는 다른 방법이 필요하다. Fig. 5.와 Fig. 6.은 각각 두 개의 파라미터를 넘겨받고 반대로 처리하는 *min* 함수와 *max* 함수의 제어 흐름 그래프를 통해서 얻을 수 있는 경로를 보여준다.

함수 식별을 위한 첫 번째 단계로 먼저 경로마다 가지고 있는 조건문을 formal하게 변형한다. cmp 명령어와 직후에 오는 명령어를 묶어서 수식을 만드는 것이 가능하다. cmp 연산자의 두 개의 피연산자는 수식의 변수가 되고, 직후의 연산자를 이용하여 수식의 부등호를 결정할 수 있다. 예를 들어, fig. 5.의 *min path1*의 조건을 아래 수식 1과 같이 표현할 수 있다. 수식과 함께 경로 탐색 결과로 정해진 조건도 함께 저장한다.

$$eax \geq [ebp + 0xc] \quad \cdots\cdots (1)$$

다음 단계로, 반환 값이 존재하는 함수는 그 값을 eax에 저장하는 특성을 이용한다. 함수의 실행 결과이기 때문에 마지막으로 eax에 값을 할당하는 명령어의 피연산자 값에 대한 정보를 저장한다. 예를 들면 Fig. 5.의 *min path1*에서는 4번 라인의 명령어가 반환 값을 저장하는 것을 알 수 있다.

```
min path-1
condition : False
cmp     eax, [ebp+0xc]
jge     0x4014e2

1. push ebp
2. mov  ebp, esp
3. mov  eax, [ebp+8]

4. mov  eax, [ebp+8]

5. pop  ebp
6. ret
```

```
min path-2
condition : True
cmp     eax, [ebp+0xc]
jge     0x4014e2

1. push ebp
2. mov  ebp, esp
3. mov  eax, [ebp+8]

4. mov  eax, [ebp+0xc]

5. pop  ebp
6. ret
```

Fig. 5. All path of minimum function.

탐색된 경로마다 분기 조건과 함께 조건문의 수식과 반환 값의 정보를 비교하여 비슷하지만 다른 함수를 식별할 수 있다.

```
max path-1
condition : False
cmp     eax, [ebp+0xc]
jle     0x401462

1. push ebp
2. mov  ebp, esp
3. mov  eax, [ebp+8]

4. mov  eax, [ebp+8]

5. pop  ebp
6. ret
```

```
max path-2
condition : True
cmp     eax, [ebp+0xc]
jle     0x401462

1. push ebp
2. mov  ebp, esp
3. mov  eax, [ebp+8]

4. mov  eax, [ebp+0xc]

5. pop  ebp
6. ret
```

Fig. 6. All path of maximum function.

결과적으로 Fig. 5.와 Fig. 6.의 경로는 Fig. 7.과 같이 간결하게 변형할 수 있다. 두 줄의 정보로 간소화 하였지만 실행 흐름과 결과를 완전히 내포하는 것을 알 수 있다. Fig. 7.에서 *min path1*은 eax가 [ebp+0xc]의 값보다 크지 않으면 [ebp+8]의 값을 반환하고, *min path2*는 eax가 [ebp+0xc]의 값보다 크면 [ebp+0xc]의 값을 반환하는 것으로 보아 *min* 함수는 [ebp+8]

의 값과 [ebp+0xc]의 값 중에서 더 작은 값을 반환하는 과정을 수행한다는 것을 알 수 있다. 반대로 *max* 함수는 더 큰 값을 반환한다는 것도 알 수 있으므로 두 함수가 다른 것으로 구별할 수 있다.

```
min path-1
eax >= [ebp+0xc] : False
eax = [ebp+8]
```

```
min path-2
eax >= [ebp+0xc] : True
eax = [ebp+0xc]
```

```
max path-1
eax <= [ebp+0xc] : False
eax = [ebp+8]
```

```
max path-2
eax <= [ebp+0xc] : True
eax = [ebp+0xc]
```

Fig. 7. Simplification for path data in Fig. 5 and Fig. 6.

IV. 결론

기존의 유사도 분석 방법이나 함수 내에 존재하는 strand 집합 비교 방법으로는 미세한 차이지만 의미가 완전히 달라지는 함수들을 식별할 수 없었다. 본 논문에서는 함수 내에서 탐색 가능한 정적 실행 경로를 이용하여 함수를 식별하는 방법을 제안하였다.

향후에는 분기 조건이나 함수의 반환 값을 직접적으로 알 수 없는 경우 즉, 피연산자에 레지스터나 메모리를 참조할 때 피연산자를 구체화할 수 있는 방법을 연구할 예정이다. 또한 함수 내의 분기문이 많아짐에 따라 탐색되는 경로의 수가 기하급수적으로 증가할 때 효과적인 경로 탐색 방법과 비교 방법을 연구할 예정이다.

[참고문헌]

[1] D. Kim, S. Cho. "*Detection of an Open-Source Software Module based on Function-level Features.*" Journal of KIISE, 42.6 (2015.06): 713-722.

[2] Y. Kim, E.-S. Cho. "*Efficient Similarity Analysis Methods for Same Open Source Functions in Different Versions.*" Journal of KIISE, 44.10 (2017.10): 1019-1025.

[3] N. Shalev, N. Partush. "*Binary Similarity Detection Using Machine Learning.*", The 13th Workshop on Programming Languages and Analysis for Security (PLAS 2018), October, 2018.

[4] Y. David, E. Yahav. "*Tracelet-Based Code Search in Executables.*", Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2014), June, 2014.

[5] Y. David, N. Partush, E. Yahav. "*Statistical Similarity of Binaries.*", Proc. of the 37th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2016), June, 2016.

비정형 위협 데이터 수집 및 연관 분석을 통한 공격 트렌드 분석

박성권, 장성욱*, 김종훈**, 이유정*, 김예준*, 박상호***, 곽경주****
한동대학교, *순천향대학교, **서울과학기술대학교, ***(주)크로젠트, ****금융보안원

Analysis of Cyber Attack Trend with Unstructured Data and Correlation Analysis

Seongkwon Park, Sung-Wook Jang*, Kim Jong Hun**, Lee YuJung*, Ye-Jun Kim*, Park Sangho***, Kyoung-ju Kwak****
Handong Univ., *Soonchunhyang Univ., **Seoul National University of Science and Technology Univ., ***CROZENT Inc., ****Financial Security Institute

요 약

우리나라에 대한 사이버 공격은 지속적으로 이루어지고 있다[1]. 최근에는 새로운 취약점, 진화된 악성코드, 해킹사고로 인해 사이버 공격에 대한 위협이 고조되고 있고 이로 인해 직,간접 피해도 증가하고 있다[2][3]. 대표적인 침해사고 공격 유형은 악성코드를 이용한 공격이며 공격에 이용된 악성코드를 분석하여 도출된 IOC(Indicator Of Compromise)는 공격자를 추적하는데 필수적이다[4]. 본 논문에서는 공개된 비정형 위협 데이터를 수집 및 연관성 분석을 통해 침해상황에서 실시간으로 활용 가능한 시스템을 설계하고 분석 결과를 바탕으로 공격 트렌드를 도출한다.

I. 서론

최근의 악성코드는 특정 목적을 위해 설계되어 단일기능을 수행하는 악성코드가 아닌 다양한 용도로 사용하는 다기능 악성코드에 대한 수요가 증가하고 있다[5]. 또한 IoT 기기를 공격한 변종 악성코드는 12만종이 넘은 것으로 기록됐다. 이는 2017년에 비해 3배가 넘는 수치이다[6]. 이러한 악성코드 분석을 통한 IOC는 공개소스(블로그, 트위터, 보고서 등)을 통해서 공유된다.

대표적으로 금융보안원의 "Threat Intelligence Report - Campaign RIFLE" 분석 보고서에서는 100여개가 넘는 IOC 정보를 공유하고 있다[7].

이처럼 많은 IOC 정보들 중 다루어지지 않고 잊혀지는 IOC 정보들은 그 양이 방대하기 때문에 관리하는데 어려움이 존재한다[8].

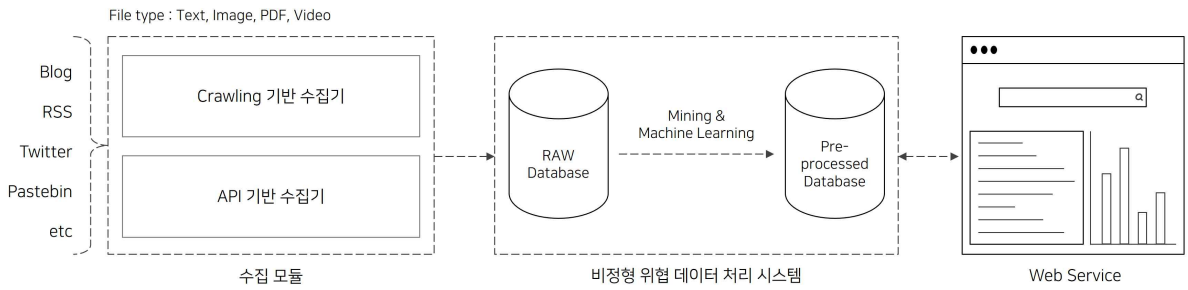
본 논문에서는 공개된 비정형 위협 데이터를 수집하고 수집된 문서 간 연관분석을 통해 공격 트렌드를 도출한다.

II. 본론

2.1 시스템 설계

비정형 위협데이터 수집 및 연관성 분석 시스템은 블로그 게시글, 보고서 등과 같이 다양한 수집원으로부터 데이터를 수집하고 수집된 데이터로부터 IOC요소를 추출하여 연관성이 있는 데이터를 제공하는 시스템이다. <그림 1>은 전체 시스템 구성도이다.

크롤러로부터 텍스트, 이미지, PDF, 영상 등 타입의 데이터를 수집하고 각 타입에 따라 IOC를 추출하여 데이터베이스에 저장한다. 이후 비정형 게시물 유사도 분석을 통해 연관성 분석을 수행한다. 사용자는 웹 기반 검색 서비스를 통해 IOC를 검색할 경우 입력된 IOC의 연관 분석 결과를 시각화 서비스와 상세 리포트로 확인할 수 있다.

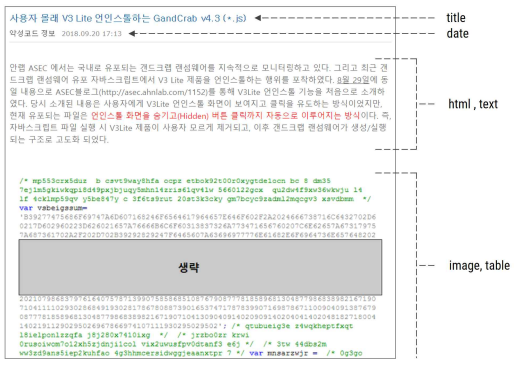


<그림 1> 전체 시스템 구성도

2.2 데이터 수집

비정형 위협 데이터는 국내 4개 사이트, 해외 17개 사이트로부터 수집 시스템을 통해 수집하였다. 효과적인 수집을 위해 수집 전 IOC 요소가 있는 게시물의 카테고리를 선별하였다.

수집 데이터는 각 사이트의 특징에 맞게 수집되며 대표적으로 title, date, html, text, image 등의 데이터가 수집된다.



<그림 2> 수집 데이터 형식[9]

2.2.1 수집 데이터 통계

수집 데이터는 Ahnlab ASEC 블로그, EST Security, 알약 블로그 등 국내 4개 사이트와 Kaspersky Lab-securelist, McAfee-securingtomorrow, Paloaltonetworks-unit42 등 해외 17개 사이트를 대상으로 104,704개의 게시물을 수집하였다.

2.3 데이터 처리

비정형 위협 데이터 처리 시스템은 악성코드 샘플 정보, 악성코드 / 침해사고 분석 보고서, 악성 트래픽 샘플 정보, 사이버 취약점 정보 등 다양한 형태를 가지는 사이버 위협 정보로부터 IOC를 추출한다.

텍스트 데이터 처리는 Raw Data로부터 정규 표현식 (Regular Expression) 을 이용하여 IOC 정보를 추출한다. 추출되는 IOC 정보는 IP, Hash (MD5, SHA-1, SHA-256), E-mail, Domain 정보이다.

PDF 텍스트 데이터 처리는 문서 전체 데이터를 추출하는 과정을 통해 전체 데이터를 추출하고 추출된 데이터를 정규 표현식을 이용하여 IOC 정보를 확보한다.

이미지 처리는 각 게시물과 PDF 문서 등에 포함된 이미지에 삽입된 IOC를 추출한다. Google의 Tesseract-OCR[10]과 Google Cloud Vision API[11]를 통해 OCR (Optical Character Recognition)을 수행한 결과로부터 정규 표현식을 이용하여 IOC를 추출한다.

영상 처리는 영상에 포함된 IOC를 추출한다. 먼저 영상을 이미지화 하는 과정을 통해 초 단위로 데이터를 전처리하여 앞서 제시한 OCR을 수행한 결과로부터 정규 표현식을 이용하여 IOC를 추출한다.

2.4 연관 분석

본 시스템에서는 비정형 게시물 유사도 분석을 통한 연관 분석을 수행한다. 한국어 게시물의 경우 Customized_konlpy[12]와 NLTK[13]를 사용하였다.

2.4.1 자연어 처리

자연어 처리는 게시물의 전문으로부터 명사만 가져온다. 해당 추출 과정에서 koNLPy의 Twitter API를 기반으로 수행하였고 해당 API의 딕셔너리에 정의되지 않은 복합단어의 경우 Customized_konlpy의

n-gram을 이용하여 단어 set을 한 단어로 결합하였다[14]. 예를 들어 "랜섬"과 "웨어"는 각각의 명사로 분류하지만 복합단어 처리 과정 이후 "랜섬웨어" 명사로 인식한다.

스톱워드(Stopword)는 Twitter API에서 분류된 데이터 중 데이터 분석에 도움이 되지 않는 단어를 정의한다. [표 1]은 "Campaign Rifle" 보고서에 등장한 문장을 형태소 분석한 결과이다.

[표 1] 형태소 분석 결과

[('이', 'Noun'), ('공격', 'Noun'), ('은', 'Josa'), ('최소', 'Noun'), ('8', 'Number'), ('개월', 'Noun'), ('전', 'Noun'), ('부터', 'Josa'), ('치밀하게', 'Adjective'), ('준비', 'Noun'), ('된', 'Verb'), ('APT', 'Alpha'), ('의', 'Noun'), ('사례', 'Noun'), ('로', 'Josa'), ('공격', 'Noun'), ('진행', 'Noun'), ('과정', 'Noun'), ('은', 'Josa'), ('다음', 'Noun'), ('과', 'Josa'), ('같다', 'Adjective'), ('.', 'Punctuation')]

'이', '전'과 같이 데이터 분석에 도움이 되지 않는 단어들이 명사로 정의되어 함께 추출되는 경우를 스톱워드 처리 과정을 통해 필터링 된다.

2.4.2 연관 분석

형태소 분석이 완료된 명사 데이터의 등장 빈도는 TF-IDF[15] 계산에 큰 영향을 미친다. 각 게시물에 등장한 명사들의 TF-IDF 계산 결과를 Cosine 유사도[16] 측정을 통해 비정형 게시물 유사도 분석 결과를 도출한다.

2.5 트렌드 분석(Trend Analysis)

트렌드 분석은 게시글의 제목에 있는 문장을 형태소 분석한 뒤 문서간 단어의 빈도수를 계산하여 날짜순으로 분석하였다. 형태소 분석은 ElasticSearch의 Analysis-nori plugin[17]을 사용하였다.

2.5.1 동의어 사전 구성

Ahnlab ASEC 블로그, EST security 알약 블로그 등 국내 4개의 웹사이트에서 분석 리포트 제목에 사용하는 단어들은 각 사이트마다 다르며, 공통적으로 영문과 한글을 번갈아 사용하는 경우가 빈번하다.

[표 2] 동의어 사전 구성 (일부)

동의어 구성	결과
도깨비, DOKKAEBIE, GhostScript, Bindata, 한글문서, .hwp, 한글, hwp	hwp_doc
마이랜섬, Myransom, 메그니버, 메그니베르, 매그니베르	Magniber
랜섬노트, 랜섬, ransom, RansomeWare, 랜섬웨어	Ransomware
스팸, 스팸, 스팸 메일, 스팸 이메일, spam	Spam
트위터, twitter, 페이스북, 페북, facebook	sns
spear phishing, 피싱	Phishing
갠드크랩, gandcrab	GandCrab
rat, 나노코어, Nanocore, 원격제어	RAT
포스트스크립트, Postscript	PostScript

2.5.2 불용어 사전 구성

동의어 사전과 nori plugin 사전을 이용해 해당 단어의 문서간 빈도수(Document Frequency)를 추출했을 때 [표 3]에 있는 일부 단어와 같이 불필요한 단어들을 필터링 하기 위해 불용어 사전을 추가로 구성하였다.

[표 3] 형태소 분석을 통한 빈도수 (상위 10개)

형태소	빈도수
악성	674
코드	484
주의	376
유포	232
분석	231
파일	214
위장	200
랜섬웨어	179
취약	162
점	153

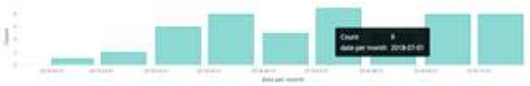
2.5.3 트렌드 분석 그래프

앞서 언급한 전처리 과정을 수행하여 추출한 상위 20개 중요 단어는 [표 4]와 같다.

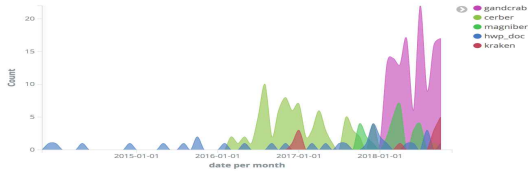
추출된 단어는 각 문서의 Tag로 사용될 수 있으며, 해당 tag는 <그림 3, 4>의 트렌드 분석 그래프로 사용되며 시각화를 통해 표현된다.

[표 4] 사전을 통한 상위 20개 단어

ransomware, mobile, spam, sns, gandcrab, hwp_doc, phishing, trojan, pharming, apt, magniber, spyware, ddos, zeroday, microsoft, linux, botnet, campaign, operation, adobe



<그림 3> "갠드크랩"에 대한 트렌드 그래프



<그림 4> 빈도수 상위 5개 단어에 대한 트렌드 그래프

III. 결론

본 논문에서는 비정형 위협 데이터 수집 및 연관성 분석 서비스를 제공하기 위한 시스템을 설계하고 분석된 결과를 바탕으로 공격 트렌드를 도출하였다. 게시물의 제목에서 핵심 단어를 Tag로 선정하여 공격 트렌드 그래프를 나타내는 것뿐만 아니라 특정 IOC를 검색할 경우 이와 연관된 게시물의 Tag를 확인하여 트렌드 그래프를 제공하는 것도 가능하다.

이로써 비정형 데이터를 활용한 국내의 공격 동향을 분석할 수 있고 추가적으로 영문 자연어 처리를 통한 글로벌 공격 트렌드 분석이 가능할 것으로 기대된다.

[ACKNOWLEDGMENT]

본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best) 지원을 받아 수행되었음.

[참고문헌]

[1] 국가지표체계, 과학기술정보통신부 - 해킹 사고 건수, April, 2018.

[2] 한국인터넷진흥원, 사이버 위협 동향 보고서 (2018년 1분기), pp. 2., April, 2018.

[3] 마이크로소프트, Understanding the Cybersecurity Threat Landscape in Asia Pacific: Securing the Modern Enterprise in a Digital World, May, 2018.

[4] 성정숙, "보안 침해사고 방지 연구", 보안공학연구논문지 제9권 제6호, pp. 504., December, 2012.

[5] KasperskyLab, "Botnet activity in H1 2018: Multifunctional bots becoming more widespread", August, 2018.

[6] Kaspersky Lab, "New IoT-malware grew three-fold in H1", September, 2018.

[7] 금융보안원 침해대응부, "Campaign Rifle :Andariel, the Maiden of Anguish", July, 2017.

[8] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, Raheem Beyah, "Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence", CCS '16 Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 755., October, 2016.

[9] Ahnlab ASEC 블로그, "사용자 몰래 V3 Lite 언인스톨하는 GandCrab v4.3 (*.js)", (http://asec.ahnlab.com/1161), September, 2018.

[10] Google, Tesseract OCR, (https://opensource.google.com/projects/tesseract)

[11] Google, Cloud Vision, (https://cloud.google.com/vision/)

[12] Customized_konlpy, (https://github.com/lovit/customized_konlpy)

[13] NLTK, "Natural Language Toolkit", (https://www.nltk.org/)

[14] 박은정, 조성준, "KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지", 제26회 한글 및 한국어 정보처리 학술대회 논문집, 2014.

[15] 이성직, 김한준, "TF-IDF의 변형을 이용한 전자뉴스에서의 키워드 추출 기법", 한국전자거래학회지 제14권 제4호, pp. 62., November, 2009

[16] 위키백과, "코사인 유사도", (https://ko.wikipedia.org/wiki/코사인_유사도)

[17] ElasticSearch Korean (nori) Analysis Plugin, (https://www.elastic.co/guide/en/elasticsearch/plugins/6.x/analysis-nori.html#analysis-nori)

안드로이드 환경에서의 서드 파티 라이브러리 탐지에 대한 연구

오세라*, 김영갑*†

*세종대학교 정보보호학과 보안공학연구실

A Study on Third-party Library Detection in Android

Se-Ra Oh*, Young-Gab Kim*†

*Security Engineering Lab., Dept. of Computer and Information Security, Sejong University.

요 약

안드로이드 앱을 개발하는데 사용되는 서드 파티 라이브러리는 편리한 도구지만, 검증 없이 무분별하게 사용된다면 보안을 위협할 수 있는 취약점이나 악성 코드들이 앱에 포함될 수 있다. 따라서 보안적으로 취약하거나 악의적 목적을 가진 코드를 포함한 서드 파티 라이브러리는 앱에서 격리하거나 제거할 필요가 있는데, 이를 위해서는 앱 내의 라이브러리를 탐지할 수 있어야 한다. 본 논문은 그러한 서드 파티 라이브러리 탐지에 대한 연구들의 현황을 분석하고, 특히 앱 난독화에 효과적인 라이브러리 탐지를 위해 어떠한 특징(feature)들이 사용되고 있는지 알아본다. 또한, 분석한 라이브러리 탐지 기법들의 장점들을 유지하고 단점을 보완할 수 있는 복합적 라이브러리 탐지 기법을 제안한다.

I. 서론

안드로이드는 대표적인 모바일 OS이다. IDC(international data corporation)에 의하면 안드로이드는 2018년 모바일 OS 시장에서 약 85% 정도의 점유율을 확보했으며[1], Google Play에만 약 250만개의 안드로이드 앱이 존재하는 것으로 분석됐다[2].

안드로이드의 기능들은 모바일의 발전에 따라 과거보다 다양해졌으며 앱에 대한 사용자들의 요구사항들 또한 다양화 및 구체화되면서 앱 개발자들은 기능 구현의 편의성을 위해 서드 파티 라이브러리를 보편적으로 사용하기 시작했다. 하지만 개발자의 의도와 달리 무분별하게 사용되는 서드 파티 라이브러리들에는 미처 파악되지 못한 보안적 위험 요소가 내재되어 앱 사용자에게 악영향을 끼칠 수 있다. 라이브러리들은 일반적으로 한 번 앱에 포함되면 업데이트되지 않은 채 오래도록 사용되므로, 라이브러리에 악성 코드나 취약점이 존재한다면 앱 사용자는 개인정보 유출 등의 보안적 위협에 장기간 노출된다. 따라서, 호스트 앱에서 취약점이 존재하는 서드 파티 라이브러리를 제거하거나 격리하기 위해서는 가장 먼저 앱 내의 서드 파티 라이브러리를 탐지할 수 있어야 한다. 이를 통해 취약점이나 악성 코드가 라이브러리에 존재하는지, 혹은 호스트 앱에 존재하는지에 대한 추적성 또한 제공될 수 있다[3].

본 논문은 안드로이드 환경에서의 서드 파티 라이브러리 탐지 기법의 연구를 위해 최근 3년 사이의 중요 연구들[3, 6, 7]을 분석한다. 특히, 각 연구들이 탐지에 사용한 특징(feature)을 위주로 탐지 기법들의 장점과 단점을 서술하고 이를 기반으로 앱 난독화에 효과적이고 빠른 복합적 라이브러리 탐지 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 라

이 연구는 NAVER 주식회사로부터 지원되었습니다.

† 교신 저자

<표 1> 핵심 서드 파티 라이브러리 탐지 논문의 요약

논문	요약	특징(feature)
LibRadar [7]	• 코드 난독화 방지를 위해 API 특징을 통해 라이브러리 탐지 수행 • 라이브러리 탐지를 위해 클러스터링[8] 활용 • 알려지지 않은 라이브러리 탐지에 강점	• 패키지 별 안드로이드 API 호출 빈도와 종류
LibScout [3]	• 일반적인 코드 난독화에도 유효하고 정확한 버전까지 식별할 수 있는 라이브러리 탐지 방법 제안 • 오리지널 라이브러리 데이터베이스 구축 및 유지보수 (2018년 기준, 248개 라이브러리와 4,965개 버전)	• 패키지 해시와 클래스 해시, 메소드 해시를 기반으로 구성한 머클 트리(Merkle tree)
LibD [6]	• 앱의 내부 코드 의존성과 해싱에 기반한 라이브러리 추출과 분류 수행 • 특히 패키지명과 메소드명 난독화에 효과적	• 기본 블록(basic block) 내의 연산 부호(opcode)

이브러리 탐지 연구들을 분석하고, 3장에서 복합적 라이브러리 탐지 기법을 제안한다. 마지막으로 4장에서는 본 논문의 결론을 서술한다.

II. 서드 파티 라이브러리 탐지 연구

본 장에서는 안드로이드 기반의 서드 파티 라이브러리 탐지 연구들에 대해서 분석한다. 본 논문에서는 각 연구들을 알려진 라이브러리 데이터베이스를 기반으로 하는 것과 그렇지 않은 것으로 구분하였으며, 핵심 요약과 특징을 <표 1>로 정리하였다.

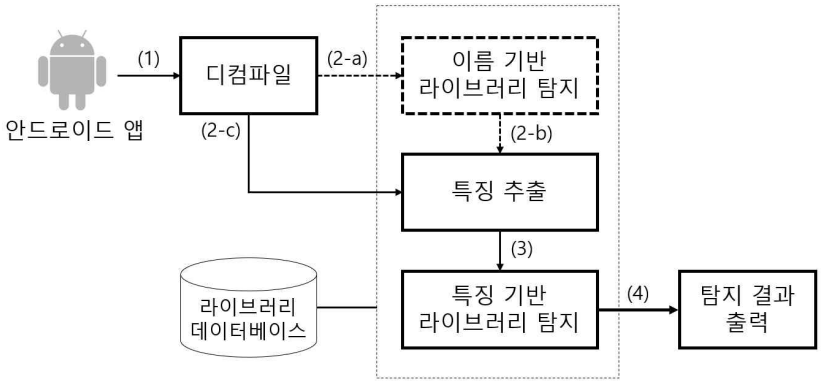
본 논문에서 분석하는 핵심 연구들을 포함한 여러 라이브러리 탐지 연구들은 일반적인 탐지 과정이 존재했다. 가장 먼저 (1)라이브러리의 특징을 추출한 프로파일(profile)들의 데이터베이스를 구축하고 (2)분석할 앱에 존재하는 라이브러리들의 프로파일들을 추출한 뒤, (3)추출한 앱의 프로파일들을 데이터베이스 내의 프로파일들과 비교하여 라이브러리를 탐지하였다. 이때, 연구들의 가장 큰 차이는 라이브러리의 프로파일을 추출할 때 사용되는 특징에 있었다.

1.1 라이브러리 데이터베이스 기반 연구

대부분의 서드 파티 라이브러리 탐지에 대한 연구는 알려진 라이브러리의 데이터베이스를 기반으로 수행되었다. 2010년대 초반, 난독화가 중요하게 고려되기 전 Book 외[4]와 Chen 외[5]는 라이브러리 이름이 나열된 데이터베이스를 기반으로 라이브러리를 탐지하였으나, 앱 난독화에 의해서 이러한 기법의 효과가 적어진 이후 난독화를 고려한 탐지 방안이 2010년 중반부터 주류를 이루었다.

LibScout[3]는 앱에 존재하는 악의적인 코드에 대한 추적성을 제공하기 위한 라이브러리 탐지 도구이며, 오리지널 라이브러리 파일들을 버전별로 세세하게 수집하여 데이터베이스를 구축했기 때문에 탐지한 라이브러리의 버전까지 식별한다. 또한, 난독화에 효과적으로 라이브러리를 탐지하기 위해 난독화될 수 있는 가능성이 큰 코드를 특징으로 사용하지 않고 퍼지화한 메소드 시그네처의 해시들을 기반으로 클래스 해시와 패키지 해시들을 생성하여 머클 트리(Merkle tree)를 구성하여 특징으로 활용하였다. 따라서 탐지를 위한 비교 속도가 빠르며, 앱 난독화 기술 중에서도 API hiding, Control-flow randomization, identifier renaming, string encryption에 강점을 갖는다.

LibD[6]는 기본 블록(basic block) 내의 연산 부호(opcode)를 특징으로 사용하여 라이브러리를 탐지하였다. 한 줄의 코드는 여러 개의 연산 부호로 이루어지므로 라이브러리가 조금만 수정되어도 연산 부호의 변화는 상대적으로 더 크다. 따라서 LibD는 앱에서 사용된 라이브러리가 오리지널 파일에서 변형되었는지 효과적으로 알아낼 수 있으며, 난독화는 일반적으로 피



(그림 1) 제안하는 탐지 아키텍처 개요

연산자(operand)를 변경하므로 앱이 난독화되어 있어도 효과적으로 라이브러리를 탐지할 수 있다. 다만, LibD는 앱에서 사용하는 라이브러리들을 추출하기 위해서 클래스의 상속 관계나 메소드의 호출 관계들을 활용하였는데, 이는 앱 난독화나 라이브러리의 수정 과정에서 바뀔 수 있으므로 라이브러리 추출 정확성이 저하될 수 있고, 이를 기반으로 하는 라이브러리 탐지 과정 또한 신뢰하기 어려워진다.

1.2 그 외의 연구

LibRader[7]는 LibScout나 LibD와는 달리 오리지널 라이브러리 파일들을 기반으로 데이터베이스를 구축하지 않고, 약 백만개에 이르는 안드로이드 앱들에 대해서 패키지 별로 안드로이드 API의 사용 빈도와 종류를 특징으로써 추출하고 클러스터를 구성하여 탐지에 활용하였다. 라이브러리 데이터베이스의 구축이 자동화되는 것은 유용하지만, 탐지 정확성은 오리지널 라이브러리 파일에 기반한 탐지 연구들에 비하면 부족했다. 이는 앱에서 추출한 불완전한 라이브러리들에 기반한 탐지는 오탐(false positive)의 가능성이 높기 때문이다.

III. 제안하는 복합적 탐지 아키텍처

본 장에서는 2장에서 분석한 라이브러리 탐지 기법들의 장점들을 유지하면서 단점을 보완하는 복합적 라이브러리 탐지 아키텍처를 제안한다. (그림 1)은 제안하는 아키텍처의 개요를 보여준다. 라이브러리 탐지를 위해서, 가장 먼저 (1)앱을 디컴파일해야 한다. 최근 라이브러리 탐지 연구들은 난독화를 고려하여 라이브러리명을 탐지에 활용하지 않지만, 앱이 난독화되어 있지 않은 경우 라이브러리명 기반의 탐지 기법도 여전히 효과적이다. 따라서 앱이 난독화되어 있지 않은 경우에는 (2-a)미리 구축한 라이브러리 데이터베이스에 존재하는 라이브러리명들을 기반으로 디컴파일된 앱에서 라이브러리를 검색한다. 그러나 코드나 패키지, 클래스명 등에 라이브러리를 식별할 수 있는 문자열이 존재하지 않거나 앱이 난독화되어 있을 경우 이름에 기반한 탐지는 효과가 적다. 이 경우는 (2-b, 2-c)난독화에 유효한 특징들을 추출한 뒤, 이를 (3)오리지널 라이브러리 파일들에서 추출해놓은 특징들과 비교하여 라이브러리를 탐지하고 (4)사용자에게 결과를 보여준다. 제안하는 탐지 기법은 만약 (2-b) 단계에서 라이브러리가 검색되어도 탐지를 중지하지 않고 특징 기반 라이브러리 탐지도 수행하여 두 가지의 탐지 기법들에 의해 검색된 라이브러리들을 모두 결과로 보여준다.

라이브러리 탐지를 위한 특징과 특징의 비교 과정은 기존 연구[3, 6]에서 제안된 기법들을 복합적으로 활용한다. 즉, 라이브러리 탐지의 가장 기반이 되는 특징인 메소드 해시는 연산 부호로 만들고 이를 기반으로 클래스 해시와 패키지 해시를 만든 뒤 머클 트리를 구성하여 라이브러리를 비교함으로써 빠른 탐지를 수행한다. 한 줄의 고급 언어 코드는 여러 개의 연산 부호를 만들어내므로, 연산 부호를 이용하여 만든 특징은 앱에 포함되어 있는 라이브러리가 오리지널 라이브러리에서 변형되었는지 알아내는데 효과적이다. 또한, 난독화는 일반적으로

연산 부호가 아닌 피연산자를 변경하므로 연산 부호를 이용하여 생성한 특징은 앱 난독화에도 유효하다. 더 나아가, 특징들은 오리지널 라이브러리 파일들을 기반으로 생성되기 때문에 LibRader와 달리 오탐이 적다.

마지막으로, 제안하는 탐지 기법은 특징 해시 값들을 생성하는데 유사도 해시(similarity digest) 알고리즘을 사용한다[9]. 기존의 해시 알고리즘들은 라이브러리 코드가 조금만 변경되어도 완전히 다른 해시 값을 만들어내기 때문에 라이브러리의 변화가 탐지 결과에 미치는 영향이 크다. 하지만 유사도 해시 알고리즘은 라이브러리가 수정된 만큼만 해시 값을 다르게 생성한다. 그러므로 유사도 해시를 통하여 라이브러리의 변형이나 버전 차이 등이 머클 트리의 해시 값, 즉 라이브러리 탐지에 미치는 영향을 최소화할 수 있다.

IV. 결론 및 향후 연구

본 논문은 안드로이드 환경에서의 서드 파티 라이브러리 탐지 연구들을 분석하였다. 특히 라이브러리를 탐지하는데 사용된 특징들과 각 탐지 기법들의 장점 및 단점을 알아보았으며, 이를 기반으로 빠르고 앱 난독화에 효과적인 복합적 라이브러리 탐지 기법을 제안하였다. 향후 연구에서는 제안한 탐지 기법을 실제로 구현하여 성능을 평가할 필요가 있다.

[참고문헌]

[1] IDC. Smartphone OS Market Share. https://www.idc.com/promo/smartphone-market-share/os.

[2] AppBrain. Number of Android apps on Google Play. https://www.appbrain.com/stats/number-of-android-apps.

[3] M. Backes, S. Bugiel, and E. Derr. Reliable Third-Party Library Detection in Android and its Security Applications. CCS '16 Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp.356-367. 2016.

[4] T. Book, A. Pridgen, and D. S. Wallach. Longitudinal Analysis of Android Ad Library Permissions. In Proceedings of IEEE Mobile Security Technologies (MoST' 13), 2013.

[5] K. Chen, P. Liu, and Y. Zhang. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets. In Proceedings of the 36th International Conference on Software Engineering (ICSE), pp.175-186. 2014.

[6] M. Li, W. Wang, P. Wang, S. Wang, D. Wu, J. Liu, R. Xue, and W. Huo. LibD: Scalable and Precise Third-Party Library Detection in Android Markets. 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). pp.335-346. 2017.

[7] Z. Ma, H. Wang, Y. Guo, and X. Chen. LibRadar: Fast and Accurate Detection of Third-Party Libraries in Android Apps. 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). pp.653-656. 2016.

[8] Z. Ma, H. Wang, Y. Guo, and X. Chen. WuKong: a scalable and accurate two-phase approach to Android app clone detection. Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA). pp.71-82. 2015.

[9] Y. Wang, H. Wu, H. Zhang, and A. Rountev. Orlis: Obfuscation-Resilient Library Detection for Android. MOBILESoft '18 Proceedings of the 5th International Conference on Mobile Software Engineering and Systems. pp.13-23. 2018.

속성 기반 암호화 방식을 이용한 블록에 대한 제한적 접근 서비스

류혜원, 김형종
서울여자대학교 정보보호학과

Access controlled Blockchain services based on Attribute-Based Encryption

Hyewon Ryu, Hyung-Jong Kim,
Division of information security, Seoul Women's University.

요 약

기존 블록체인 기반 가상화폐 서비스는 사용자에 거래 서비스를 제공하고 모든 거래 내역을 블록에 저장한다. 그러므로 누구나 블록에 접근하여 정보를 읽을 수 있다. 속성기반 암호화 방식은 속성집합과 접근구조에 따라 복호화할 수 있는 대상이 달라지는 암호 프로토콜이다. 이와 같은 암호화 방식을 이용해 본 논문에서는 모든 사용자의 거래를 지원하지만 이에 따라 생성된 모든 블록을 암호화한다. 이를 통해 복호화할 수 있는 사용자에게만 블록의 접근을 허용하고 블록에 대한 제한적 접근 서비스의 프로토콜을 제시한다.

I. 서론

가상화폐는 지폐와 같은 실물 없이 네트워크로 연결된 가상 공간에서 사용되는 화폐를 말한다. 가장 상용화된 가상화폐는 블록체인 기반으로 개발된 비트코인이다. 비트코인 플랫폼을 이용해 거래가 이루어지며 거래는 블록체인의 모든 참여자 노드가 할 수 있다. 거래 내역은 블록에 저장되기 때문에 수정하거나 위조할 수 없다[1]. 블록에 대한 접근 제한이 없기에 어느 사용자든 거래 내역을 볼 수 있다.

속성기반 암호는 기존 ID 기반암호에서의 개념을 다중화시킨 것으로 사용자의 ID뿐만 아니라 사용자를 식별할 수 있는 속성집합과 이러한 속성집합을 바탕으로 만들어진 접근구조를 통해 암호화와 복호화를 진행한다[2]. 즉, 복호화할 수 있는 속성을 가진 사용자가 한 명이 아니라 다수가 될 수 있다.

위와 같은 속성기반 암호의 특징을 기반으로 본 논문에서는 블록체인 가상화폐 서비스에서[1] 거래의 결과로 생성된 블록에 대한 접근을 제한하는 프로토콜을 제안한다. 블록을 복호화하는 조건을 만족하는 경우만 블록에 접근하여 거래 내역을 확인할 수 있다.

II. 배경 지식 및 관련 연구

2.1 블록체인 가상화폐 서비스[1]

블록체인은 중앙 관리자 없이 자율적으로 동작하는 분산 기술 시스템이다. 중앙 관리자가 없더라도 암호기술과 합의 시스템 기반으로 상호 신뢰를 실현한다. 블록체인 내에서 이뤄진 거래는 모두 블록에 기록된다. 블록을 새로 생

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2018-2018-0-01799)

성하고 이전 블록과 연결하는 것을 채굴이라 하며 채굴에 성공한 노드에는 보상이 제공된다. 비트코인은 이러한 블록체인으로부터 구현된 서비스이다. 중앙 통제자 없이 참여자 노드의 합의를 걸쳐 운영되는 P2P 네트워크로 누구나 참여할 수 있다. 각 노드는 개인키, 공개키, 지갑의 주소로 구성된 지갑을 가지고 있다. 이를 통해 Key와 잔고 관리, 거래 생성과 인증 등의 기능을 수행한다. 노드들의 상호 거래는 거래 내역이 담긴 새로운 블록을 생성되고 다른 노드들에 의해 유효성이 검증되면 승인이 된다.

2.2 속성기반 암호화 방식[2]

속성기반 암호화는 기존 ID 기반 암호화의 속성이 제한적이라는 한계를 극복하기 위해 제안되었다. ID 기반 암호화는 사용자의 공개된 아이디를 공개키로 사용하는데 이것은 하나의 속성만을 가지고 한 것이므로 단편적이고 한계가 있다. 이와 달리, 속성기반 암호화는 공개된 아이디뿐만 아니라 직업, 사는 지역과 같은 사용자의 속성 정보들을 다수로 사용할 수 있고 속성 정보들이 이룬 집합을 바탕으로 접근구조를 지정해 개인키를 생성한다. 접근구조를 만족하는 사용자라면 누구든 복호화를 진행할 수 있다.

2.2.1 CP-ABE (Ciphertext-Policy)[6]

속성기반 암호화를 구현하는 방식으로는 접근구조가 결정되는 순간을 기준으로 2가지가 있다. CP-ABE 방식과 KP-ABE 방식이 있지만 본 논문에서는 CP-ABE 방식을 사용한다. CP-ABE 방식은 암호문 생성 시 속성집합을 이용해 접근구조를 지정한다. 예를 들어, 앨리스가 속성집합 A={'정보','보호','연구'}를 바탕으로 접근구조를 지정하여 암호화를 진행하였다면 이것을 복호화하기 위해 밥은 해당 접근구조에 적합한 속성집합이 있어야 한다.

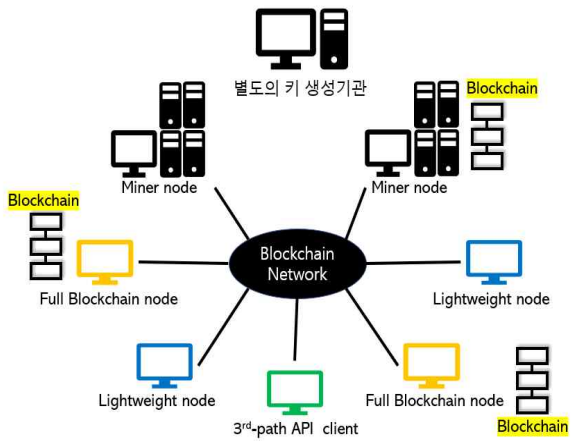
2.3 관련 연구

속성기반 암호화 방식을 서비스에 적용한 연구를 통해 속성기반 암호화의 알고리즘과 특징을 알 수 있다.

속성기반 암호화 기법을 이용해 메신저 시스템을 제안한 연구에서는 속성기반 암호를 통해 이기종 디바이스 간의 안전한 암호화 키 동기화 기능을 가능하게 하며 스마트 폰 기기 분실과 변경에 따른 메시지 동기화를 가능하게 하는 프로토콜을 제안하였다[3]. DRM Cloud 환경에서 속성기반 암호화 방식을 적용한 콘텐츠 서비스 방식에 관한 연구에서는 보안성을 높이고 세부화된 접근 제어가 가능한 서비스를 제안하였다[4]. 패스워드 인증키 교환 프로토콜에 속성기반 암호화 방식을 적용한 연구에서는 다중 서버 프로토콜을 제안하며 속성값을 패스워드로 보고 공개키 기반 암호화가 가능한 다중 서버를 제시하였다[5].

III. 연구내용

3.1 시스템 가정



[그림 1] 속성기반 암호화 방식 적용 시 블록체인 네트워크 구성

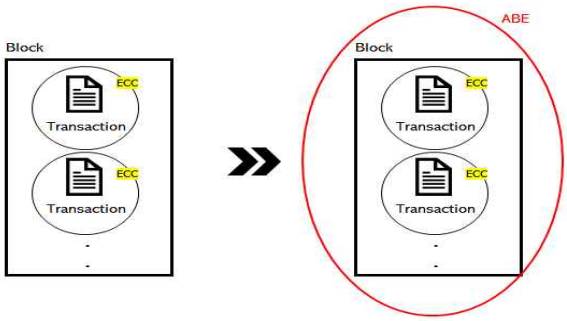
기존 비트코인의 기능과 블록체인에서의 노드의 구성은 같게 구성된다. 하지만 속성기반 암호화 방식을 적용하기 위해 [그림 1]과 같이 별도의 키 생성기관이 필요하다. 또한, 채굴에 성공하여 블록이 생성되었을 때 블록은 속성기반 암호화 방식으로 암호화되어야 한다.

3.2 키 생성기관의 역할

속성기반 암호화 방식을 구현하기 위해서는 블록체인 내의 작동될 수 있는 키 생성기관이 필요하다. 블록을 암호화하기 위해 속성집합을 바탕으로 접근구조를 만들어야 하며 이는 새롭게 생성되는 블록마다 모두 달라야 한다. 또한, 마스터키와 공개키를 생성하고 블록의 유효성 검증을 진행할 노드에게 전달해야 한다. 복호화를 진행하기 위해서는 해당 노드가 가지고 있는 속성집합을 바탕으로 개인키를 생성해줘야 한다.

3.3 블록의 구성

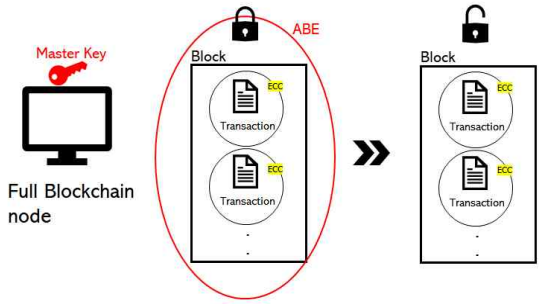
기존 블록 내부에는 ECC로 암호화된 다수의 transaction이 있다. 하지만 블록 자체에는 또 다른 암호화가 되어 있지 않았다. 본 논문에서 제안하는 프로토콜의 블록은 내부에 ECC로 암호화된 transaction이 존재하고 블록 자체도 생성될 때 자동으로 속성기반 암호로 암호화된다.



[그림 2] 속성기반 암호화 방식으로 암호화된 블록

3.4 블록의 암호화와 복호화

채굴을 통해 새로운 블록이 생성되면 키 생성기관에서 해당 블록에 대한 속성집합의 접근구조를 지정한다. 본 논문에서 제공하는 블록체인 가상화폐 서비스는 CP-ABE를 기반으로 속성기반 암호화 방식이 구현되므로 속성집합을 바탕으로 한 접근구조는 블록이 암호화될 때 정해진다. 또한, 이 과정에서 키 생성기관은 마스터키와 공개키를 함께 생성한다. 그리고 블록은 시스템 설정에 의해 자동으로 속성기반 암호화 방식으로 암호화되어 마스터키와 함께 유효성을 검증하는 노드에게 전달된다. 이것을 전달받은 노드는 [그림 3]과 같이 마스터키를 이용해 블록을 잠시 복호화하여 유효성을 검증할 수 있게 된다. 검증된 블록은 암호화된 상태로 블록체인에 연결된다. 검증되지 않은 블록은 폐기된다. 블록마다 정해진 접근구조가 다르게 설정되어 새로운 블록이 생성될 때마다 다른 접근구조로 암호화된다. 그러므로 복호화하기 위해 만족 되는 속성집합의 개인키 역시 다르다. 블록에 저장된 거래 내역을 보기 위해서는 블록을 복호화해야 한다. 블록체인 관리자가 지정한 사용자에게만 접근구조에 일치하는 속성집합이 전달되고 이에 해당하는 노드의 개인키가 키 생성기관에 의해 생성되어 복호화를 진행할 수 있게 된다. 그 외의 사용자에게는 접근구조에 일치하지 않는 속성집합이 있으므로 블록의 거래 내역을 볼 수 없다.



[그림 3] 마스터키를 이용해 블록을 복호화

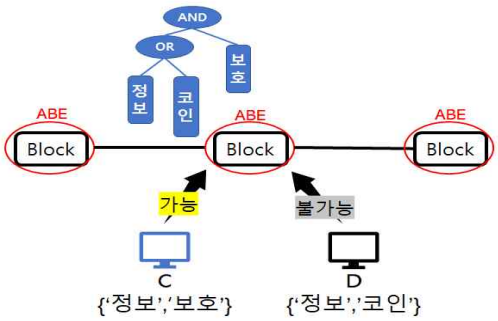
3.5 거래 과정

사용자 A가 B에게 코인을 송금하는 상황을 가정한다. 이때 송금에 대한 transaction이 생성되고 전자 서명을 통해 검증된 transaction은 모든 노드에게 전달이 되고 임시 풀에 쌓인다. 새로운 블록을 채굴하기 위해 임시 풀에서 후보 블록을 구성하게 되며 채굴이 시작된다. 채굴에 성공한 노드에는 보상이 주어지고, 생성된 블록을 속성기반 암호 방식으로 암호화하기 위해 키 생성기관은 마스터키와 공개키를 생성하고 속성집합을 바탕으로 접근구조를 지정한다. 마스터키와 암호화된 블록은 블록의 유효성을 검증할 노드에게 전달이 된다. 이것을 받은 노드는 마스터키를 이용해 블록을 복호화하고 이

에 대한 유효성을 검증한다. 다시 암호화된 상태로 기존 블록체인에 연결되고 해당 거래는 완료된다.

3.6 블록 내 거래 내역에 대한 접근

기존 블록체인 가상화폐 서비스에서는 거래 내역을 누구든 확인할 수 있다. 하지만 속성기반 암호화 방식을 블록에 적용한 서비스의 경우 복호화할 수 있는 사용자만이 블록을 복호화하여 거래 내역을 확인할 수 있다. 따라서 복호화 가능한 속성집합을 가진 사용자만이 거래 내역을 확인할 수 있다. [그림 4]의 경우, 블록의 접근구조는 {{정보∨코인}∧보호}이다. 사용자 C는 {'정보','보호'}의 속성집합을 지니며 사용자 D는 {'정보','코인'}의 속성집합을 지닌다. 블록의 접근구조에 만족하는 속성집합은 {'정보','보호'}와 {'코인','보호'}가 있다. 사용자 C의 속성집합은 첫 번째 속성집합과 일치하므로 해당 블록에 대한 복호화를 진행하여 저장된 정보를 확인할 수 있다. 반면, 사용자 D는 요구되는 속성집합과 일치하지 않으므로 블록의 복호화를 진행할 수 없다.



[그림 4] 블록에 접근하는 경우

IV. 결론

본 논문에서는 기존 블록체인에서의 거래 기능을 유지하면서 속성기반 암호화 방식을 이용해 블록을 암호화, 복호화하여 블록에 저장된 정보에 접근을 제한할 수 있는 프로토콜을 제안하였다.

연구내용을 바탕으로 본 논문은 블록체인에 속성기반 암호화 방식을 적용하는 방법을 구현하였으며 이를 통해 블록체인 기능을 유지하면서 속성기반 암호화 방식뿐만 아니라 다른 암호화 방식도 블록체인에 접목할 수 있다는 가능성을 제시하였다. 향후 과제로는 다른 암호화 방식으로 구현한 블록체인 거래 서비스에 관한 연구가 있다. 또한, 속성기반 암호화 방식을 사용할 때 탈중앙화를 위배할 수 있는 별도의 키 생성기관을 사용하는 것이 아니라 난수 생성기나 seed값을 이용해 키를 생성해 사용하는 방법과 스마트 계약으로 마스터키를 노드에게 전달하는 방식에 관한 연구가 필요할 것으로 보인다.

[참고문헌]

[1] 이동영, 박지우, 이준하, 이상록, 박수용, 블록체인 핵심 기술과 국내외 동향. 정보과학회지, 35(6), 22-28, 2017.

[2] 박광용 송유진, 속성기반 암호기술, 정보보호학회지, 20(2), 85-92., April 2010.

[3] 김연태, 김효승, 조효진, 이동훈, 속성기반암호화 기법을 이용한 안전한 메신저 시스템, Jounal of Security Engineering Vol. 12, No.5 pp469-486, 2015.

[4] 허창수, 이혜주, 신상욱, 속성 기반 암호화를 사용한 DRM 클라우드 서비스. 한국정보과학회 학술발표논문집, 1000-1002, 2014.

[5] 박민경, 조은상, 권태경. 속성 기반 암호화 방식을 이용한 다중 서버 패스워드 인증 키 교환, 한국통신학회논문지, 40(8), 1597-1605, 2015.

[6] J.Bethencourt, A.Sahai and B.Waters, Ciphertext-Policy Attribute-Based Encryption, IEEE Symposium on Security and Privacy, pp.321-334, May 2007.

[7] P.P..Kumar, P.S.Kumar and P.J.A.Alphonse, Attribute based encryption in cloud computing: A survey, gap analysis, and future directions, Journal of Network and Computer Applications, 108, pp.37-52, 2018.

IoT 환경에서 타원곡선 암호를 활용한 OAuth 프로토콜1)

이대휘, 이임영
순천향대학교 컴퓨터학과
순천향대학교 컴퓨터소프트웨어공학과

An OAuth Protocol using Elliptic Curve Cryptography in IoT environment

Dae-Hwi Lee, Im-Yeong Lee
Dept of Computer Science and Engineering, Soonchunhyang University

요 약

IoT는 사물과 사람, 그리고 공간이 연결되어 서비스를 제공할 수 있는 환경이자 기술을 의미한다. 최근 다양한 IoT 서비스가 등장하면서 각 서비스를 연동할 수 있는 기술들이 등장하고 있으며, OAuth도 이 중 하나의 기술이다. OAuth 프로토콜은 IoT 환경뿐만 아니라 인터넷에 연결된 클라이언트 서비스에게 리소스 서버의 계정 정보를 공유하지 않고 리소스 서버에 저장된 사용자의 자원에 접근할 수 있도록 접근 권한을 인가해주는 표준이다. 하지만, OAuth는 OAuth 2.0이 나온 이후로 지속적으로 보안 위협에 대해 논의되고 있다. 최근 많이 사용되는 OAuth 2.0 프로토콜에서는 이미 재전송 공격이나 CSRF 등의 보안 취약점들이 발표되어 있으며 이를 방지하기 위해 OAuth를 기반으로 한 다양한 접근제어 프로토콜이 연구되고 있다. 본 논문에서는 IoT 환경에서 다양한 보안 위협에 안전한 타원곡선 암호 기반의 OAuth 프로토콜을 설계하여 IoT 서비스가 외부 리소스 서버의 사용자 리소스에 대한 접근 권한을 안전하게 인가받을 수 있도록 한다.

I. 서론

최근 IoT(Internet of Things) 환경은 4차 산업혁명의 중심이 되면서 지속적으로 발전하고 있다. 이로 인해 다양한 IoT 서비스들이 출시되어 우리의 생활을 더욱 편하게 만들어주고 있다. 각국의 주요 제조업체들은 여러 종류의 IoT 디바이스를 출시하고 있으며, 이런 다양한 IoT 디바이스가 원활하게 통신할 수 있도록 oneM2M, IETF 등에서 IoT 통신 기술에 대한 표준화 작업을 진행하고 있다[1].

하지만, IoT 환경에서는 다양한 디바이스가 여러 통신 기술로 통신하는 만큼 보안에 대해 고려되어야 한다. 이기종의 다양한 디바이스가 통신을 할 때 각 디바이스들의 보안 취약점들이 모여서 새로운 보안 취약점이 발생할 수 있으며, 센서 디바이스가 수집한 데이터가 처리되는 과정에서 프라이버시 관련 문제 또한 발생할 수 있다[2]. 특히 프라이버시 관련 문제는 IoT에서 매우 중요하다. 스마트홈 서비스를 예로 들면 가정에서 발생하는 스마트홈 서비스 데이터는 스마트홈 사용자의 일상생활과 가장 밀접하게 관련 있는 민감한 데이터이다. 때문에 이 데이터들이 노출된다면 사용자는 금전적인 피해를 입을 수 있으며 가택 침입 등의 상해까지 발생할 가능성이 있다. 따라서 프라이버시에 대한 문제는 IoT에서 중요하게 다루어야할 부분이다.

본 논문에서는 IoT 환경에서 프라이버시 노출 해결을 위해 OAuth(Open Authorization)를

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음
(IITP-2018-2015-0-00403)

사용하여 각 서비스가 연동되는 IoT 환경에서 다양한 보안 위협에 안전한 접근제어 기법을 제안한다. 최근 많은 서비스들이 지원하는 OAuth는 2012년에 제안된 표준으로 인터넷에 연결된 IoT 서비스에게 사용자의 정보를 공유하지 않고 외부의 리소스 서버에 저장된 사용자의 자원에 접근할 수 있도록 권한을 인가해 준다. 이를 이용해 IoT 서비스에 사용자의 정보를 남기지 않도록 하여 프라이버시 문제 해결이 가능하다[3,4].

II. 관련연구

2.1 OAuth(Open Authorization)

OAuth는 인터넷에 연결된 클라이언트 서비스에게 리소스 서버의 계정 정보를 공유하지 않고 리소스 서버에 저장된 사용자의 자원에 접근할 수 있도록 권한을 인가해줄 수 있는 표준이다. OAuth를 통해 클라이언트와 리소스 서버는 별도의 인증 과정이 필요 없게 되므로 여러 서비스를 통합하여 사용 가능하다. 인가 서버에서 클라이언트에게 토큰을 부여하고 클라이언트는 토큰을 이용해 리소스 서버에 접근하여 사용자의 자원을 받을 수 있다. 하지만, OAuth는 재전송 공격과 CSRF(Cross Site Request Forgery) 공격 등의 보안 취약점이 존재하기 때문에 이를 해결하기 위한 방안이 필요하다.

2.2 타원곡선 암호

타원곡선 암호는 유한체상의 타원곡선 위의 점들 사이의 연산에서 정의되는 타원곡선 이산대수 문제(Discrete Logarithm Problem)의 어려움을 이용한 암호 시스템이다. 주로 기존 공개키 기반 보안 서비스인 전자서명이나 키 교환 알고리즘에 사용된다. RSA, ElGamal이나 Diffie-Hellman 키 교환과 같은 기존 공개키 기반 암호 기술들에 비해 타원곡선 암호는 키 길이가 작기 때문에, 최근에는 기존의 암호 기술들을 타원곡선 암호를 이용해 많이 구현하고 있다. 특히, IoT와 같은 제한적인 자원을 갖는 디바이스 네트워크에서 메모리의 처리 능력이 낮은 디바이스를 대상으로 적용할 때 효율적으로 사용될 수 있다.

III. 제안방식

2.1 클라이언트 인증 단계

Step 1. 사용자는 IoT 서비스를 제공하는 클라이언트에게 요청 메시지를 보낸다.

Step 2. 클라이언트는 사용자에게 인가 서버의 URI를 전송한다.

Step 3. 사용자는 인가 서버의 URI를 이용해 인가 서버에 리다이렉트한다.

Step 4. 사용자는 등록되어 있는 클라이언트의 URI를 통해 사용자의 요청 신호를 받고, 이를 통해 클라이언트에 대한 인증을 수행한다.

Step 5. 인가 서버는 클라이언트의 리다이렉트 URI와 함께 인증 코드를 사용자에게 전송한다.

Step 6. 사용자는 클라이언트에게 리다이렉트 한다.

2.2 액세스 토큰 발급 단계

Step 1. 사용자는 클라이언트에게 리다이렉트 하여 인증 코드를 보내주고, 클라이언트는 인가 서버에 접근하여 사용자의 인증 요청을 수행한다.

Step 2. 인가 서버는 사용자를 인증하기 위해 타원곡선 암호로 암호화된 인증 SMS를 발송한다.

Step 3. 사용자는 자신의 PW를 이용하여 암호화된 내용을 확인하고, 이를 통해 인가 서버에 접근하여 사용자 인증을 수행한다.

Step 4. 인가 서버는 클라이언트에게 액세스 토큰을 발급한다.

Step 5. 정상적으로 토큰을 발급받은 클라이언

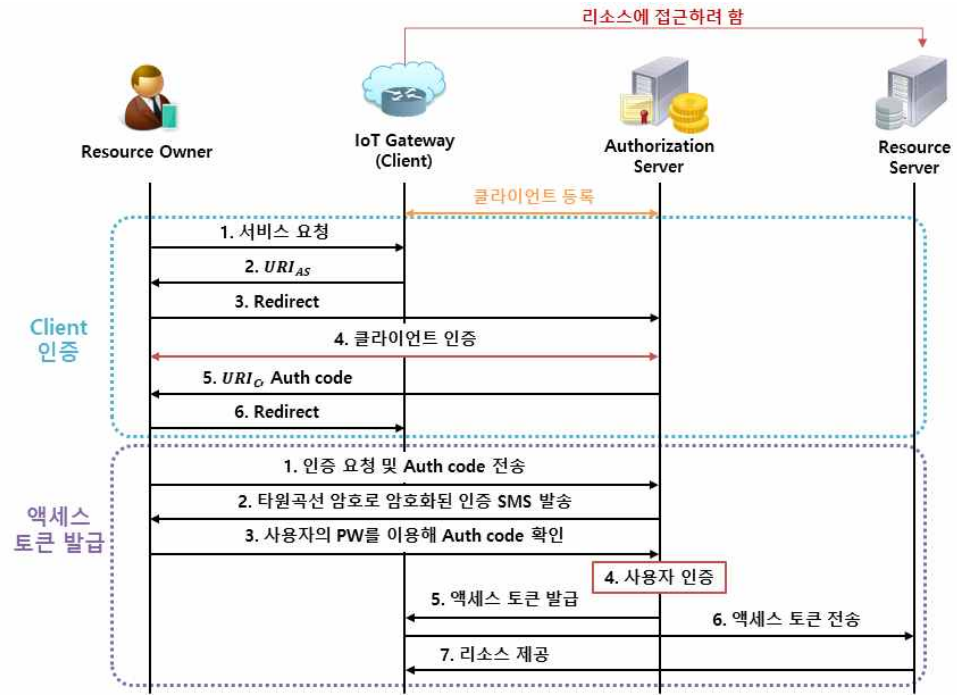


그림 1. 제안방식 시나리오

트는 리소스 서버에 접근하여 리소스 서버로부터 사용자의 리소스를 제공받는다.

II. 결론

본 제안방식에서는 IoT 환경에서 안전한 OAuth 기반의 접근제어 기법을 제안하였다. OAuth 표준의 구조를 만족하며, 각 메시지가 전송되는 과정에 있어서 재전송 공격에 안전하고 CSRF 공격에도 또한 안전하도록 설계되었다. 기존에 제안된 IoT 환경에서 OAuth를 적용한 방법들보다 통신 횟수는 비교적 높고 공개키 암호를 추가로 사용하지만, 재전송 공격과 CSRF 공격에 대해 안전하도록 고려되어 이를 통해 클라이언트가 안전하게 리소스 서버의 사용자 리소스에 접근할 수 있는 권한을 획득한다.

OAuth는 IoT 환경의 발달로 최근 많은 연구가 진행되고 있으며 여러 서비스가 적용하고 있는 기술로, 좀 더 다양한 환경에 맞도록 지속적인 연구가 진행되어야 한다. 또한, 이후에는 연산 횟수를 줄이고 IoT 환경에서 좀 더 효율성을 제공할 수 있는 방향으로 연구가 필요할 것으로 판단되며, 실제 구현을 통해 비교 및 분석을 진행할 계획이다. IoT 서비스는 우리 생활에 밀접하게 연관되어있는 만큼 보안이 취약하다면 큰 사고를 불러올 수 있을 것이다. 따라서 이후에도 IoT 환경을 위한 접근제어 기법에 대해 많은 연구가 진행되어야 한다.

[참고문헌]

[1] M. Karaköse and H. Yetis, "A Cyberphysical System Based Mass-Customization Approach with Integration of Industry 4.0 and Smart City," Wireless Communications and Mobile Computing 2017, 2017.

[2] J. H. Ziegeldorf, O. G. Morchon and K. Wehrle, "Privacy in the Internet of Things: threats and challenges," Security and Communication Networks, Vol. 7, No. 12, pp. 2728-2742, 2014.

[3] E. Hammer-Lahav, "The OAuth 1.0 Protocol", IETF, RFC5849, 2010.

[4] D. Hart, "The OAuth 2.0 Authorization Framework," IETF, RFC6749, 2012.

클라우드 스토리지 환경에서의 Multi-use ID based 프록시 재암호화+

김원빈*, 이임영**
순천향대학교 컴퓨터학과

Multi-use ID based proxy re-encryption in a cloud storage environment

Won-Bin Kim*, Im-Yeong Lee**
Dept of Computer Science and Engineering, Soonchunhyang University*,**

요 약

프록시 재암호화 기술은 데이터 소유자가 위임자에게 데이터를 복호화할 수 있도록 재암호화 하는 기술을 의미한다. 하지만 일반적인 프록시 재암호화 기술은 한 번 재암호화 된 암호문을 다시 재암호화 할 수 없다. 따라서 Alice가 Bob에게 재암호화하여 제공한 암호문을 Bob이 Charlie에게 제공하기 위해서는 Bob이 암호문을 복호화하여 평문을 획득하고, 이를 다시 암호화 한 뒤, 재암호화를 거쳐 Charlie에게 제공하여야 한다. 본 연구에서는 이러한 불편함을 해결하기 위해 재암호화된 암호문을 다시 재암호화 할 수 있는 multi-use 방식을 적용하기 위해 연구를 수행하였다.

I. 서론

프록시 재암호화 기술은 소유자가 소유한 데이터를 암호화한 뒤, 위임자가 해당 데이터를 복호화 할 수 있도록 프록시 서버가 재암호화 하는 기술을 의미한다. 이 과정에서 프록시 서버는 데이터의 원본을 확인할 수 없으며, 위임자는 자신의 개인키로 재암호화 된 데이터를 복호화할 수 있어야 한다. 이는 데이터를 Alice에서 bob으로 전달되는 형태인 클라우드 스토리지를 이용하여 데이터를 공유하는 환경에서 주로 이용된다.

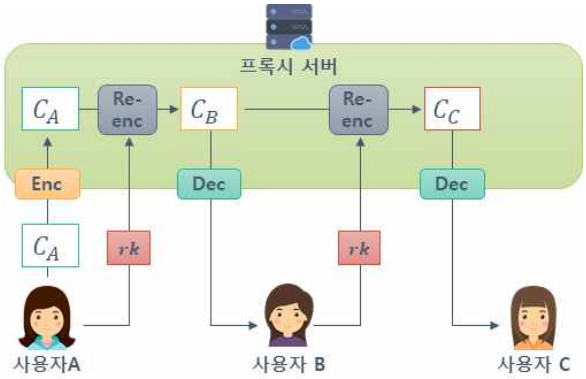
2007년 M. Green과 G. Ateniese는 multi-use 프록시 재암호화 방식을 제안하였다[1]. Multi-use 프록시 재암호화 방식은 Alice가 Bob에게 제공하기 위해 재암호화 한 암호문을 다시 Charlie가 복호화 할 수 있도록 재암호화 하고, 이어서 Dave가 복호화할 수 있도록 재암호화를 여러 번 수행할 수 있도록 하는 기술이다. 이는 클라우드 스토리지를 이용하여 데이터를 반복적으로 공유하는 환경에 더욱 적합한 특징을 갖는다. 이에 따라 본 연구에서는 multi-use 방식을 이용하여 데이터를 보다 효율적으로 공유하는 방법을 제안한다.

+ 이 논문은 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업(IITP-2018-2015-0-00403) 및 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2016R1D1A1B03935917)

II. 관련 연구

2.1 프록시 재암호화

프록시 재암호화 기술은 Alice가 소유한 데이터를 프록시 서버가 재암호화 하여 Bob에게 전달하는 기술을 의미한다[2][3]. 프록시 재암호화 기술은 다양한 형태가 존재하며 본 연구에서는 ID Based 방식을 이용하여 수행되었다. ID Based 방식은 위임자인 Bob의 ID를 Bob의 공개키로 이용하는 방식이며, 재암호화 키의 생성을 위해서는 Bob의 ID가 이용된다. 이후,



(그림 1) Multi-use 프록시 재암호화 개요

Bob은 재암호화 데이터를 복호화하기 위해 자신의 개인키를 이용하여 복호화를 수행한다[4].

2.2 Multi-use

Multi-use는 (그림 1)과 같이 프록시 재암호화 기술이 갖는 특성 중 하나이며 single-use와 대비되는 속성이다. 이 속성은 프록시가 단일 암호문에 대해 여러 번 재 암호화를 수행하도록 하는 속성이다. 예를 들어 Alice에서 Bob으로 암호화한 다음, 다시 Bob에서 Charlie로 재암호화 하는 것을 의미한다.

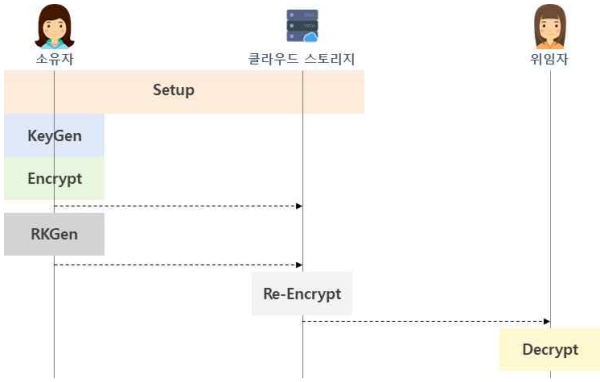
III. 보안 요구사항

본 제안방식에서는 프록시 재암호화 기술에 다음과 같은 사항을 요구한다.

- 기밀성(Confidentiality) : 클라우드 스토리지에 업로드된 데이터는 정당한 소유자 이외에는 원본을 확인할 수 없도록 해야 한다.
- 단방향성(Unidirectional) : 소유자가 위임자의 암호문을 해독하지 못하면서도 소유자가 위임자에게 복호 권한을 위임할 수 있도록 해야 한다.
- 다중 사용(Multi-use) : 클라우드 스토리지가 단일 암호문에 대해 여러 번 재 암호화를 할 수 있도록 해야 한다.

IV. 제안방식

본 논문은 ID Based 프록시 재암호화 방식을 제안하며, 해당 방식의 설명은 (그림 2)와 같다.



(그림 2) 제안방식의 개요도

4.1 Setup(1^{λ})

공개 파라미터를 생성하여 분산된 사용자에게 공개하며, 마스터 비밀 키(MSK)를 생성하여 안전하게 보관한다.

4.2 KeyGen($param, MSK, id_o$)

소유자는 공개 파라미터와 마스터 비밀 키, id_o를 입력하여 소유자의 개인키를 출력한다.

4.3 Encrypt($param, id_{o,m}$)

소유자는 공개 파라미터, 자신의 id_o와 함께 암호화하려는 메시지 m을 입력하여 암호문 C_{id_o}를 획득하고, 이를 클라우드 스토리지에 저장한다.

4.4 RKGen($param, sk_{id_o} id_o, id_d$)

소유자는 공개 파라미터와 자신의 개인키, id_o와 위임자의 id_d를 입력하여 재암호화 키 $rk_{id_o \to id_d}$를 생성한다.

4.5 Re-encrypt($param, rk_{id_o \to id_d}, C_{id_o}$)

클라우드 스토리지는 암호문 C_{id_o}와 $rk_{id_o \to id_d}$를 이용해 재암호화를 수행하여 C_{id_d}를 획득한다.

4.6 Decrypt($param, dk_{id_d} C_{id_d}$)

위임자는 자신의 개인키 sk_{id_d}를 이용하여 C_{id_d}를 복호화하고 m을 획득한다.

V. 제안방식 분석

• 기밀성(Confidentiality) : 클라우드 스토리지에 업로드 되는 데이터는 데이터 소유자가 자신의 공개키로 암호화를 수행한 뒤 업로드를 한다. 따라서 업로드된 데이터는 소유자의 개인키로만 복호화를 수행할 수 있다. 만약 소유자가 위임자의 공개키를 이용하여 재암호화 키를 생성하고, 암호문을 재암호화 할 경우, 위임자의 개인키로만 암호문을 복호화 할 수 있다.

• 단방향성(Unidirectional) : 소유자는 위임자의 공개키를 취득하여 재암호화 키를 생성할 수 있다. 하지만 공개키 암호화 방식의 특성상 위임자의 공개키를 이용하여도 위임자의 공개키로 암호화된 암호문을 복호화 할 수 없다.

• 다중사용(Multi-use) : 소유자가 위임자에게 재암호화하여 제공한 암호문은 위임자가 다른 위임자에게 재암호화하여 제공할 수 있다. 이는 재암호화 과정에서 암호문의 형태가 유지되기 때문이다.

VI. 결론

본 제안방식은 클라우드 스토리지 환경에서 데이터를 보다 효율적으로 공유하기 위한 기술을 다룬다. 이를 위해 프록시 재암호화 기술을 사용하여 데이터 소유자가 위임자에게 데이터 복호화 권한을 위임받는 기술을 연구하였다. 하지만 일반적인 프록시 재암호화 방식은 재암호화 된 암호문의 추가 재암호화가 불가능하여 데이터를 제 3자에게 전달하기 힘든 구조를 갖고 있다. 이를 해결하기 위해 본 제안방식은 재암호화 된 암호문의 반복적인 재암호화가 가능하게 하여 데이터를 Alice로부터 Bob으로, Bob에서 Charlie로, Charlie에서 Dave로 반복하여 전달할 수 있다. 이를 통해 본 제안방식은 추가 재암호화를 수행하기 위해 복호화, 암호화, 재암호화를 매번 수행하지 않고도, 재암호문을 다시 암호화하여 연산량 감소와 스토리지 관리 측면에서 효율성을 갖도록 하였다.

[참고문헌]

[1] Green, Matthew, and Giuseppe Ateniese. "Identity-based proxy re-encryption." Applied Cryptography and Network Security. Springer, Berlin, Heidelberg, 2007.

[2] Chu, Cheng-Kang, and Wen-Guey Tzeng. "Identity-based proxy re-encryption without random oracles." International Conference on Information Security. Springer, Berlin, Heidelberg, 2007.

[3] Shao, Jun, et al. "Multi-use unidirectional proxy re-encryption." Communications (ICC), 2011 IEEE International Conference on. IEEE, 2011.

[4] Shao, Jun, and Zhenfu Cao. "Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption." Information Sciences 206 (2012): 83-95.Security Symposium, 2013.

임베디드 펌웨어의 유사 함수식별을 위한 분석 방법

허정민* 김지민* 지청민* 홍만표**

*아주대학교 컴퓨터공학과 **사이버보안학과

The analysis method to identify similar function of the embedded firmware

Jung-Min Heo* Ji-Min Kim* Cheong-Min Ji* Man-Pyo Hong**

*Dept. of Computer Engineering, **Cyber security, Ajou University

요 약

임베디드 디바이스의 보급이 널리 확산되면서 디바이스에 내재된 펌웨어의 보안은 더욱 중요시되고 있다. 그러나 펌웨어의 버전 또한 계속 업데이트되어 출시되고 있기 때문에 새로 출시되는 임베디드 디바이스 펌웨어들에 대한 안전성 평가를 수행하는 것이 필요하다. 안전성 평가는 다양한 분석 기술을 사용하여 펌웨어의 취약점 여부를 확인할 수 있는데 본 논문에서는 Diaphora를 사용한 바이너리 디핑 결과를 기반으로 분석 할 대상 함수를 선정한다. 두 바이너리 파일의 디핑 결과는 Diaphora에 정의되어 있는 41개의 Heuristic에 의해 결과가 생성되며 이 중 하나라도 매치되면 이는 식별된 함수로 분류되고, 매치되지 않는 경우 식별되지 못한 함수로 분류한다. 그러나 동일함수임에도 식별되지 못하는 문제점이 있어 이를 해결하고자 식별되지 못한 함수를 식별할 수 있는 새로운 분석 방법 네 가지에 대해 제안한다. 이를 평가하기 위해 본 논문에서는 Coreutils에서 사용되는 du, ls, mv 모듈의 유틸리티 버전, 최적화 옵션, 컴파일러 버전의 차이점을 두고 디핑 결과에서 식별되지 못한 함수를 대상으로 제안된 분석 방법을 적용하여 평가 시 추가 함수를 더 식별하였다.

I. 서론

사물인터넷의 발전이 지속적으로 증가함에 따라 현재 유무선 공유기나 Router 등 네트워크 임베디드 디바이스들의 보급은 계속적으로 증가하고 있다. 위와 같은 임베디드 디바이스는 ARM 또는 MIPS 등의 CPU 아키텍처로 설계된 시스템 펌웨어를 포함하며 이 펌웨어 내에는 이더넷 또는 WiFi 등 인터넷 연결을 위한 모듈들이 내장되어 있다. 이러한 네트워크 임베디드 디바이스 펌웨어의 경우 주로 공개된 공간에서 많이 사용되고 있기 때문에 안전하지 않은 펌웨어를 사용할 경우 알려진 위협 및 알려지지 않은 보안 위협에 노출될 수 있다 [1], [2]. 그렇기 때문에 임베디드 디바이스 시스템 펌웨어를 분석하여 취약점 존재 여부를 확인하는 것이 필요하다 [3]. 일반적으로 임베디드 디바이스들의 펌웨어는 신규 업데이트 버전이 계속 출시되기 때문에 분석가들은 정적 분석 및 바이너리 디핑과 같은 다양한 분석 기술을 통해 기존 펌웨어를 기반으로 새로 업데이트 된 펌웨어를 비교 분석하여 취약점을 파악할 수 있다. 이때 사용되는 정적 분석 도구에는 IDA(Interactive DisAssembler) Pro가 있으며 IDA Pro는 바이너리 실행파일을 디스어셈블 하여, 코드를 함수 단위까지 구분해 주고 함수간의 호출 관계 및 제어 흐름 그래프를 생성한다 [11]. 또한 바이너리 디핑을 위해 사용되는 대표적인 상용 도구에는 Diaphora가 있으며 본 논문에서는 Diaphora 도구를 통해 Coreutils 모듈 세 개에 대한 바이너리 디핑 결과를 추출하여 두 바이너리에서 동일 이름을 사용하는 함수임에도 불구하고 식별되지 않은 함수들을 추가 식별 대상으로 선정한다 [9]. 추가 식별 대상으로 선정된 함수들은 Diaphora에서 바이너리 디핑을 수행할 때 동일 함수인지를 판별하는 Heuristic이 41개 정의되어 있는데 위에서 언급한 함수들은 정의 된 Heuristic에 일치하는 것이 없었으며, 두 바이너리 간 식별되지 못한 함수들을 대상으로 유사한 함수가 있다면 이를 식별하기 위한 새로운 함수식별 분석방법 네 가지를 제안한다. 첫 번째 방법은 어셈블리 코드 내 피연산자를 통해 호출된 함수 목록을 사용하며, 두 번째 방법은 함수의 리턴 타입과 매개변수의 리턴 타입 정보를 활용하고, 세 번째 방법은 함수 내 어셈블리 코드의 Mnemonic 리스트를 추출하여 대상 함수의 Mnemonic과 비교한다. 마지막 네 번째 방법은 어셈블리 코드 내에서 포함된 문자열 정보를 통해 비교한다.

본 논문의 구성은 다음과 같다. 2장에서는 배경이 되는 바이너리 디핑과 Diaphora 도구에 대해 기술하고, 3장에서는 두 바이너리 간 유사 함수 여부를 판별하기 위한 새로운 분석 방법에 대해 설명한다. 4장

에서는 실험을 통해 제안하는 분석 방법을 평가하고 검증한다. 5장에서는 평가 결과를 토대로 제안 기법의 한계점 대해 서술하고 결론을 맺는다.

II. 관련 연구

2.1 바이너리 디핑

바이너리 디핑은 서로 다른 두 개의 바이너리실행 파일을 비교하여 분석하는 기술이다. 이 기술은 패치 분석, 함수식별, 악성코드 탐지 등을 새로 출시된 버전과 이전버전의 바이너리 실행 파일 비교 통해 그 차이를 분석하여 보안 취약점을 식별하고 해결하는데 주로 쓰인다 [5], [7], [8].

2.2 Diaphora

Diaphora는 Bindiff, DarunGrim2와 같은 바이너리 디핑 도구 중 하나이다 [6], [10]. Diaphora는 IDA Pro에서 스크립트 파일 형태로 실행 시킬 수 있으며 Github에 소스코드가 공개되어 있어 구하기 쉽고, UI가 단순하기 때문에 바이너리 디핑 도구로 많이 사용되고 있다.

Diaphora에서 바이너리 디핑을 수행하는 방법은 먼저 대상이 되는 바이너리와 비교하려는 바이너리를 IDA Pro에 로드한다. IDA Pro에 로드된 정보를 기반으로 Diaphora에서는 함수의 정보를 SQLite 파일에 각각 저장한다. 그리고 이렇게 만들어진 2개의 SQLite 파일을 이용하여 디핑을 수행하게 되는데, 이때 Diaphora에서는 함수 간 매치를 위해 정의된 Heuristic을 이용하여 함수들을 비교한다. 그리고 결과를 함수의 유사한 정도에 따라 Best Matches, Partial Matches, Unreliable Matches, 그리고 Unmatched로 분류한다.

III. 유사 함수식별을 위한 분석 방법

본 논문에서 제안하는 방법은 대상 바이너리와 비교 바이너리를 Diaphora 도구를 통해 분석한 결과를 기반으로 수행된다. 제안하는 분석 방법을 수행하기 위해 먼저 기준이 되는 펌웨어 바이너리의 SQLite 파일과 대상이 되는 바이너리의 SQLite 파일, 그리고 이 둘을 디핑한 결과 파일이 필요하다. 각각의 SQLite 파일에는 Diaphora에서 추출한 함수들의 정보가 포함된다. 본 논문에서는 그림 1과 같이 추출된 함수 와 결과 파일 정보를 기반으로 Diaphora에서 식별되지 않는 함수들을 추가적으로 식별하기 위한 네 가지 방법을 제안한다.

3.1 식별 대상 선정

유사 함수식별을 위한 대상은 다음과 같이 선정한다. 먼저 바이너리 파일 A와 B가 존재하는 경우, 바이너리 A 는 함수 {a,b,c,d,g,h,i}를 포함하고 있으며 바이너리 B 는 함수 {a,b,c,d,e,f,g}를 포함하고 있다.

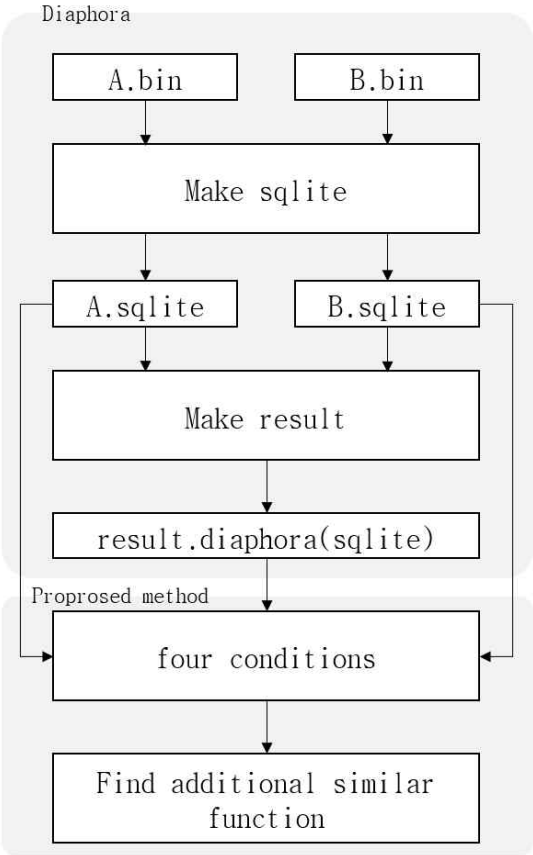


그림 1. Flow chart of proposed analysis method

위 두개 함수의 디핑 결과가 가장 이상적인 경우는 D_{ab}={a,b,c,d,g}, U_a={h,i}, U_b={e,f} 이다. 이는 공통된 함수가 모두 매치된 경우이며, 공통되지 않은 함수는 모두 미스매치로 분류되었다. 그러나 일반적인 바이너리 디핑 시에는 위와 같이 이상적인 결과보다 다음의 경우가 주로 발생한다. 예를 들어 A와 B가 디핑되었을 때 매치된 결과가 D_{ab}={a,b,c}인 경우, U_a={d,g,h,i}, U_b={d,e,f,g} 이다. U_a와 U_b 에 포함된 {d,g} 는 공통된 함수로 보이지만 디핑 도구에서는 이를 식별하지 못한다. 위와 같은 결과를 통해 본 논문에서는 식별되지 못한 함수를 대상으로 3.2절에서 설명하는 네 가지 방법을 적용하여 식별될 수 있는 함수를 찾는다.

3.2 유사함수 분석 방법

3.1절에서 선정된 함수들을 대상으로 아래 네 가지 분석 방법을 적용하여 두 함수가 유사한지를 판별한다. 첫 번째 방법은 어셈블리 코드 내의 피연산자를 통해 호출된 함수 목록을 사용하며, 분석 대상 함수와 비교 대상 함수 간 동일하게 호출되는 함수가 있는지 판별한다. 3개 이상 동일하게 호출된 함수가 있는 경우 두 함수를 유사 함수로 식별한다. 두 번째 방법은 함수의 리턴 타입과 함수에서 사용하는 매개변수 타입의 목록을 이용 한다. 두 함수가 동일할 경

	Utilities version			Optimization option			Compiler version		
	6.12 ↔ 8.26	6.12 ↔ 8.30	8.26 ↔ 8.30	O0 ↔ O2	O0 ↔ O3	O2 ↔ O3	5.4 ↔ 7.3	5.4 ↔ 8.2	7.3 ↔ 8.2
du	5 (181)	5 (181)	1 (371)	31 (376)	43 (323)	38 (323)	10 (372)	8 (373)	3 (378)
ls	3 (244)	5 (244)	3 (320)	46 (340)	64 (311)	31 (310)	7 (337)	9 (338)	2 (341)
mv	9 (187)	4 (187)	2 (282)	25 (324)	52 (297)	30 (297)	5 (321)	9 (321)	2 (321)

표 1. The number of unmatched functions per module by Diaphora(the number in round brakets the total number of function)

	Utilities version			Optimization option			Compiler version		
	6.12 ↔ 8.26	6.12 ↔ 8.30	8.26 ↔ 8.30	O0 ↔ O2	O0 ↔ O3	O2 ↔ O3	5.4 ↔ 7.3	5.4 ↔ 8.2	7.3 ↔ 8.2
du	2	2	0	4	2	17	5	6	1
ls	1	3	1	4	3	6	5	6	1
mv	4	1	1	9	8	7	3	3	0

표 2. The number of functions that match three or more conditions by the proposed analysis method

우 일반적으로 리턴 타입과 사용되는 매개변수가 같기 때문에 이 경우 두 함수를 유사하다고 판단할 수 있다. 세 번째 방법은 함수 내 어셈블리 코드의 Mnemonic 정보를 이용한다. Mnemonic이란 어셈블리 인스트럭션을 기호화 하여 표현한 것으로 하나의 함수 안에는 여러 개의 Mnemonic이 존재하게 된다. 제안하는 분석 방법에서는 기준 함수의 Mnemonic들과 비교 함수의 Mnemonic들을 비교하여 10개 이상의 동일한 Mnemonic 리스트가 5개 이상 존재할 경우 이를 유사한 함수로 분류하게 된다. 네 번째 방법은 함수 내의 어셈블리 코드에서 사용되는 문자열들을 모두 취합하여 하나의 리스트로 만든 후 이를 대상 바이너리의 어셈블리에서 사용된 문자열 리스트와 비교하였을 때 두 문자열 리스트가 반 이상 같은 경우 유사 함수로 분류한다. 제안하는 네 가지 방법을 통해 바이너리 디핑을 수행하게 되면 기존에 식별되지 않는 함수를 추가적으로 식별할 수 있게 되어 전체 함수들의 식별률 향상을 기대할 수 있다.

IV. 실험 및 평가

실험을 위해 사용한 모듈에 대한 설명과 3장에서 제안한 방법을 사용하여 평가를 수행한 결과에 대해 기술한다.

4.1 평가 환경

본 논문에서는 제안한 유사 함수식별 분석 방법의 성능 평가를 위해 유닉스 계열 운영체제에서 주로 사용되는 유틸리티인 GNU Coreutils를 선정하였다. Coreutils에는 수십 개의 다양한 기본 모듈들이 포함되어있으며 그 중 평가에 사용된 모듈은 du, ls, mv 이다. 해당 모듈의 바이너리 파일을 획득하기 위해 공개된 소스코드를 확보하여 동적 링크(Dynamic link) 옵션을 주어 ARM 아키텍처 환경에서 GCC 컴파일러를 통해 컴파일 하였다.

4.2 평가 방법

제안하는 유사함수 식별 방법을 검증하기 위해 본 논문에서는 세 가지의 평가 환경을 다음과 같이 구성하였다. 첫 번째는 유틸리티 버전 두 번째는 최적화 옵션 마지막 세 번째는 컴파일러 버전에 따른 평가를 진행한다 [4]. 첫 번째로 소스 코드의 버전에 차이가 생길 때 두 함수를 동일하게 식별할 수 있는지 판별하기 위해 Coreutils 6.12버전과 8.26버전, 6.12버전과 8.30버전, 그리고 8.26버전과 8.30버전 간 디핑을 수행하였다. 두 번째로 컴파일 시 최적화 옵션이 달라졌을 때 두 함수를 동일하게 식별할 수 있는지 판별하기 위해 최적화 옵션을 O0부터 O3까지 다르게 적용하여 디핑을 수행하였다. 마지막으로 컴파일러 버전을 5.4부터 8.2까지 다르게 적용하여 실험을 수행하고, 위의 세 가지 기준에 대한 디핑을 수행하였고 버전 별 식별률의 변화 정도를 측정하였으며 디핑 결과는 표 1과 같이 총 함수의 개수와 식별되지 않은 함수의 개수로 구분된다.

4.3 평가 결과

표 2은 Diaphora에서 식별되지 않는 함수들을 대상으로 3장에서 제시한 네 가지의 유사 함수식별 분석 방법을 적용하여 각각의 평가 방법에 대해 수행한 결과이며, 식별된 함수들의 개수는 제안된 네 가지의 조건 중 세 가지 이상 맞은 경우에 해당된다. 먼저 Coreutils의 유틸리티 버전 별로 디핑을 수행할 경우 표 1의 유틸리티 버전 표의 2197개의 전체 함수 중 식별되지 않은 함수의 개수가 37개로 약 1.7%의 함수가 식별되지 않았다. 이들 37개의 함수를 대상으로 본 논문의 제안 방법을 적용한 결과 16개의 함수를 추가적으로 식별하였다. 표 2의 데이터에서 6.12 버전에서 8.26으로 변경되었을 때 비 식별된 함수의 개수와 추가로 식별한 함수의 개수가 가장 많았으며, 8.26 버전에서 8.30 버전은 거의 차이가 없음을 알 수 있다. 두 번째로 최적화 옵션이 달라지는

경우에는 전체 2901개의 함수 중 360개의 함수가 식별되지 않았는데, 이는 전체의 약 12.4%에 해당되며 컴파일 시 최적화 옵션에 따른 바이너리의 코드 차이가 발생할 경우 두 바이너리는 거의 다르다고 볼 수 있을 정도로 변화의 정도가 크다. 또한 제안하는 함수식별 분석 방법도 이에 영향을 받아 360개의 비 식별함수 중 67개가 식별되었다. 마지막으로 컴파일러의 버전에 차이가 생기는 경우 전체 함수의 개수는 3102개이며 비 식별된 함수는 55개로 약 1.8%이다. 최적화 옵션과 달리 컴파일러는 버전에 따른 바이너리 파일의 변화가 크지 않다는 것을 확인할 수 있었으며 제안된 분석 방법을 통해 비 식별된 55개 함수를 대상으로 함수를 식별할 경우 30개의 추가 함수를 찾을 수 있었다.

본 논문에서 평가한 세 가지 방법은 Diaphora에서 생성한 SQLite 파일의 데이터를 기반으로 수행되기 때문에 같은 소스코드라 할지라도 유틸리티 버전, 최적화 옵션, 컴파일러 버전 등의 차이로 인한 바이너리 파일의 변화 정도에 따라 결과가 달라지는 것을 확인할 수 있었다 또한 제안된 함수식별 분석 방법의 평가 결과 컴파일러 버전 > 유틸리티 버전 > 최적화 옵션의 순으로 바이너리 파일의 변화 정도가 적을수록 식별되는 함수의 개수가 많다는 것을 확인할 수 있다.

V. 결론

4장의 평가 결과에 따르면 같은 소스 코드의 바이너리를 컴파일 할 때 최적화 옵션 명령어 여부에 함수의 식별 비율이 현저히 떨어지게 된다. 이를 통해 바이너리는 컴파일 되는 시점의 아키텍처 환경, 컴파일 옵션 등에 민감하기 때문에 우리는 여러 가지 환경에서 제안된 방법과 비슷한 형태의 바이너리 디핑을 수행하여 조금 더 나은 결과를 도출해낼 필요가 있다.

본 연구에서는 임베디드 디바이스 펌웨어의 유사 함수식별에 대한 분석 방법을 새로 제안하였다. 이를 위해 기존 바이너리 디핑 도구인 Diaphora에서 사전 정의된 Heuristic에 일치하지 않아 Unmatched 로 분류된 함수를 대상으로 제안된 방법을 적용하여 다양한 평가를 수행하였고 비 식별된 함수로부터 의미 있는 데이터를 도출해내었다.

이는 임베디드 펌웨어가 새로 업데이트 되었을 경우 1-Day 취약점 탐지를 위해 함수 분석이 필요 할 때 우리는 Diaphora와 제안된 분석 방법을 통해 보다 많은 함수를 식별할 수 있게 된다.

ACKNOWLEDGEMENTS

본 연구는 과학기술정보통신부 및 국가보안기술연구소의 위탁연구개발사업의 연구결과로 수행되었음 (2018-059)

[참고문헌]

[1] Jannik Pewny, Behrad Garmany, Robert Gawlik, Christian Rossow, Thorsten Holz "Cross-Architecture Bug Search in Binary Executables", IEEE Symposium on Security and Privacy, 2015

[2] Andrei Costin, Apostolis Zarras, Aurélien Francillon "Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces", ACM Asia Conference on Computer and Communications Security (ASIACCS), May, 2016.

[3] Daming D. Chen, Manuel Egeley, Maverick Woo, David Brumley "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware", Network and Distributed Systems Security Symposium (NDSS), February, 2016.

[4] Yaniv David, Nimrod Partush, Eran Yahav, "Similarity of Binaries through re-Optimization", ACM SIGPLAN Conference on Programming Language Design and Implementation, June, 2017.

[5] Mnuel Egele, Maverick Woo, Peter Chapman, "Blanket Execution: Dynamic Similarity Testing for Program Binaries and Components", USENIX Security Symposium, August, 2014.

[6] Jeongwook Oh, "Fight against 1-day exploits: Diffing Binaries vs Anti-diffing Binares", Blackhat USA, 2009.

[7] Stephen Sims, "Bruh! Do you even diff? Diffing Microsoft Patches to Find Vulnerabilites", RSA conference, 2016.

[8] Heewan Park, Seokwoo Choi, Sunae Seo, Taisook Han, "SCV: Structure and Constant Value based Binary Diffing", IEEE computer society, 2008.

[9] joxeankoret, diaphora. A binary diffing tool, https://github.com/joxeankoret/diaphora

[10] Zynamics, BinDiff, A binary diffing tool, https://www.zynamics.com/

[11] Hex-rays, The Ida Pro disassembler and debugger ,A reverse engineering tool, https://www.hex-rays.com/products/ida/index.shtml

크롬환경에서의 웹 로그정보를 활용한 개인 성향분석 방법 연구

김지영*†, 박예원*, 안지은*, 양지수*, 임규희*, 김형종*‡
*서울여자대학교 정보보호학과

A Study on Analyzing Method of Personal Tendency by Synthesizing Web Log

Jiyeong Kim*†, Yewon Park*, Jieun Ahn*, Jisu Yang*, Kyuhee Lim*, Hyung-jong Kim*‡
*Division of information security, Seoul women's University.

요 약

디지털 포렌식(Digital forensics)이란 과학적 원리를 사용하여 전자적으로 저장된 정보를 분석하는 프로세스이며 정보보호 침해 사고 조사나 기업 감사 등 전문적 기술로써 활용되었다. 이러한 포렌식 기술은 일반적인 디지털 기기 사용자들에게 활용도가 떨어진다는 한계점이 있다. 따라서 본 연구에서는 일반인을 위해 디지털 포렌식으로 확보된 증거에 대한 분석 활용방안을 고려하고자 하였고 그 중에서도 PC 웹 사용자에게 초점을 맞추었다. 또한, 일반적인 웹 사용자의 정보를 평상시에도 볼 수 있도록 하는 PC 애플리케이션을 그 대안으로 제시한다. 브라우저 중 점유율이 가장 높은 크롬을 중심으로 연구가 진행되었으며 웹 로그 수집에서 더 나아가 웹 사용자의 성향분석까지 제공하는 것이 이 논의의 골자다. 또한 본 논문은 수집된 증거를 바탕으로 제작된 애플리케이션의 활용성과 고려사항, 그리고 기대효과를 기술한다.

키워드 : 포렌식, 웹 포렌식, 사용자 분석, 성향분석, 비전문가(일반 사용자)
Keyword : Forensics, Web forensics, User analysis, Tendency analysis, Layman(Normal user)

I. 서론

디지털 포렌식은 정보기기에 남은 흔적을 분석하여 행위를 증명하는 기술이다. 현재는 상용 디지털 포렌식 도구가 발전하여 일반인도 사용할 수 있을 정도로 사용이 간편해지고 있고[1], 포렌식 툴들의 활용방안에 대한 연구도 이루어지고 있다[2]. 그 중, 웹 브라우저는 사용자들이 인터넷에 많은 흔적을 남기는 만큼 웹 아티팩트에 대한 분석이 필수적이며[3], 이를 이용한 사용자 성향분석 또한 유의미한 결과를 도출해 낼 수 있다.

따라서 본 논문은 디지털 포렌식의 활용범위를 확장시킬 수 있도록 성향분석이라는 대안을 모색하고 그 고려사항과 기대효과를 목표로 제시한다. 구체적으로, 일반적인 PC 웹 사용자로 초점을 맞춘 애플리케이션을 제시하고자 한다. 그에 따라, PC 웹 브라우저 중에서 가장 점유율이 높은 크롬을 중심으로 개발이 진행되었다[4]. 이 애플리케이션은 PC 크롬 사용자의 성향분석과 디지털 포렌식 비전문가도 사용하기 쉬운 여러 기능을 제공하도록 고려되었다.

디지털 포렌식은 데이터의 수집과 분석이 이루어져야 하며, 이 애플리케이션에서 크롬 사용자의 성향분석과 다양한 사용자 편의 기능을 제공하고자 하므로 이에 적합한 데이터의 수집과 분석이 요구된다. 이에 따라 본 논문에서는 먼저 수집한 크롬 데이터의 종류와 이에 대한 설명을 기술한다. 그리고 이 데이터들을 애플리케이션에서 시각적으로 표현한 방식, 애플리케

이션의 사용자 편의 기능, 그리고 성향분석에 대해 서술할 것이다. 또 이에 대한 적용 시 고려사항과 기대효과, 향후연구에 대해 다룬다.

II. 수집한 크롬 로그 정보

크롬의 로그는 SQLite 데이터베이스 파일 형식으로 저장되어 있으며, 이 파일을 통해 수집할 수 있는 데이터에는 Cookie와 History 정보가 있다. 또한, History에서는 PC 웹 사용자가 크롬 상에서 검색한 키워드와 크롬에서 다운로드한 자료 관련 정보도 수집할 수 있다.

2.1 Cookie

Cookie는 웹 사용자가 해당 웹 페이지를 방문했다는 증거이므로 포렌식에 필수적인 존재이다. Cookie의 컬럼(column)중 본 애플리케이션에 사용한 핵심정보는 [표 1]과 같다.

[표 1] cookies 테이블

creation_utc	host_key
Cookie 생성시간	host 정보
name	**path**
Cookie 이름 정보	Cookie 저장 경로
expires_utc	**last-access_utc**
Cookie 만료 시간	마지막 접근 시간
secure	**httponly**
Cookie 보관 안정성	Cookie 접속 가능여부

2.2 History

History는 본 애플리케이션에서 사용자의 성향분석을 위한 가장 중요한 정보로 활용된다. History의 여러 테이블 중 본 애플리케이션에서 사용한 테이블과 컬럼의 정보는 [표 2], [표 3], [표 4], [표 5]와 같다.

[표 2] urls 테이블

url	title
url 주소	url title
visit_count	**last_visit_time**
방문한 횟수	마지막 접근시간

[표 3] visits 테이블

url	visit_time
url 주소	접근한 시간

[표 4] keyword_search_term 테이블

term	url_id
검색한 키워드	url의 id값

[표 5] download 테이블

target_path	start_time
다운로드받은 로컬경로	다운로드 시작시간
total_bytes	**state**
다운로드한 크기	다운로드 한 상태
site_url	
다운로드한 사이트	

이와 같이 수집된 Cookie, History 데이터들을 사용하여 성향분석이 이루어진다.

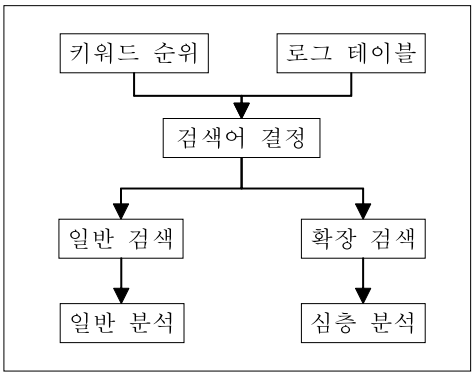
III. 크롬 로그와 성향분석 제공 방안

3.1 크롬 로그 정보 제공 방식

본 애플리케이션은 크롬 로그 중 Cookie, History에 저장되는 로그들을 추출한다. 더해서, History의 keyword_search_term과 download 테이블을 재추출하여 개별적인 데이터로 만들었다. 이들을 각각 Cookie, History, Keyword, Download 네 가지 탭으로 만들어 테이블 형식으로 타임라인에 따라 제공한다.

3.2 성향분석 방법

본 성향분석 모델은 검색 기능을 중심으로 하여 궁극적으로 프로그램 사용자의 성향을 분석하고자 한다. 검색은 일반 검색, 확장 검색의 두 가지로 구분되고 각각의 검색 수준에 따라 일반적인 분석과 심층적인 분석이 이루어질 수 있다. 이러한 분석 과정에서 [그림 1]과 같이 로그정보를 나타내는 테이블과 키워드 순위 또한 참고 될 수 있다.



[그림 1] 분석 과정 도표

3.2.1 일반 분석

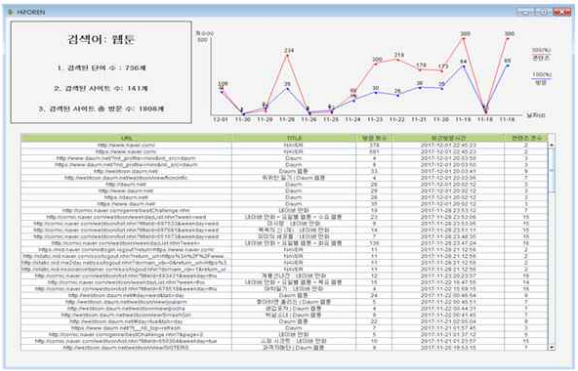
일반 분석은 일반 검색과 키워드 순위를 기반으로 이루어진다. 일반 검색이란 네 가지의 탭 마다 원하는 정보를 검색하는 기능으로, 검색어가 포함되어 있는 크롬 로그 테이블 셀을 하이라이트로 표시한다. 이를 통해 타임라인으로 정리된 로그 정보를 확인하여 해당 검색어가 사용된 흐름을 추적할 수 있다.

키워드 순위는 PC 웹 사용자의 Keyword 내 term에 대한 것이며, term의 빈도수에 따라 정해진다. 또한, 키워드 순위에 나타나는 단어들은 term을 형태소로 분리하여 명사만 추출한 결과 값이다. 이러한 키워드 순위를 통해서 PC 웹 사용자의 관심사를 파악할 수 있으며 일반 검색 기능을 이용할 시, 이 정보를 참고할 수 있다.

3.2.2 심층 분석

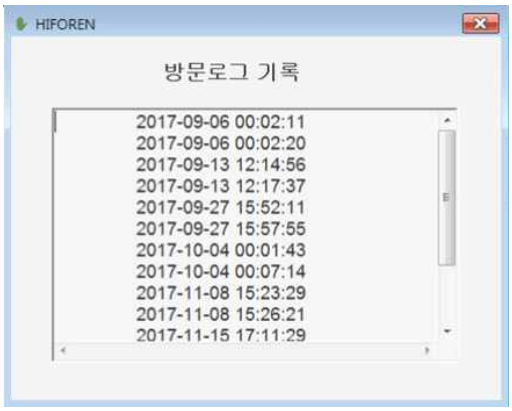
심층 분석은 확장 검색을 기반으로 이루어진다. 단순히 수집된 데이터 리스트 내에서 행해지는 일반 검색과는 달리, 확장 검색은 이에 더해서 방문 사이트의 콘텐츠 내용까지 검색 범위를 넓혀 더 상세한 검색결과를 제공한다.

확장 검색을 통해 검색한 키워드가 방문한 웹 사이트 내 콘텐츠에 포함된 경우, 새로운 화면을 통해 그 결과를 보여준다. 결과 화면은 [그림 2]와 같이 하단, 좌측 상단, 우측 상단 순서대로 검색 결과 테이블, 검색 요약, 그래프의 세 가지 정보로 나뉘어 보여 진다.



[그림 2, 부록 1] 확장 검색 후 화면

[그림 2]의 하단에 위치한 검색 결과 테이블에서는 검색된 리스트를 URL, Title, 방문 횟수, 최근방문시간, 콘텐츠 개수로 구성된 5개의 컬럼으로 제공한다. 그리고 이 테이블의 리스트 중에서 하나의 행을 더블 클릭하면, [그림 3]과 같이 다이얼로그 형식으로 해당 URL의 방문 시간 로그가 제공된다.



[그림 3] 방문 시간 로그 화면

이렇게 출력된 테이블을 기반으로 [그림 2]의 좌측 상단에 위치한 검색 요약이 제공된다. 검색 요약에서는 검색어와 검색된 단어의 총 개수, 검색된 사이트의 수, 사이트들의 총 방문 횟수를 분석하여 간략화한 자료로 제공한다. 마지막으로 [그림 2]의 우측 상단에 있는 그래프를 이용해 시각적으로 날짜 별 사이트 방문 횟수, 사이트 내의 검색어 개수를 보여줌으로써 PC 사용자의 활동을 최근 시간 순으로 한 눈에 파악할 수 있다.

즉, 확장 검색은 로그 정보만으로 파악하기 힘든 정보를 시각화하여 보여줌으로써 심층적인 분석을 용이하게 할 수 있다.

3.3 활용방안

디지털 흔적을 수집하는 포렌식 기술을 바탕으로 한 성향분석은 여러 분야에서 활용될 수 있다. 예를 들어, 성향분석을 통해 행위를 예측함으로써 사전에 사회적 문제를 방지할 수 있다. 이 같은 사회적 문제에는 자살, 학교폭력, 우울증 등이 해당된다. 한 예로, 이 예시와 관련된 단어의 키워드 순위가 높고, 확장 검색 화면의 그래프에 관련 단어가 최근 기록으로 남아 있는 등의 로그가 존재한다고 가정한다. 이를 사전에 성향분석을 통해 확인할 수 있다면 이후의 행위를 예측하고 사전에 조치를 취할 수 있을 것이다. 이처럼 성향분석은 사회 공헌적 기능을 수행하는 등 여러 방안으로 활용될 수 있다.

IV. 적용 시 고려사항

본 애플리케이션은 크롬 외의 브라우저에서 이루어진 행위에 대해서는 성향분석을 제공하지 않는다. 따라서 PC 웹 사용자에 대한 성향분석은 그 사용자가 주로 사용하는 브라우저에 따라 신뢰도가 상이할 수 있다. 또한 이 애플리케이션의 정보 열람 권한 등에 대해서 보안의 고려가 필요하다.

V. 결론

본 연구에서는 크롬의 로그를 활용한 성향분석 모델을 제안하였다. 제안된 모델은 개개인의 관심사 분석, 그리고 더 나아가 행위예측을 통해 사회적 문제에 대한 방안을 마련하기 위한 지표로 활용될 수 있다. 향후 심층 분석을 구체화 시켜 정확한 행위예측 결과를 이끌어 낼 것이며, 여러 행동에 대한 패턴 도출 방법을 모색할 것 이다.

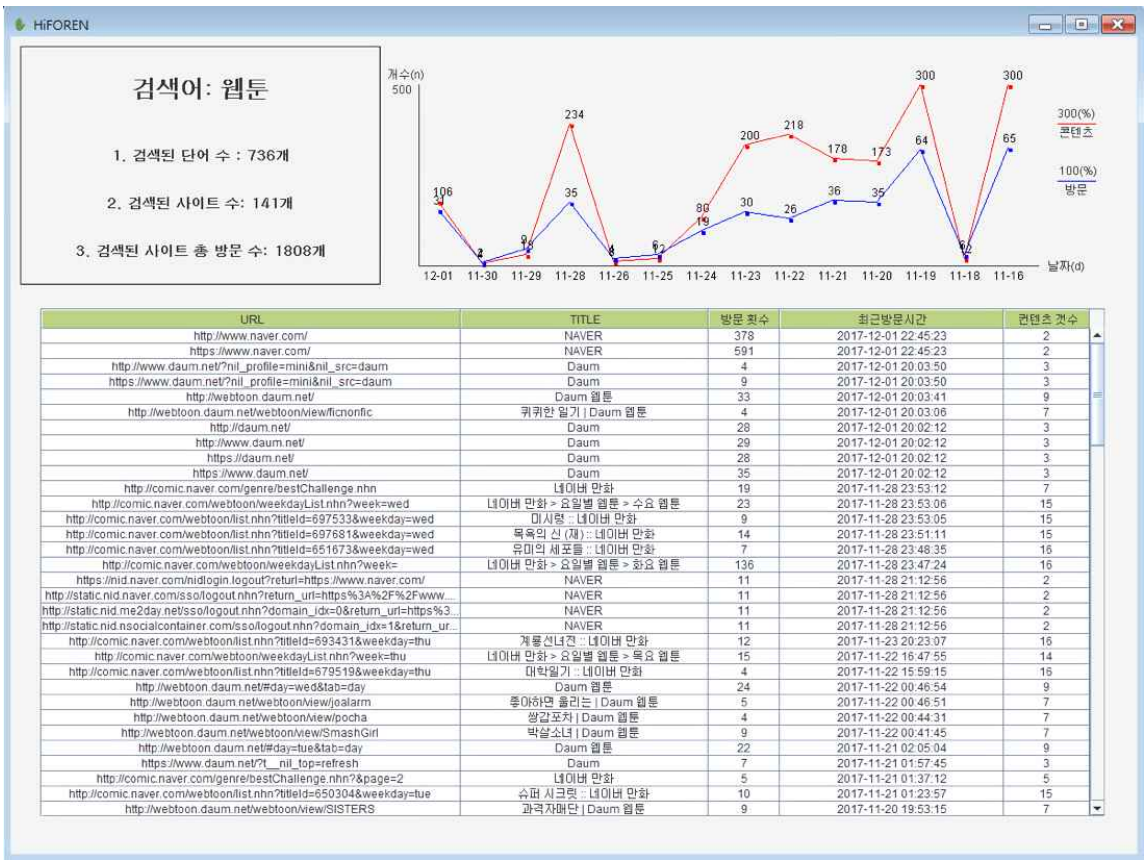
[참고문헌]

[1] 조병철. "디지털 포렌식 직무의 효율 재고방안 고찰." 디지털포렌식연구 9.1 (2015): 89-98.

[2] 정익래, 홍도원 , 정교일 , 암호기술연구팀. "디지털 포렌식 기술 및 동향." 전자통신동향분석, (2007): 97-104

[3] 서미나. 웹 브라우저의 포렌식 분석 기법 비교 연구. Diss. 서울대학교 대학원, 2018.

[4] Statcounter, "Browser Market Share Worldwide", 2017.9-2018.9
URL : http://gs.statcounter.com/



URL	TITLE	방문 횟수	최근방문시간	컨텐츠 갯수
http://www.naver.com/	NAVER	378	2017-12-01 22:45:23	2
https://www.naver.com/	NAVER	591	2017-12-01 22:45:23	2
http://www.daum.net/?nil_profile=mini&nil_src=daum	Daum	4	2017-12-01 20:03:50	3
https://www.daum.net/?nil_profile=mini&nil_src=daum	Daum	9	2017-12-01 20:03:50	3
http://webtoon.daum.net/	Daum 웹툰	33	2017-12-01 20:03:41	9
http://webtoon.daum.net/webtoon/view/ficnonfic	퀴퀴한 일기 | Daum 웹툰	4	2017-12-01 20:03:06	7
http://daum.net/	Daum	28	2017-12-01 20:02:12	3
http://www.daum.net/	Daum	29	2017-12-01 20:02:12	3
https://daum.net/	Daum	28	2017-12-01 20:02:12	3
https://www.daum.net/	Daum	35	2017-12-01 20:02:12	3
http://comic.naver.com/genre/bestChallenge.nhn	네이버 만화	19	2017-11-28 23:53:12	7
http://comic.naver.com/webtoon/weekdayList.nhn?week=wed	네이버 만화 > 요일별 웹툰 > 수요 웹툰	23	2017-11-28 23:53:06	15
http://comic.naver.com/webtoon/list.nhn?titleId=697533&weekday=wed	미시령 :: 네이버 만화	9	2017-11-28 23:53:05	15
http://comic.naver.com/webtoon/list.nhn?titleId=697681&weekday=wed	목욕의 신 (재) :: 네이버 만화	14	2017-11-28 23:51:11	15
http://comic.naver.com/webtoon/list.nhn?titleId=651673&weekday=wed	유미의 세포들 :: 네이버 만화	7	2017-11-28 23:48:35	16
http://comic.naver.com/webtoon/weekdayList.nhn?week=	네이버 만화 > 요일별 웹툰 > 화요 웹툰	136	2017-11-28 23:47:24	16
https://nid.naver.com/nidlogin.logout?returl=https://www.naver.com/	NAVER	11	2017-11-28 21:12:56	2
http://static.nid.naver.com/sso/logout.nhn?return_url=https%3A%2F%2Fwww...	NAVER	11	2017-11-28 21:12:56	2
http://static.nid.me2day.net/sso/logout.nhn?domain_idx=0&return_url=https%3...	NAVER	11	2017-11-28 21:12:56	2
http://static.nid.nsocialcontainer.com/sso/logout.nhn?domain_idx=1&return_ur...	NAVER	11	2017-11-28 21:12:56	2
http://comic.naver.com/webtoon/list.nhn?titleId=693431&weekday=thu	계룡선녀전 :: 네이버 만화	12	2017-11-23 20:23:07	16
http://comic.naver.com/webtoon/weekdayList.nhn?week=thu	네이버 만화 > 요일별 웹툰 > 목요 웹툰	15	2017-11-22 16:47:55	14
http://comic.naver.com/webtoon/list.nhn?titleId=679519&weekday=thu	대학일기 :: 네이버 만화	4	2017-11-22 15:59:15	16
http://webtoon.daum.net/#day=wed&tab=day	Daum 웹툰	24	2017-11-22 00:46:54	9
http://webtoon.daum.net/webtoon/view/joalarm	좋아하면 울리는 | Daum 웹툰	5	2017-11-22 00:46:51	7
http://webtoon.daum.net/webtoon/view/pocha	쌍갈포차 | Daum 웹툰	4	2017-11-22 00:44:31	7
http://webtoon.daum.net/webtoon/view/SmashGirl	박살소녀 | Daum 웹툰	9	2017-11-22 00:41:45	7
http://webtoon.daum.net/#day=tue&tab=day	Daum 웹툰	22	2017-11-21 02:05:04	9
https://www.daum.net/?t__nil_top=refresh	Daum	7	2017-11-21 01:57:45	3
http://comic.naver.com/genre/bestChallenge.nhn?&page=2	네이버 만화	5	2017-11-21 01:37:12	5
http://comic.naver.com/webtoon/list.nhn?titleId=650304&weekday=tue	슈퍼 시크릿 :: 네이버 만화	10	2017-11-21 01:23:57	15
http://webtoon.daum.net/webtoon/view/SISTERS	과격자매단 | Daum 웹툰	9	2017-11-20 19:53:15	7

[부록 1]

OpenSSL에서 제공하는 의사난수 API 구조 분석 연구

김현기*, 송행권*, 이옥연*
*국민대학교 금융정보보안학과

A study on Pseudo-Random Number API Structure analysis provided by OpenSSL

Hyun-Ki Kim*, Haeng-Gwon Song*, Ok-Yeon Yi*
*Division of Financial information security, Kookmin University.

요 약

OpenSSL은 통신프로토콜의 한 종류로서 TCP/IP계층에서 사용하는 보안 프로토콜 SSL(Secure Socket Layer)을 구현해 놓은 오픈소스 라이브러리이며 현재 다양한 곳에서 응용되고 있다. 본 논문에서는 OpenSSL에서 암호 키 생성 및 프로토콜의 Challenge/Response 등 난수 생성이 요구될 때 OpenSSL에서 제공하는 난수 생성 프로세스를 분석하여 응용 프로그램 개발자 및 OpenSSL 사용자가 암호학적 난수 생성 시 고려해야 할 사항을 제시한다.

I. 서론

신 ICT 환경으로 발전함에 따라 통신의 환경 및 지원하는 서비스 등이 다양해지며 통신의 양이 증가하고 있다. 이 때 송수신되는 정보는 개인의 비밀정보들도 포함되어 있다. 이를 보장하기 위해 신뢰성 및 기밀성을 제공하는 보안 프로토콜을 적용하는 것은 중요한 요소가 되었다.

통신 프로토콜 중에 적용되는 대표적인 보안의 종류로는 개체 인증 프로토콜 및 암호키 생성 프로토콜 등이 있다. 이러한 프로토콜에서 가장 핵심적인 요소는 난수를 생성하는 것이다. 보안 프로토콜을 설계할 때 프로토콜을 진행할 개체에서 암호학적으로 안전한 난수 생성이 가능함을 일반적으로 가정하고 시작한다. 하지만 보안 적용환경에서 암호학적으로 안전한 난수를 생성하는 것은 쉬운 일이 아니다.

OpenSSL은 통신프로토콜의 한 종류로서 TCP/IP계층에서 사용하는 보안 프로토콜 SSL(Secure Socket Layer)을 구현해 놓은 오픈소스 라이브러리이며 현재 다양한 곳에서 응용되고 있다.

본 논문에서는 OpenSSL에서 암호 키 생성 및 프로토콜의 Challenge/Response 등 난수 생성이 요구될 때 OpenSSL에서 제공하는 난수 생성 프로세스를 분석한다.

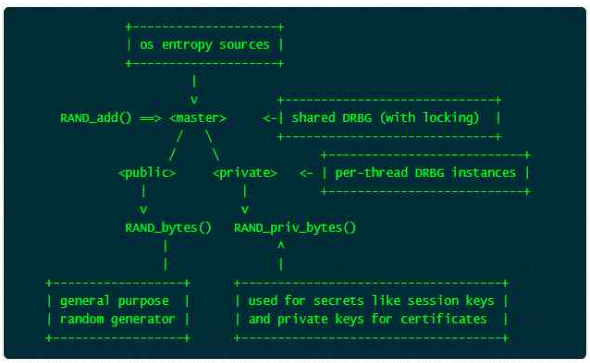
II. OpenSSL에서의 난수 API

OpenSSL 1.1.1 버전의 난수 생성 알고리즘은 [NIST SP800-09A release Ver.1]의 암호학적으로 안전한 의사난수발생기(Cryptographically Secure Pseudo Random Number Generator, CSPRNG)를 사용한다. OpenSSL에서 사용하는 난수 API는 두 가지의 종류로 구분할 수 있다. 그 구분의 기준은 OpenSSL 영역 바깥의 응용 프로그램 개발자가 사용 할 수 있는 'Front-End' 난수 API와 OpenSSL 영역 내의 개발자에게 OS 엔트로피 소스의 접근을 가능하

게 하는 'Back-End'난수 API로 나누어진다. 본 논문에서는 OpenSSL에서 제공하는 'Front-End' 난수 API의 구조를 분석하여 응용 프로그램 개발자가 OpenSSL에서 난수를 사용할 때 암호학적인 용도로 사용하기 위해 필요한 부분을 명시한다.

OpenSSL에서 제공하는 'Front-End' 난수 API는 난수의 쓰임새에 따라 공개되어도 무관한 난수와, 비공개로 유지되어야 할 난수 크게 두 가지로 분류하여 DRBG를 정의하고 있다. 공개되어도 무관한 난수를 생성할 때에는 <Public> DRBG 방식으로써 RAND_bytes() API를 사용하고 비공개로 유지되어야 할 난수를 생성할 때에는 <Private> DRBG 방식으로써 Rand_priv_bytes()를 사용한다. <Public> 또는 <Private>DRBG에 난수성(씨드)을 제공하기 위해 <Master> DRBG 방식을 사용하고 있으며, 이는 운영체제 자체에서 잡음원을 수집하거나 OpenSSL 운영 환경에서 잡음원을 수집하여 엔트로피 풀을 갱신하고 씨드를 생성한다.

다음 그림은 OpenSSL에서 DRBG가 운영되는지에 대한 전반적인 구조도를 나타낸 그림이다.



[그림 1] OpenSSL의 DRBG 구조

2.1 <Public> DRBG: Rand_bytes()

<Public> DRBG는 OpenSSL에서 암호학적으로 공개되어도 무관한 난수를 생성하는 용도로 사용된다. OpenSSL 라이브러리는 기본적으로 정의하고 있는 DRBG와, 사용자가 별도로 정의하여 OpenSSL 서비스와 결합해 사용할 수 있는 DRBG 총 두 가지 형태의 DRBG를 제공한다. 후자의 경우 사용자가 정의하여 생성한 난수열을 사용하여 발생하는 취약점은 전적으로 사용자의 책임이다. <Public> DRBG의 경우, Rand_bytes()라는 Front-End API를 통해 함수를 호출할 때 요구하는 바이트 길이만큼 의사난수를 생성하며 이 때 로컬 스레드를 생성하여 <Private> DRBG와 논리적으로 분리 후 사용한다.

2.2 <Private> DRBG: Rand_priv_bytes()

<Private> DRBG는 OpenSSL에서 암호학적으로 비공개되어야 하는 난수를 생성하는 용도로 사용된다. <Public> DRBG와 동일하게 기본으로 정의하고 있는 DRBG와, 사용자가 추가로 정의하여 OpenSSL 서비스와 결합해 사용할 수 있는 DRBG의 형태를 제공하며 후자의 API를 사용하여 발생하는 취약점은 사용자의 책임이다. <Private> DRBG는 RAND_priv_bytes()라는 Front-End API를 통해 함수를 호출하며 로컬 스레드를 생성하여 <Public> DRBG와 별도의 파라미터로 출력 값을 생성한다.

2.3. <Public> DRBG와 <Private> DRBG의 차이

사용자가 DRBG 관련 함수를 추가로 정의하지 않았을 경우에 OpenSSL에서 난수 사용하기 위해 제공하는 두 개의 API는 구조적으로 동일한 형태를 가진다. <Public> DRBG로 난수를 생성하면 Random_bytes()를 호출하며 내부의 RAND_DRBG_get0_public()을 통해 기본적으로 제공하는 DRBG API 집합과 매핑되어 의사난수열을 출력한다. 또한 <Public>DRBG의 경우와 동일하게 사용자에 의한 별도의 함수 추가 매핑이 없는 상태로 <Private> DRBG로 난수를 생성하면 Random_priv_bytes()를 호출하며 내부의 RAND_DRBG_get0_private()을 통해 기본적으로 제공하는 DRBG API 집합과 매핑되어 의사난수열을 출력한다. 위에서 언급한 두가지 경우의 API RAND_DRBG_get0_public()와 RAND_DRBG_get0_private()는 서로 다른 함수명을 가지고 있지만 내부적인 논리는 정확히 동일하며, 유일한 차이는 각각의 API 호출 시

스레드를 분리하여 사용한다는 점이다.

<Public>DRBG (RAND_bytes())
Session ID 생성, Challenge/Response 진행
인증용 nonce 생성
mac용 salt 생성
암호화용 IV 생성
임의의 Big Number 생성
서명용 난수 생성(DSA의 q)
<Private>DRBG (RAND_priv_bytes())
SSL 통신용 master 키 생성
TLS 통신용 premaster_secret 생성
블록암호 키 생성
서명용 nonce 생성(DSA)
private 모드용 Big Number 생성

[표 2] OpenSSL에서의 <Public/Private> DRBG 사용처

<public> DRBG 사용에 있어서 내부 상태 및 일말의 기타 정보 등이 누출된다고 해도 해당 정보가 <private> DRBG에는 영향을 미치지 않도록 설정하기 위함이다.
OpenSSL은 난수생성에 있어서 내부적인 큰 차이점 없이 <Public>과 <Private> API를 구분하여 응용프로그램에서 사용한다. 다음 표는 OpenSSL 상에서 각 <Public/Private> DRBG가 일반적으로 사용되는 상황을 나타낸다.

III. OpenSSL에서 제공하는 의사난수 알고리즘 분석

NIST의 SP800-90A[1]는 암호 시스템에서 사용할 수 있는 검증대상 DRBG 알고리즘 및 상태 갱신 방법(Reseed), DRBG가 지원해야하는 보안 강도(Security Strength)를 정의하고 있다. SP800-90C[2]는 암호 시스템에서 난수발생기(RBG)를 운용할 때 지켜져야 할 예측 내성 등의 권고사항을 정의하고 있다.

현재 OpenSSL에서의 난수발생은 SP800-90A의 AES-CTR-DRBG(AES-128/192/256-CTR)을 기반으로 이루어진다. 90A에 의하면 AES-CTR-DRBG를 사용할 시, 정의된 보안 강도만큼 지원할 수 있어야 하며, 90C에 의하면 DRBG는 상태 갱신(리씨드) 혹은 예측 내성을 가지기 위해 검증된(Approved) 엔트로피 소스를 입력받아야 받아야 한다. 하지만 OpenSSL에서 지원하는 DRBG API는 검증된 엔트로피 소스를 입력받지 않고 있으며, 운영체제 및 운영환경에서 지원하는 엔트로피 소스를 사용하고 있기 때문에 충분한 엔트로피를 공급받고 있는지를 보장할 수 없다. 따라서 OpenSSL 내의 보안 강도 확인 프로세스 및 예측내성 활성화를 확인하는 부분은 비워져 있는 상태 이거나, 실패를 반환하도록 구현되어 있다.

또한 DRBG 상태 갱신에 포함시키기를 권고하고 있는 개인화 문자열(Personalization String)은 사용자가 입력 시마다 임의로 변경을 해주어야 하는 문자열이지만 별다른 설정 없이는 "OpenSSL NIST SP 800-90A DRBG"라는 문자열로 고정되어 있어 난수성을 제공하는 데에 큰 영향을 주지 못한다. 게다가 OpenSSL을 사용하는 모든 운영 환경에서 스레드 분리만으로 상호 간의 영역이 완벽한 독립성을 보장받는다고 할 수는 없기 때문에 <Public> 과 <private> DRBG의 구분이 효과가 없을 가능성도 배제할 수 없다.

설계상 지원 가능한 최대 보안 강도	사용상 적용 가능한 최대 보안 강도
112-bit	112-bit
128-bit	112,128-bit
192-bit	112,128,192-bit
256-bit	112,128,192,256-bit

[표 3] DRBG가 지원하는 보안강도

IV. 결론

현재 OpenSSL 라이브러리는 프로토콜 진행 과정 및 보안매개변수 생성 등 다양한 곳에서 난수를 생성하고 있다. 하지만 내부적으로 구현되어 있는 알고리즘 및 논리 구조는 권고하고 있는 검증된 엔트로피 소스와, 검증된 DRBG 알고리즘을 사용하고 있지 않으므로 OpenSSL 라이브러리를 통해 서비스를 이용할 사용자는 난수 생성을 무분별한 신뢰는 금물이다. 따라서

사용자는 보안매개변수 생성 시 OpenSSL 내에서의 난수 생성 매커니즘을 정확히 파악하고 난수 생성 관련 API를 호출해야하며, 가능하다면 사용자가 난수 생성 관련 API에 난수성을 보완할 수 있도록 추가로 매핑하여 사용하는 것이 바람직하다.

[사사 정보]

본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

[참고문헌]

[1] NIST, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90Ar1, June, 2005

[2] NIST, Recommendation for Random Bit Generator (RBG) Constructions, Special Publication 800-90C, April, 2016

[3] OpenSSL Cryptography and SSL/TLS Toolkit, Final version of OpenSSL 1.1.1, https://www.openssl.org/

다크웹 악성유저 추적방안에 대한 연구

안상현,유동민,정주현,김만수,이유진,김영웅,이해민

*BestOftheBest Dark Knigh7

A Study on Dark Web Malicious User Tracking

*Sanghyun Ahn, Dongmin Yoo, JuHyeon Jeong, Mansu Kim, Youjin Lee, Yeongwoong Kim, Haemin Lee

*BestOftheBest Dark Knigh7.

요 약

전 세계적으로 다크웹 상에서의 불법 행위가 증가하고 있다. 다크웹은 강력한 익명성을 제공하며, 여러 국가의 노드를 거치기 때문에 악성유저를 식별할 수 있는 실제 IP정보를 노출하지 않는다. 따라서 다크웹에서 범죄활동에 효과적으로 대응하기 위해서는 각 국가 수사기관들의 국제 공조가 필수이지만, 실제로는 국제 공조 수사의 어려움이 있기 때문에 다른 검거수단이 필요한 실정이다. 다크웹 범죄에 대응하기 위해 수사관은 실시간으로 발생하는 범죄에 대해서 모니터링을 해야하며, 식별한 범죄에 대해서는 다크웹 악성유저와 표면웹과의 OSINT를 통한 분석, 토르브라우저의 취약점 그리고 다크웹 사이트의 취약점을 이용하여 악성유저의 실마리를 제공할 수 있다.

I. 서론

최근, 전 세계적으로 다크웹 상에서의 불법 행위가 증가하고 있다[1]. 이는 다크웹의 가장 큰 특징인 '익명성' 때문인데, 보통 토르 네트워크로만 접근 가능한 웹은 16바이트의 주소와 onion 확장자를 이용한 주소를 할당받게 된다.[2] 여러 프록시를 거치는 특성상, 각 단계별 통신 과정을 거치는 과정에서 복호화해내지 않는 이상 접속자의 IP나 정보를 파악하기가 어려운데, 토르 네트워크의 이러한 특성을 이용하여 범법자들은 자신의 신분을 숨긴 채 쉽게 불법적인 행위를 할 수 있다.[3]

현재 다크웹에서는 살인청부나 마약, 무기밀매, 불법 포르노, malware 등의 거래가 암암리에 이루어지고 있는데[3], 앞서 설명했듯이 다크웹의 특성 자체가 '뛰어난 익명성'이기 때문에 전 세계적으로 수사기관들은 해당 내용에 대한 수사에 어려움을 겪고 있다[4].

최근 국내에서도 다크웹을 이용한 마약거래 및 유통이 적발되었으며, 2018년 초 한국인 아동음란물 사이트 운영자가 미국 수사기관과 공조를 통해 체포되었다[5]. 국내 언론에서도 다크웹에서의 범죄활동에 대해 심각하다고 보도가 많이 되고 있지만[6] 지금까지도 다크웹내 범죄사이트들이 차단되지 않고 지속적으로 운영되고 있는 실정이다. 다크웹을 이용한 범죄에 가장 효과적인 대응을 하기 위해서는 국제협력이 필수조건이다. 토르 네트워크의 특징상 사용자의 실제IP를 추적하려면 여러 국가와 국제 공조가 필요해 실질적으로 추적이 어려운 실정이다. 다크웹상에서 범죄활동에 효과적으로 대응하기 위해서는 각 국가의 수사기관들의 국제 공조외에 다른 검거수단이 필요하다.[7]

이 연구에서는 다크웹의 동작원리를 설명하며, 다크웹내 수사나 추적의 어려움에 대해 논하고 범죄유저를 색출하고 추적하기 위한 방법론을 제시한다. 논문에서 제시할 방법론의 검증은 차후에 실시할 예정이다.

1.1 다크웹의 정의

다크웹에 대하여 논의하기 전, 먼저 딥웹 다크웹 표면웹에 대한 차이를 설명할 필요가 있다. 딥웹은 익스플로러,크롬 등 인터넷 브라우저를 이용하여 검색엔진을 이용하는 방법으로 접근할 수 없는 개인 이메일, 온라인 뱅킹, 회사 내부망 등을 모두 포함하는 개념이다. 반면에 검색엔진을 통하여 접근가능한 자원을 표면웹(Surface Web)으로 정의하고 있다[8].

다크 웹(dark web)은 공공 인터넷을 사용하지만 접속하기 위해 Tor브라우저와 같은 특정 소프트웨어를 사용해야만 하는 월드 와이드 웹 컨텐츠이다.[9]

1.2 국내 다크웹 현황

2016년 정부 발표에 의하면 불법 거래, 정보 유출 등이 이뤄지는 다크웹 사이트가 1,574개에 달하는 것으로 확인되었다고 하였다. 이 숫자는 토르(Tor)브라우저를 통해 접속 가능한 다크웹 불법사이트의 숫자이다. 그리고 우리나라에서 접속하는 규모는 6000명에서 많으면 1만명 수준으로 전세계 20위권 수준이며, 불법금융와 마약거래이 각각 327개와 423개로 가장 많은 비중을 차지하였고, 극단주의(140개)·해킹(96개)·무기거래(42개) 등의 비중도 큰 걸로 확인되었다.

1.3. Tor브라우저의 특징, 문제점

일반적인 인터넷 통신은 사용자의 PC가 ISP를 통해서 목적지에 연결해서 데이터를 주고 받는 식으로 이루어진다. 이 경우 트래픽을 전담하는 중계자가 국가 또는 준 국가 기관이기 때문에 쉽게 검열과 감청을 할 수 있다. 그러나 토르 브라우저를 사용하면 토르 디렉터리 서버(Tor directory server)로부터 전 세계에 존재하는 토르 라우터로 등록된 서버 목록은 전달받아 그 서버 중 임의로 선택한 3개의 서버를 노드로 사용하여 토르 서킷(Tor circuit)을 형성한다. 토르 서킷을 형성할 때에는 입구노드로부터 출구노드까지 한 개씩 노드를 늘려가며 각각의 노드의 공개키를 이용하여 각각의 노드와 별개로 협상된 대칭키를 암호화하여 공유하고, IP 주소를 암호화한다. 사용자는 위와 같이 협상한 노드로 인터넷 패킷을 출구 노드의 키로 시작하여 입구 노드의 키의 순서로 중첩하여 암호화하며, 암호 층은 한 개씩 순서대로 각각 노드에 의하여 복호화 된다.[11] 따라서 패킷의 출발지와 최종 목적지를 알아내려면 거의 모든 노드를 장악해야한다. 이 처럼 토르브라우저는 강력한 익명성을 제공하며, 악성유저를 식별할 수 있는 실제 IP정보를 얻는데 어려움이 있다.[12]

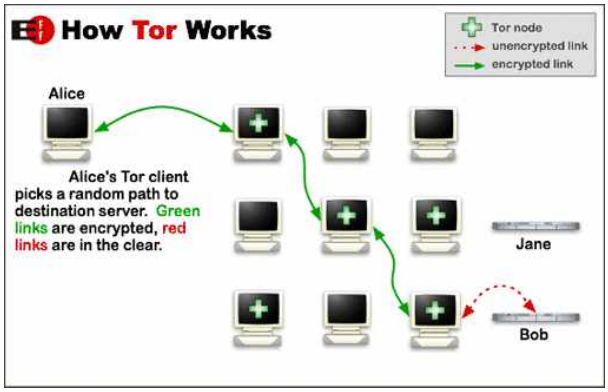


Fig 1. How Tor Works

1.4 국내 다크웹 범죄자 식별 현황

우리나라의 경우 '국회입법조사처'에 따르면 다크웹의 정보는 정보통신망을 이용하는 관점에서 '정보통신망 이용촉진 및 정보보호 등에 관한 법률' 제44조의7 제2항과 제3항에 근거하여 방송통신심의위원회의 심의를 거쳐 다크웹에 유통되는 불법 정보의 삭제 및 접속 차단을 명령할 수 있다. 그러나 방송통신위원회의 경우 주로 표면웹에 대한 불법 정보 차단에 집중하고 있으며, 다크웹에 유통되는 불법 정보에 대해서는 유출된 정보가 표면웹에 노출되는 경우에 해당 정보를 표면웹에서 차단하는 수준으로 대응하고 있다. 또한 2018년 경찰청 사이버수사 시스템 구축안을 보면 다크웹 범죄에 대응하기 위한 내용은 포함하지 않고 있다. [13]

1.5 다크웹 범죄 대응책

1),다크웹 모니터링을 통한 불법행위 탐지

다크웹상 불법행위를 즉각적으로 적발하기 위해서는 다크웹을 실시간으로 모니터링하는 소프트웨어가 있어야 한다. 2018년 대전대학교에

서 4만 2천여 건의 개인 정보가 유출된 사건이 있었다.[14] 2018년 3월 5일 경 다크웹 토르 사이트에 홈페이지를 만들어 학생 등 이름을 넣어 검색하면 검색 결과가 '목록' 형태로 화면에 출력되는 방법으로 개인정보를 누설했다. 5월 8일 대학교 관계자의 신고로 발각되었으며, 약 2개월간 다크웹에 업로드된 개인정보를 아무도 확인 인지하지 못하였다. 또한 Tor 브라우저의 경우, 사이트 생성 및 폐쇄의 용이성으로 인하여, 몇몇 불법거래사이트는 일시적으로 페이지를 생성한 뒤 판매가 완료되면 바로 폐쇄한다. 이 같은 문제점으로 보았을 때 다크웹 크롤링을 통하여 다크웹에 대한 주기적인 모니터링을 실시하고 식별되지 않은 사이버범죄를 실시간으로 인지하고 대응할 필요가 있다.

2) OSINT를 이용한 다크웹 사용자 추적

다크웹 악성유저의 식별을 위해 공개출처정보(OSINT) 프레임워크를 활용할 수 있다. OSINT란, 공개출처정보(Open Source INTelligence)로 공개된 출처에서 얻은 정보를 뜻한다. 수사관은 오픈소스 및 상용 소프트웨어 도구를 활용하여 블로그, 소셜미디어 플랫폼, 정부웹사이트, 표면웹, 다크웹 등의 다양한 출처의 데이터를 분석하고 정보를 확인하여 사이버 범죄를 식별 할 수 있으며, 신뢰할 수 있는 데이터를 선정하고 비트코인,위치,계정,이메일주소, IP주소 및 생활 데이터를 확인하여 인터넷 신원을 확인 할 수 있다. 다크웹 사용자들 또한 표면웹에서는 가면을 쓰지 않고 활동하기 때문에 다크웹과 표면웹간의 OSINT를 활용하여 연관성을 분석을 할 수 있다면, 다크웹 범죄자에 대한 중요한 실마리를 제공할 수 있을 것이다.

3) 취약점을 이용한 다크웹 운영자 추적

다크웹은 프로그램 취약점을 이용한 공격방법을 이용하여 다크웹 운영자를 추적 할 수 있다. 취약점이란 권한 없는 자에 의하여 노출 될 수 있는 잠재적인 시스템의 약점을 의미하는데.
취약점은 프로그래밍 상의 결함이나 버그를 의미할 수도 있고, 사용자의 실수로 생긴 정보유출이 될 수도 있다. 이러한 응용프로그램 단계의 취약점은 비단 토르네트워크 상이 아닌 일반적인 웹 브라우저 환경에서도 실제 발생할 수 있는 문제들이고, 대부분의 경우 이러한 취약점을 이용한 추적 방법의 경우에는 취약점을 이용한 해킹 공격이 수반될 수 있다. 현재 많은 표면웹상에 동작하는 취약점 점검 도구들은 다크웹을 대상으로 서비스를 하고 있지 않다. 다크웹페이지와 표면웹상의 차이는 단순히 IP를 몇 단계 우회하는 것 일뿐 웹페이지의 구조는 동일하다. Tor 브라우저를 실행한 후 아이피를 127.0.0.1, 포트를 9150으로 프록시 설정을 하면 다른 프로그램에서도 토르 네트워크를 사용해 통신 할 수 있다.(socks5 프록시) 위 방법을 적용하여 표면웹상에 동작하는 웹 취약점 점검도구를 활용해 다크웹 범죄사이트에 대한 웹 취약점 점검결과를 알 수 있으며 그 결과를 바탕으로 웹 해킹을 통한 운영자의 식별 또는 불법정보를 차단을 하기 위한 SQL인젝션, 디렉토리 리스팅, XSS, 웹쉘 업로드 등의 방법을 이용하여 사용자 본인의 실제 IP주소를 콘텐츠로서 누군가에게 보내준다거나, 멀웨어에 감염되어 토르네트워크를 거치지 않고 특정 서버 에 접속하는 등의 실제 IP주소를 특정할 수 있을 것이다. 적법성 문제를 논외로 하고 기술적인 측면에서 보았을 때 실제 토르네트워크를 추적한 사례들은 대부분 이러한 취약점을 이용한 것이므로 이에 대해 자세히 확인 할 필요성이 있다. 이외에 "How to Find Hidden Users: A Survey of Attacks on Anonymity Networks"에서는 응용프로그램단 네트워크단으로 나누어 다크웹 범죄자 추적방법에 대한 여러 가지 설명을 하고 있으며, 네트워크 레벨에서의 추적방법의 경우에는 주로 토르네트워크 에서 사용되는 노드를 다수 장악하여 트래픽을 모니터링 패킷의 패턴, 시간 차를 분석하여 실제 히든서비스를 추적하는 방법 등이 제시 되어있다.[15]

II. 결론

본 논문에서는 국내에서 다크웹을 이용한 범죄가 지속적으로 발생하면서, Tor브라우저의 익명성과 다크웹상 범죄자 식별의 어려움을 설명하며, 전통적인 연구들이 보여주듯이 양적,질적

연구를 통한 결과 보다는 다크웹 모니터링, OSINT, 취약점을 통해서 범죄유저를 색출하고 추적하기 위한 방법론을 제시한다. 논문에서 제시한 방법론의 검증은 차후에 기능을 구현하여 실제 사례를 통해 검증하고자 한다.

[참고문헌]

[1]이석원,http://www.techholic.co.kr/news/articleView.html?idxno=57350, TechHolic,2016.8

[2]임성희,"다크웹과 표면웹간의 연관성 분석 BestoftheBest", 2017

[3]이대영,http://www.itworld.co.kr/news/105885, ITworld, 2017.08

[4]임성희,"다크웹과 표면웹간의 연관성 분석"BestoftheBest, 2017

[5]김경애,https://www.boannews.com/media/view.asp?idx=68989,보안뉴스,2018.05

[6]김연주,http://www.sedaily.com/NewsView/1OL6UMGZ5I,서울경제,2017.09

[7]정태지, "한국경찰연구 제17권 제1호, 다크웹 범죄활동 동향: 국제공조 및 대응전략" 한국경찰연구학회, 2018.03

[8]MICHAEL K. BERGMAN, "The Deep Web: Surfacing Hidden Value", BrightPlanet, 2001

[9]Solomon, Jane . "The Deep Web vs. The dark web", 2015.05

[10]정태지, "한국경찰연구 제17권 제1호, 다크웹 범죄활동 동향: 국제공조 및 대응전략" 한국경찰연구학회, 2018.03

[11]정태지, "한국경찰연구 제17권 제1호, 다크웹 범죄활동 동향: 국제공조 및 대응전략" 한국경찰연구학회, 2018.03

[12]홍은표, "토르(Tor)와 익명으로 읽을 권리, 한국정보법학회, 제21권 제3호" , 2017

[13]최진응,"다크웹(Dark Web)상 사이버 범죄 정보 유통 현황 및 대응 방안",국회입법조사처 2017.11

[14]김경애,https://www.boannews.com/media/view.asp?idx=69215 ,보안뉴스,2018.05

[15]이경빈 "토르 히든서비스 서버 분석 사례 및 추적기법에 관한 연구",경찰청,2018

안드로이드 환경에서의 O2O 서비스 현황 및 보안이슈

문현지*, 주완빈*, 신지호*, 현지산*, 정성조*, 조민재*, 박세한*, 이성권**
*KITRI Best Of the Best
**고려대학교 정보보호대학원

O2O Service Status and Security Issue in Android Environment

Hyun-Ji Moon, Wan-Been Joo, Ji-Ho Shin, Ji-San Hyun,
Sung-Jo Jung, Min-Jae Jo, Se-Han Bak, Sung-Kwon Lee**
*KITRI Best Of the Best, Korea Information Technology Research Institute
**College of Informatics, Korea University

요 약

국내 스마트 폰 시장에서 O2O 서비스 기반 애플리케이션(이하 앱)은 높은 사용률을 보이고, 우리 생활 깊숙이 자리 잡고 있다. 하지만, 개인정보 등 중요정보의 유출 사건으로 안드로이드 앱 보안성에 대한 요구사항이 늘어나고 있다. 본 논문에서는 O2O 서비스 기반 애플리케이션이 가지고 있는 보안상의 문제점은 어떤 것들이 있으며, 실제 취약점 분석 사례와 보안 이슈들을 제시한다.

I. 서론

1.1 모바일 애플리케이션 관련 보안 이슈

이제 스마트 폰을 활용하여 모든 것을 할 수 있게 되어, 별다른 지식 없이도 단순히 화면만 터치하면 모든 것을 할 수 있는 세상이다. 누구나 활용 가능한 편리한 조작성, 휴대가 용이한 이동성 등의 이점을 무기로 이제 스마트 폰은 우리 생활에 필수 불가결한 요소가 되었다. 이러한 특징으로 인해 PC 화면에서는 고려되지 않던 개인위치 정보와 같은 사생활과 밀접한 보안위협들이 모바일 환경의 특수성으로 이슈가 되고 있다[1].

기존 안드로이드 기반 애플리케이션 관련 취약점은 표1과 같다[2]. 이와 같은 보안 위협은 데이터 유출, 인증 회피, 네트워크 공격, 데이터 무결성 파괴 등이 있다.

점검 항목	요약
무결성 확인	클라이언트 프로그램 조작 여부를 확인하는지 점검
코드 난독화	프로그램 코드에 대한 난독화 여부 점검
데이터 난독화	주요 데이터에 대한 난독화 여부 점검
네트워크 통신 암호화	통신에 주고받는 데이터 암호화 여부 점검
필요 없는 정보	디버그 정보 등 실행에 필요 없는 추가 정보가 존재하는지 점검

[표1] 안드로이드 기반 앱 관련 취약점

안드로이드 OS는 모든 소스가 공개되어 있어 위·변조가 가능해 해킹 위협이 크다. 악의적인 사용자는 이점을 이용해 쉽게 공격할 수 있는 포인트를 얻어낼 수 있다. 예를 들어, 악의적인 사용자는 권한상승과 같은 악성코드로 개인정보를 탈취하거나 정상적으로 동작하는 앱을 악성

앱으로 리패키징하는 방법을 통해 정보를 유출하고, 금전적인 피해를 입힌다.

1.2 O2O 서비스 기반 앱 관련 보안 이슈

O2O는 온라인과 오프라인 시장의 유기적 융합이다. 이것은 스마트 폰의 높은 사용과 함께 M2O라는 새로운 시장을 창출했고, 이미 우리 생활 깊숙이 자리 잡고 있다. M2O 서비스를 이용하는 사용자는 서비스의 판매처를 직접 찾을 필요가 없고, 판매자는 더 많은 사용자를 쉽게 찾을 수 있다[3]. O2O 서비스 오프라인 '중개' 서비스를 대체하는 경향이 있지만 중개의 대상이 되는 오프라인 서비스 '자체'를 대체하는 것이 아닌 오프라인 중개서비스가 디지털·온라인화된 것이다. 개인의 모바일 기기 사용이 크게 증가하면서 숙박, 배달, 택시, 쇼핑 등 실생활과 밀접한 관련이 있는 서비스에 사용되기 시작하였다, 이후 실생활과 밀접한 관련이 있는 서비스들을 모바일을 통해 쉽게 접근할 수 있다 보니 민감한 개인정보와 같은 보안위협들이 모바일 환경의 특수성으로 이슈가 되었다. 같은 보안취약점이라도 일반 앱보다 개인정보를 많이 다루고 있는 앱에 더 위협적으로 보인다. O2O 서비스는 유형에 따라 사생활 등 민감한 정보를 다루는 경우가 많아 침해사고로 해당 정보가 유출될 경우 이용자에게 2차 피해를 유발할 가능성 또한 존재한다.

현재는 17년도에 발생한 개인정보 유출사건으로 집단소송까지 이어지는 심각성이 증가하고 있는 상태이다[4]. 본 논문에서는 안드로이드 환경에서 대표적으로 사용되고 있는 O2O 서비스 기반 애플리케이션 중 업체를 선택하여 실제로 어떤 취약점이 존재하는지를 분석한 사례를 소개하려고 한다. 온라인에서의 동작이 오프라인에서 어떤 악영향이 끼치는지 주목하였다.

II. 관련연구 및 사례

O2O 서비스 기반 애플리케이션은 단기간에 급격하게 성장을 하게 되어 보안을 고려하지 않고 개발이 이루어지는 경우가 존재하였다. 최근 개인정보 등 중요정보의 임의 유출 사건으로 인한 피해가 증가하면서 모바일 애플리케이션에 대한 보안성 검증 필요성이 증가하였다. 17년도에 숙박 서비스를 제공하는 애플리케이션에서 이용자 수의 절반인 개인정보가 유출된 사건이 발생하였다. 해당 사건의 공격원인으로는 홈페이지 보안취약점인 'SQL injection' 으로 밝혀졌다. '2016년 4분기 웹 공격 분석 보고서'에 따르면 'SQL injection' 공격은 두 번째로 자주 이용되는 해킹 수법으로 조사되었다[5].

[그림1] 취약점 유형별 웹 공격

또한, 사용자 권한 정보를 가로채는 세션 하이재킹 수법도 쓰였는데, 이를 탐지·차단하는 체계가 없었다. 이러한 해킹 수법이 보안 점검과 업데이트 등을 통해 대부분 막을 수 있는 것이었다는 지적을 받았다. 위의 사고를 바탕으로 미래부는 관련 기업의 유사 피해 발생을 예방하기 위해, 개인 및 결제 정보 등 민감 정보를 다룰 것으로 예상되는 배달, 숙박, 쇼핑 등 70여개 O2O 서비스를 중심으로 보안실태점검을 실시했다. 점검의 결과로 67개의 취약점을 식별하였으나, 지속적으로 증가하는 시장규모에 비해 아직 명확한 취약점 대응 방안이 존재하지 않고 있는 상황이다[6]. 하지만, 이렇게 유효한 취약점이 도출되었음에도, 지속적으로 증가하는 시장 규모에도 불구하고 아직 명확한 취약점 대응방안이 존재하지 않는 상황이다.

III. 취약점 분석 사례

O2O 서비스를 제공하는 애플리케이션은 다른 애플리케이션과 마찬가지로 이를 설치하기 위한 APK를 제공한다. 이 APK는 공격자의 역공학(Reverse Engineering)을 통한 취약점 분석에 노출될 수 있다.

3.1 취약점 상세 내용

다음은 O2O 서비스를 기반으로 하는 애플리케이션을 역공학을 이용하여 취약점을 찾아낸 상세 내용이다. 로그인을 인증하는 과정에서 발견한 취약점과 패킷으로 전달되는 인증번호를 탈취하여 인증우회가 가능한 취약점을 발견하였다.

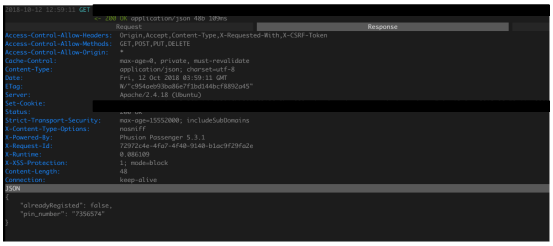
3.1.1 로그인 인증 취약점

A업체의 서비스를 분석한 결과 XML injection을 통하여 임의로 타 사용자의 계정에 접근이 가능함을 발견하였다. 사용자에 대한 식별을 database의 회원 인덱싱 하나만을 unique id(private key)로 사용한다는 점을 이용하여 타 사용자의 계정으로 접근 가능하며, 계정 정보의 변경 또한 가능하다. brute forcing 공격 기법을 이용하거나 또는, 앱을 실행한 후 해당 바코드의 마지막 번호를 통해서 임의로 타겟인 사용자의 id 값을 획득할 수 있었다. 해당 취약점은 사용자의 계정에 임의로 접근 가능하며, 계정 정보의 변경 또한 가능하다. 계정 정보 변경을 통해서 사용자의 이메일을 임의로 할 수 있기에 특정 이메일로 변경을 하였다. 지정한 특정 이메일로 인증번호를 받고, 임시 비밀번호를 발급받음으로서 계정 탈취가 가능했다.

즉, 사용자의 id 값만 알 수 있다면 해당 사용자에 대한 정보를 열람할 수 있을 뿐만 아니라, 패스워드, 이메일 주소 변경, 그리고 계정 탈퇴로까지 악용이 가능한 것으로 보여진다.

3.1.2 인증우회를 통한 가상화폐 적립

포인트 서비스는 해당 서비스 가입자의 고유권한이다. B업체에 경우 회원 가입 시 추천인을 입력하게 되면 추천한 사람은 포인트를 적립할 수 있다. 이때 자신의 아이디에 고의적으로 아이디를 생성하여 포인트를 적립하는 행위를 막기 위해서 추천인 코드 입력 시 핸드폰 번호 인증이 필요하다. 여기서 발견된 취약점은 실제로 사용하는 핸드폰 번호가 아니더라도, 해당 패킷만 확인하면 인증번호를 알 수 있다.



[그림2] 패킷으로 전달되는 인증번호

정상적인 인증 절차는 SMS를 이용하여 인증번호를 전달받고 인증번호를 입력하는 방식인데 인증절차를 서버에서 하지 않고 클라이언트가 하기 때문에 패킷을 통해서 사용자에게 정상적인 인증값을 보내주고 기기 내부에 저장한다. 이 과정에서 패킷을 통해서 인증번호를 확인 할 수 있기 때문에 핸드폰이 없어도 인증을 진행 할 수 있다.

3.2 악용되는 사례

발견된 취약점은 실제 침해사고에 악용될 사례가 존재한다. 개인정보와 관련된 침해사고가 발생할 경우 심각한 범죄로 이어질 가능성도 존재한다.

3.2.1 사용자 계정 정보 유출

사용자의 정보가 조회된다는 사실은 정보유출사건으로 이어진다. 해당 정보를 이용해서 공개된 사이트에 사용자의 정보를 피드 형식으로 계속해서 업로드할 수 있다. 자신의 정보가 공개된 환경에 노출될 수 있는 위험성이 존재한다. 또한, 유출한 정보를 이용하여 명의도용 및 개인정보 불법거래 등 각종 범죄로 이어질 수 있다.

3.2.2 바코드 결제

바코드가 존재하는 앱은 바코드를 찍는 형태로 결제를 수행하여 추가적인 사용자 검증 절차가

없기 때문에 결제가 자유롭게 가능하다. 악의적인 사용자가 임의로 조작하여 다른 사용자의 바코드를 이용해서 재화를 불법적으로 무단 사용할 수 있을 가능성이 보인다.

3.2.3 쿠폰 사용

쿠폰 서비스가 가능한 A업체 앱의 경우 API를 통해서 조회를 할 시에 id 값을 토대로 조회를 하게 된다. 그 과정에서 현재 로그인된 사용자에 대한 검증이 없어서 다른 사용자의 쿠폰에 대한 정보를 가져와서 임의로 사용할 수 있다. 덧붙여 brute-forcing을 통해 쿠폰의 개수를 조작 또한 가능하다.

IV. 결론

국내 스마트 폰에서 이용되고 있는 애플리케이션 중에서 O2O 서비스 기반 앱은 상당한 비율을 차지하고 있지만, 해당 앱이 가진 특수성으로 정보유출의 위험이 다량 존재함을 확인하였다. 안드로이드 기반 앱의 특성상 쉽게 변조 및 접근이 가능하여 보안 위험성 또한 매우 높아 보안 대응수단을 적용하기 어려운 점이 존재하지만, 최소한 민감한 개인신상정보에 접근하는 것을 막는 대응이 필요하다.

예상되는 대응방안으로는 사용자 인증절차가 초기 로그인시 1회성으로만 존재하는 것이 아니라, 로그인 이후에도 추가적인 사용자 검증절차를 통해 해당 사용자임을 확인하는 대응 방안에 대한 기술적인 연구가 필요하다.

또한, 미래부에서 진행한 보안실태점검과 같이 보안 대책을 마련하기 위해 기업들에게 가이드라인을 제시한다고 해도 관련 업체들의 보안의식이 중요할 것으로 생각된다. 동시에 서비스를 제공하는 기업 스스로 취약점 개선을 위한 노력이 필요하다.

[참고문헌]

[1] 방지호, 하란, 강필용, 김홍근 (2012). 전자정부 모바일 앱 보안성 검증체계. 한국통신학회 논문지, 27(2), 119-131.

[2] 김휘강, 금영준 (2013). 안드로이드 환경에서의 모바일 게임 서비스 보안이슈. 정보보호학회지, 23(2), 35-42.

[3] 김동수, 김관호, 최동욱, 정재윤 (2016). 국내 O2O 서비스 이슈 진단 및 산업활성화 정책 방향 제안. 한국전자거래학회지, 21(4), 137-150.

[4] 김재섭, (2017.05.18.). '여기어때' 개인정보 유출 피해자, 집단 손해배상 소송 잇따라. 한겨레

[5] 씨디네트웍스. 2016년 4분기 웹 공격 분석 보고서

[6] 미래창조과학부(2017). O2O 서비스 보안 실태 점검 실시. 보고자료

소프트웨어 개발 생명주기에서 위협 인텔리전스 적용을 통한 소프트웨어 보안강화 방안

김예준*, 류기백*, 이유정*, 염흥열*
*순천향대학교 정보보호학과

Software Hardening by Applying Threat Intelligence in Software Development Life Cycle

Ye-Jun Kim*, Ki-Baek Ryu*, Yu-Jeong Lee*, Heung-Youl Youm*
*Department of Information Security, Soonchunhyang University.

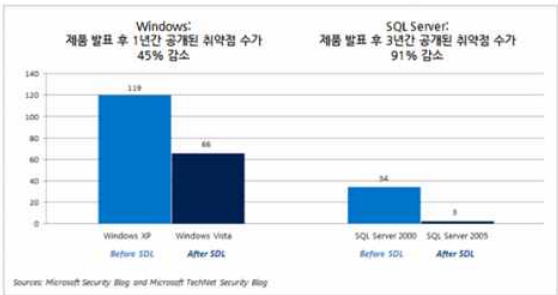
요 약

기존 국내에서 프로그램을 개발할 때, 시큐어 코딩을 통하여 코드 작성 시 취약점이 발생하지 않도록 노력하고 있다. 해당 과정은 SDLC(Software Development Life Cycle, 소프트웨어 개발 생명주기)의 개발 단계에 속하며, 소프트웨어 개발 단계의 일부에 불가하다. SDL에서는 크게 정의, 개발, 유지보수 단계가 존재하고 있지만 대부분 개발 단계 다음부터 중점적으로 검토가 진행되며 소프트웨어가 만들어지고 있다. 또한, 소프트웨어의 특성마다 각 단계의 중요성이 다르다. 따라서 정의단계에서부터의 부분적 검토 강화를 진행함으로써 소프트웨어 개발에서부터의 보안강화가 가능하다. 본 논문에서는 SDLC의 정의단계인 Requirements Analysis와 디자인 단계에서 Threat Intelligence(위협 인텔리전스)의 적용을 통해 보안 위협을 최소화하는 방안에 관한 연구를 진행하고자 한다.

I. 서론

1.1 연구배경 및 목적

NIST(National Institute of Standards and Technology, 미국 국립표준기술연구소)의 문서에서는 소프트웨어의 결함으로 발생한 사고를 처리하기 위한 유지보수 비용이 미국 GDP(Gross Domestic Product)의 0.6%인 약 590억 달러라고 한다. 또한, 소프트웨어의 취약점을 제거하는 비용은 소프트웨어의 설계 단계보다 최대 30배 더 많이 요구된다고 한다.[1] 개발 전, 설계 단계에서부터 꼼꼼히 살피고 보안 위협을 최소화하는 과정을 거치게 된다면 이후 요구되는 금액을 상당히 절약할 수 있다. [그림 1]과 같이 마이크로소프트에서 2004년 1월 출시한 MS-SDL(Microsoft-Security Development Lifecycle)을 Windows에 적용한 후, 1년간 공개된 취약점의 수가 45% 감소하며, SQL Server의 경우에는 제품 발표 후 공개된 취약점의 수가 91% 감소했다.[2]



[그림 1] SDL 출시 후 발견된 취약점 감소

SDLC는 일반적으로 요구사항분석, 설계, 구현, 테스트, 유지보수 등의 단계로 이루어져 있다. 현재 대다수가 구현 단계의 Secure Coding부터 테스트 단계의 Penetration Test (모의 해킹, 침투 테스트)에 초점을 두고 있

으며 요구사항 분석 단계와 설계 단계의 디자인적인 부분은 그에 비하여 소홀히 넘어가는 경향이 있다. 하지만 개발 전, 디자인적인 부분도 SDLC의 하나로써 중요한 부분이며, 이를 더 적극적으로 이용하게 된다면 사고 이후 발생하는 비용이 절감되면서, 국내 소프트웨어 보안 시장에 도움이 될 것으로 보인다.

2장에서는 SDLC, Threat Intelligence에 대하여 서술할 것이며, SDLC 과정에 적용 가능하다고 판단되는 Threat Intelligence의 지표에 대하여 제안하며 마지막 3장에서는 결론으로 이에 대한 연구의 필요성과 연구 결과로써의 기대효과에 대하여 기술하고 마치고자 한다.

II. 본론

2.1 SDLC

SDLC는 해킹 등 사이버 공격의 원인인 보안 취약점을 개발 단계에서 사전에 제거하고 SDLC의 단계별로 수행하는 일련의 보안 활동을 통하여 안전한 소프트웨어 개발 및 운영하기 위한 목적으로 적용하는 개발 체계로써 [그림 2]와 같은 구조를 이루고 있다.[3]

[그림 2] SDLC 과정 [3]

2.1.1 요구사항 분석 단계

소프트웨어의 기능, 성능, 신뢰도 등에 대한 요구사항뿐만 아니라 보안에 관련된 요구사항 및 보안정책들을 요구사항 명세서(SRS, Software Requirement Specification)를 작성하여 이후에 진행될 설계, 구현, 테스트 단계에서 필요한 보안 활동을 빠뜨리지 않도록 명시하는 단계이다.

2.1.2 설계 단계

설계 단계에서는 시스템을 분석해 위협들을 도출해내는 위협 모델링 과정을 통해 도출된 여러 위협과 그에 따른 보안대책은 이후에 구현, 테스트, 유지보수 단계에서의 보안 활동 기초를 설계하는 단계이다.

2.1.3 구현 단계

소프트웨어를 프로그래밍하는 단계로 보안 코딩 및 디버깅이 수행되며 그 결과를 검증하는 단위테스트를 수행한다. 단위테스트를 통해 소프트웨어에 내재되어있는 보안 취약점을 제거하며, 보안 요구사항과 위협 모델링을 통해 수립된 보안대책 등이 제대로 구현되었는지 소스 코드를 점검하는 보안 코드 리뷰가 수행되는 단계이다.

2.1.4 테스트 단계

개발된 모듈들에 대한 통합 테스트, 요구사항을 제대로 구현했는지 확인하여 시스템 테스트 등이 행해지는 단계로 위협 모델링을 통해 도출된 위협들이 구현 단계에서 충분히 제거되었는지를 동적 분석 및 모의침투 테스트를 통해 검증하는 단계이다.

2.1.5 유지보수 단계

개발된 소프트웨어를 직접 이용해보고 이용시 발생할 수 있는 문제점 및 보안사고에 대하여 관리, 사고대응, 패치가 병행되는 단계이다. 개발 단계에서 미처 처리하지 못한 취약점이나 새롭게 발견된 취약점이 있을 수 있으므로 특히 강조되는 단계이다.

2.2 Threat Intelligence

Threat Intelligence는 오늘날 복잡한 조직의 사이버 보안 정보를 집중적으로 사용하고 우선순위를 정하는 데 도움이 될 수 있다.[4] 이와 같은 Threat Intelligence는 과거에 발생하였던 침해사고에 대한 조사 및 분석을 통하여 얻을 수 있는 정보를 조합하고, 새로운 정보를 만들어 미래의 사이버 위협에 대응할 수 있는 등 Threat Intelligence에서 얻는 정

보와 데이터의 활용성은 무궁무진하다.

이와 같은 Threat Intelligence에서도 소프트웨어 개발과 같이 생명주기가 존재하며 이는 [표 1]과 같다.

[표 1] Threat Intelligence의 생명주기 [5]

Threat Intelligence의 생명주기
1. Threat landscape changes
조직에 노출되는 위협 환경은 동기, 의도, 기능 및 운영의 변화에 따라서 달라질 수 있다. 이는 조직적으로 Threat Profile에 영향을 줄 수 있으며, TTP(Tactics, Techniques, Procedures)에도 문제를 이를 킬 수 있다. 이는 얼마나 변화는 지에 따라 Intelligence의 기능에 영향을 미칠 수 있다.
2. Organization changes
조직은 사람과 기술의 전환 등 유동적인 상태에 있으며 그에 따라서 가지고 있는 기술들이 쓸모가 없어질 수 있다. 또한, 침해사고가 발생한 상황에서 필요한 지식과 대처능력은 오랫동안 유지되지 않을 수 있다.

결과적으로는 특정 조직에서 Threat Intelligence 기능을 유지하고 발전시키기 위해서는 높은 수준의 라이프 사이클이 필요하다. [5]

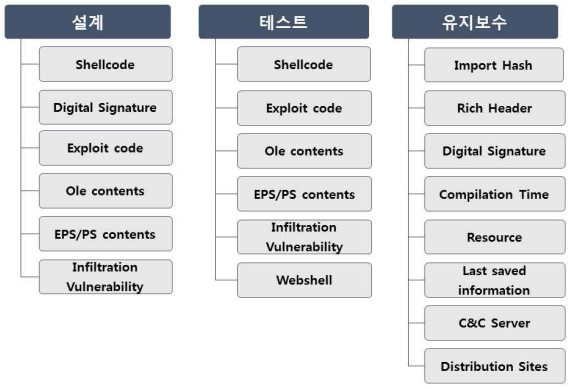
2.3 SDLC에 Threat Intelligence의 적용

SDLC의 설계 단계 중 위협원 도출을 위한 위협 모델링 부분이 존재한다. 기존에 등록되어있는 1-day 취약점을 방지하는 것처럼 위협 모델링 과정 중에서 Threat Intelligence에서 얻을 수 있는 정보나 데이터를 활용하여 과거의 침해사고 사례부터 앞으로의 공격 가능 시나리오 등을 검토하고, 적용한다면 보안성이 높은 소프트웨어가 개발될 것으로 기대하고 있다. 아래의 [표 2]는 Threat Intelligence를 이용하여 식별할 수 있는 침해지표의 일부이다.

[표 2] Threat Intelligence 단계의 침해지표[6]

Attribution에 필요한 침해지표	
Portable Executable	- Algorithm - PDB information - Section information - Debug Message - Import Hash - Mutex - Registry Key / Value - Created Process name - Rich Header - Filename - Digital Signature - Shellcode - Export Function Name (DLL) - Compilation Time - Resource language code - Zip library version - SSDeep (fuzzy hash) - Registered service name - Created Filename - Library loading method
Document	- Font - Author - Exploit code - OLE contents - Last Saved Information - Shellcode - EPS/PS contents
APK	- Classname - APK Package Structure - Package name - Image Resource - Digital Signature
Common	- C&C Server - Infiltration Vulnerability - Distribution Sites - Linguistic Analysis - Email Information - Webshell - Downloaded filename

위 나열된 지표들은 단지 Threat Intelligence에서 Attribution을 하기 위하여 사용되는 침해지표의 일부이다. 하지만 [표 2]를 이용하여 공격 그룹과 특성, 자주 사용되는 공격 방식과 취약점 등에 대한 정보를 추가로 얻고 Attribution을 하게 된다면 이에 대한 정보도 SDLC에 적용할 수 있다. [그림 3]에서는 SDLC의 각 단계에서 적용이 가능할 것으로 기대되는 침해지표이다.



[그림 3] SDLC에 적용 가능한 침해 지표

[그림 3]과 같이 단계별 지표를 이용하여 위협 모델링, 보안 설계 검토, 모의침투 테스트, 보안 패치 등이 이루어질 수 있을 것으로 기대된다. 또한, 이와 같은 지표들을 종합하여 얻을 수 있는 기대효과는 [그림 4]와 같다.

[그림 4] 단계별 적용 가능한 추가 정보

이처럼 소프트웨어에 대한 위협 모델링과 보안 통제를 수립할 때 과거 침해사고에 쓰인 지표들에 대한 대비는 되었는지, 같은 사고를 반복하지 않을 수 있는지에 대한 검토가 필요하다. 또한, 설계 단계뿐만이 아닌 유지보수 단계에서도 추가로 특정 침해사고가 발생하였을 때는 다시 한번 Threat Intelligence를 이용하여 얻는 지표들에 대한 검토와 개선이 필요할 것이다.

III. 결론

Threat Intelligence를 SDLC에 적용함으로써 보다 세밀한 검토와 설계가 필요하며, 차후 발생 가능한 사고에 대해 즉각적인 조치 또한, 필요하다. 이에 따라 기존의 라이프 사이클에서 Threat Intelligence의 지표를 적용하게 된다면 차후 발생 가능한 취약점 및 침해사고에 대한 피해와 파급효과가 줄어들 것으로 기대하고 있다. 하지만 기존의 라이프 사이클에 새로운 지표를 추가하여 적용하게 되는 것은 많은 이들의 더 정밀한 연구가 필요하다.

[참고문헌]

[1] Gregory Tassey, "The Economic Impact of Inadequate Infrastructure for Software Testing Planning Report," NIST, May 2002, pp. 169-170

[2] Park Jinseok, Kang Heesoo, Kim Seungjoo, "How to Combine Secure Software Development Lifecycle into Common Criteria", Korea Institutes of Information Security and Cryptology, 2014, pp. 171-182.

[3] 행정자치부, 한국인터넷진흥원, "전자정부 SW 개발·운영자를 위한 소프트웨어 개발 보안 가이드", 2017.1

[4] Barnum, Sean. "Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX)." MITRE Corporation 11 (2012): 1-22.

[5] Jeff Berg, "The Role of Cybe Threat Intelligence in Security Operations", 2016 [Internet], https://www.fireeye.com/blog/products-and-services/2016/11/the_role_of_cyberth.html.

[6] Kyoung-Ju Kwak, 2018 [Internet], https://brunch.co.kr/@brunt/15

DLMS/COSEM 프로토콜을 사용하는 스마트 미터의 취약점에 관한 연구

박준영, 한창현, 피대권, 송상현, 조수민
한국정보기술연구원

Study on Vulnerability of Smart meter Using DLMS / COSEM Protocol

Jun-Yeong Park, Chang-Hyeon Han, Dae-Kwon Pi,
Sang-Hyeon Song, Su-Min Cho
Korea Information Technology Research Institute

요 약

정부는 2020년까지 대한민국 2,200만 가구에 AMI(Advanced Metering Infrastructure, 원격검침 인프라)의 보급을 추진 중이다. 전자식 전력량계는 AMI를 구성하는 요소 중 사용자와 맞닿아 있는 장비로서, 공격자가 가장 쉽게 접근할 수 있어 보안에 취약하다. DLMS/COSEM 프로토콜은 스마트 에너지 미터링, 제어 및 관리를 위한 세계 표준으로 전자식 전력량계의 핵심 프로토콜이다. 공격자는 DLMS/COSEM 프로토콜을 변조하여 원격검침에 장애를 일으키거나, 수용가의 전력을 차단할 수 있다. 본 연구는 AMI 시장 현황과 전력량계에 대한 공격 방법과 시나리오 그리고 문제점에 대한 대책을 제시한다.

I.서론

정부는 2020년까지 대한민국 2200만 가구에 AMI(Advanced Metering Infrastructure, 원격검침 인프라) 보급을 추진 중이며, 2018년 현재 약 1400만 가구에 보급되었다.[1] AMI는 대한민국 스마트 그리드 사업의 핵심 인프라로서, 전자식 전력량계와 DCU(Data Concentration Unit, 데이터 집중장치), 한전 FEP(Front End Processor)로 구성되어 있다. 전자식 전력량계에서 측정한 사용자의 전력 데이터를 DCU에게 PLC(Power Line Communication, 전력선 통신)를 이용해 전송하고, DCU는 망 통신을 이용해 수집된 데이터를 한전 FEP 서버로 전송한다.

대한민국뿐만 아니라 전 세계 AMI 시장도 지속적으로 성장 중이다. 전자식 전력량계 시장은 2017년 127억 달러 규모의 시장을 이루고 있다. 일본은 세계 최대 전자식 전력량계 시장으로 2015년 기준으로 약 20억 달러 규모의 시장을 형성하고 있으며, 중국의 경우 약 3억 가구 이상에 전자식 전력량계를 보급한 상태다. 유럽지역은 2015년 10개 국가에서 100% 전자식 전력량계 설치를 완료한 상태이며, 미주지역 또한 7천만대 이상 설치되어 있다.[2]

전자식 전력량계와 PLC 모뎀은 공격자가 가장 쉽게 접근할 수 있는 장비이며, 양방향 원격검침을 지원하는 국제표준 프로토콜인 DLMS/COSEM(Device Language Message Specification/Companion Specification for Energy Metering, 장치 간 통신 명세/전력량계를 위한 보조 명세)을 사용하고 있다. 공격자는 전자식 전력량계에 접근하여 DLMS/COSEM 프로토콜 패킷을 조작해 전력량계의 데이터를 변조하거나 한전 운영프로그램과의 통신을 마비시킬 수 있다.

이전까지 전자식 전력량계에 대한 보안 위협과 관련된 많은 논문이 작성되었다. 하지만

DLMS/COSEM 프로토콜의 직접적인 취약점에 관한 연구는 많이 진행되지 않았으며, 대부분 실제 원격 검침에 사용되는 장비를 이용하지 않는 문제가 있었다. 본 연구는 실제 대한민국 AMI 장비를 기반으로 연구하였다. 연구를 통해 앞으로 발생할 전력 그리드의 보안 위협을 사전에 방지하고, 해결방안을 제시하는데 도움이 될 것으로 기대된다.

II.본론

2.1 AMI 구조

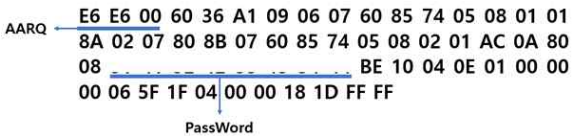
국내 보급중인 전력량계는 크게 표준형 전력량계, E-Type 전력량계, G-Type 전력량계, AE-Type 전력량계로 나뉜다. G-Type과 AE-Type은 원격에서 계절별/시간대별 요금적용을 위한 프로그램이 가능하고 설정된 요금 프로그램 때문에 유효/무효/피상 전력량, 최대 수요전력, 전압, 전류, 주파수, 역률 등을 측정하는 기능을 갖고 있다.[3] G-Type과 EA-Type은 DCU와 통신하기 위해 전력량계 내부에 PLC 모뎀을 내장하고 있다. E-Type과 표준형 전력량계는 외부에 설치된 외장형 PLC모뎀과 연결을 통해 DCU와 통신한다. DCU는 WCDMA(Wideband Code Division Multiple Access, 광대역 코드 분할 다중 접속)등 망 통신을 이용해 수집한 검침 및 모뎀 정보를 서버로 전송한다.[4]

2.2 DLMS/COSEM 프로토콜

DLMS/COSEM 프로토콜은 스마트 에너지 미터링, 제어 및 관리를 위한 세계 표준으로서 객체지향 데이터 모델, 응용 계층 프로토콜 및 매체별 통신 프로 파일을 지정한다. DLMS/COSEM 프로토콜은 다음 3가지 요소로 구성된다. 첫 번째 요소인 DLMS는 개체가 보유한 정보를 메시지로 변환하는 응용 프로그램 계층 프로토콜이다. 두 번째 요소인 COSEM은 에너지 미터링을 위한 인터페이스 모델이며, 3단계(연결-데이터통신-연결해제)로 이루어진다. 세 번째 요소인 OBIS(Object Identification System) 코드는 DLMS/COSEM 프로토콜이 계량기를 구성하는 모든 정보를 각각 객체로 모델링 하는데, 그 모델링된 객체를 매핑하는 코드다. G-Type 전력량계의 경우 약 210 여개의 OBIS 코드를 갖고 있다.[4].

DLMS/COSEM 프로토콜은 Client-Server의 요청-응답 방식으로 통신이 이루어지는데, 그 과정은 다음과 같다. SNRM - UA 데이터 교환을 통해 TX, RX 데이터 사이즈 등의 정보를 교환한다. 그리고 AARQ - AARE 데이터 교환을 통해 사용할 메서드, Max PDU Size 등의 협상 과정이 이루어지게 되는데, 이 때 AARQ 프레임에는 계량기에 연결하기 위한 인증데이터(PASSWORD)가 포함되어 있다. 정상적으로 세션을 맺었다면, OBIS 코드와 Get, Set 메서드 등을 이용하여 데이터를 읽어오거나 저장된 데이터를 변경할 수 있다. 데이터 송, 수신이 완료되면 Disconnect Request를 보내 연결된 세션을 끊는다.

세션 요청에 사용되는 AARQ 프레임 구조는 HDLC 프레임을 기반으로 하고 있다. 계량기 세션 생성 과정에서 인증이 필요한 경우 AARQ 프레임에 PASSWORD 데이터를 포함하여 전송하게 되는데, 그 구조는 다음과 같다. PASSWORD는 8Byte의 ASCII Code로 이루어져 있으며 PASSWORD 오류 발생 시 Server전력량계)측에서 인증 에러를 반환한다.



<그림 1> AARQ 패킷

2.3 공격 시나리오

AARQ-AARE 데이터 교환에 성공하면 OBIS 코드를 포함한 Get, Set 등의 메서드로 계량기 내부 데이터를 요청, 변경할 수 있다. Set Request 패킷 프레임의 Information

Data(I-frame)는 <그림 2>와 같다.

<그림 2> Set Request I-Frame

Set Request 패킷 프레임은 Set Request를 나타내는 Value 이후에 Class_Id, OBIS, Attribute_Id, Data가 순서대로 적용된다. 공격자는 <그림 3>처럼 Daisy chain 형식으로 최대 32대의 전자식 전력량계에 USB to RS485 포트를 이용하여 PC를 연결할 수 있다.

<그림 3> 공격자 접근 방식

일반적으로 전자식 전력량계의 PASSWORD 값은 기본 값으로 설정되어 있다. 공격자는 Daisy chain 형식으로 연결되어있는 모든 전력량계의 PASSWORD값을 바꾸는 AARQ Set Request 패킷 프레임을 생성하여 브로드캐스트로 전송 하면 연결된 전력량계의 모든 PASSWORD 값이 변하게 된다. 공격이 성공하면 전력 공사가 원격검침을 위해 운영프로그램을 사용하여 사용자의 전력 데이터를 요청해도 AARQ 세션 생성과정에서 에러가 발생하여 원격검침이 불가능하게 된다. 또한 전력량계의 통신 속도를 한전 운영프로그램과 불일치 되도록 변조하면 마찬가지로 원격검침이 불가능하게 된다.

<표 1>은 DLMS/COSEM 프로토콜을 이용해 쓰기가 가능한 OBIS 코드 목록이다. 이를 이용해 다양한 공격이 가능하다. 타임스위치의 개폐시간을 변조하거나, 전류 제한 설정값을 타깃의 평균 사용 값보다 낮게 설정하면, 설정값을 항상 초과하기 때문에 래치 릴레이 부하 차단으로 수용가의 전력 공급이 차단된다.

OBIS Hex	Description
0000600302FF	부가신호 장치 출력
0000603233FF	공급자 모드 표시
0000603234FF	무부하 시 부하동작 표시 설정
008063620DFF	전류제한 자동복귀 횟수 설정
010000F200FF	검침 파라미터
010000F201FF	선택유효전력량개별설정
010000F300FF	계량모드설정 파라미터
0100800008FF	경부하오결선감지체크
010080000AFF	Key설정
010080000DFF	선택무효전력량개별예약설정
018080BD03FF	잔여시간
01000B1F00FF	전류제한설정값
01000C1F00FF	Sag설정값
01000C2100FF	Sag설정시간
01000C2300FF	Swell설정값
01000C2500FF	Swell설정시간
010080000CFF	평균전압전류연산구간
0100802300FF	온도Threshod설정
0101000804FF	LoadProfile기록간격
018080BD01FF	잔여전력량[Q1+Q4]
018080BD02FF	잔여금액
018080BD04FF	구매전력량
018080BD05FF	구매금액
0101010400FF	현재/직전수요시한수요전력 (m:계측항목)
0101090400FF	현재/직전수요시한수요전력 (m:계측항목)
0000010000FF	일자/시간
00000B0000FF	정기,비정기휴일
0000280000FF	LN association
00000D0000FF	TOU
00000F0000FF	정기검침일
00000F0100FF	비정기검침일
0100800004FF	타임스위치개폐시작시간
0000160000FF	HDLC설정

<표 1> 쓰기 가능한 OBIS 코드

쓰기 가능한 값들 외에도 타깃의 전력사용과 관련된 값들을 조회하면 개인 프라이버시 문제가 생길 수 있다. 배전함에 소형 보드를 연결해

지속적으로 특정 타깃의 전력 사용 정보수집하고 분석해서 생활 패턴을 알아낼 수 있다.

2.4 예상 결과

PASSWORD 변경이나 전력량계의 통신속도 변경으로 원격검침을 불가능하게 만들 경우 한전 운영프로그램을 이용하여 자동으로 전력량을 검침하고 과금하던 전력 공사의 FEP에 문제가 발생할 것이다. 또한 타임스위치 개폐시간 변조나 전류제한 설정을 바꿔 래치 릴레이 부하 차단으로 공장과 같은 사업체의 전력을 차단하면 개인에게 큰 금전적인 손실을 입힐 수 있다. 또한 배전함에 연결되어있는 모든 전력량계의 수용가가 단체로 정전되는 상황이 일어날 수 있다. 전력 사용 정보를 수집하여 타깃의 생활 패턴을 알아내는 경우는 타깃의 외출 시간을 알아내어 범죄에 악용할 수 있을 것으로 예상된다.

2.5 대책

원격 검침 자체를 불가능하게 만드는 요소인 기본으로 설정된 PASSWORD 값을 변경하고, 주기적으로 업데이트해야 한다. 또한 AARQ 인증 레벨을 LOW에서 HIGH로 변경해야 한다. 인증 레벨을 HIGH로 변경하면 GETRequest를 통해 평문으로 전송되던 전력 정보가 노출되지 않아 보안성을 강화할 수 있다. 또한 한전 운영 프로그램과 통신 속도 불일치를 이용한 공격은 통신 속도를 정기적으로 동기화 시키는 방안이 필요하다. 타임스위치 개폐시간, 전류 제한 설정을 이용한 전력공급 차단 문제나 개인 프라이버시 침해 문제는 DLMS/COSEM 프로토콜 통신 과정에서 HCS(Header Check Sequence)와 FCS(Footer Check Sequence) 계산 알고리즘에 암호화를 적용하는 것을 고려할 수 있다. 예로 SHA256, ECC 등을 들 수 있다. 또한 물리적 접근 제어를 고려할 수 있다. 특히 노후화된 아파트나 농촌지역의 경우는 외부에 배전함이 노출되어 있어 효과적일 수 있다.

III. 결론

AMI는 현재 대한민국뿐만 아니라 전 세계적으로 전력 그리드 시장을 바꾸고 있는 핵심 기술이다. 양방향 통신을 이용해 쉽게 사용자의 전력 사용량을 점검하고 요금을 부과할 수 있지만, 많은 보안 위협이 내재되어 있다. 단기적으로 검침 정보의 변조나 전기 요금 변조 등의 이슈가 생길 수 있으며, 장기적으로 보면 국가 전력 시장의 신뢰도에 악영향을 끼칠 수 있다. 본 논문에서 언급한 바와 같이 전력량계와 AMI 구성 요소에 누구나 쉽게 접근하고 데이터를 변조시킬 수 있으므로 물리적, 기술적 보안 수준을 향상시킬 필요가 있다. 차후 추가적인 연구를 통해 PLC 모뎀과 DCU 간의 AES128 암호화 통신 취약점과 대책을 강구할 계획이다.

[참고문헌]

[1] 한국에너지신문, http://www.koenergy.co.kr/news/articleView.html?idxno=88327, 2017.03

[2] 에너지경제연구원, 세계 에너지시장 인사이트 제16-15호, 2016.4

[3] ㈜남전사, 지타입 전자식 전력량계 사용자 메뉴얼 버전 1.0, 2016.8

[4] 한국전력공사, 지중구간 PLC AMI 및 신형 전력량계 통신기술 개발, 2015.10

HastFuzz: 선행 분석 결과 기반의 실행 흐름 유도를 통한 퍼징 방법론

조승현†, 이광진†, 최광준, 진호용, 나윤종, 박기웅*
세종대학교 정보보호학과

HastFuzz : Precedence Result Analysis Based Control Flow Induction Fuzzing Scheme

Seung-Hyeon Cho†, Kwang-Jin Lee†, Kwang-Jun Choi, Ho-Yong Jin, Yoon-Jong Na, Ki-Woong Park*

Department of Computer and Information Security, Sejong University.

요 약

퍼징은 소프트웨어에 무작위 데이터를 입력하여 크래시 유발을 유도하고 이를 통해 취약점을 찾기 위한 소프트웨어 테스트 기법으로 활발하게 사용되고 있다. 그러나 기존의 퍼징 방법론에서는 코드분석과 퍼징의 과정을 분리하고 있기 때문에 이미 발견된 취약점이나 소스 코드 분석 등 선행 분석 결과를 통해 도출된 정보를 퍼징에서 활용할 수 없다는 한계가 있었다. 본 논문에서는 선행 분석 결과를 통해 도출된 정보를 퍼징에 활용할 수 있는 HastFuzz를 제안한다. HastFuzz는 공개된 자바스크립트 소스 코드의 AST(Abstract Syntax Tree) 정보에서 AST 노드 간의 상하 관계를 사전에 학습하여 해석기의 문법 검사를 통과하는 AST를 생성한다. 이렇게 생성된 AST는 휴리스틱 기법을 활용한 부분 재생성 과정을 통해 코드 커버리지를 극대화한다. 이러한 접근 방법은 선행 분석 결과를 통해 도출한 정보를 퍼징에서도 활용할 수 있도록 하며 코드 커버리지 확대 과정에 방향성을 유도할 수 있다. 이를 통해 경계값 조건에 더욱 쉽게 접근할 수 있도록 퍼징을 유도할 수 있으며, 효율적인 퍼징을 가능하게 한다.

I. 서론

퍼징은 소프트웨어 테스트 기법 중 하나로서, 프로그램에 유효하지 않거나 예상치 못한 입력을 생성하고 실행하는 방식으로 버그를 찾아내는 기법이다. 이러한 소프트웨어 버그는 치명적인 취약점으로 이어질 수 있으므로 이를 미리 찾아내어 사전에 차단하는 것은 매우 중요하다. 따라서 많은 보안 연구자들은 다양한 퍼징 기법[1, 2, 3, 4, 5, 6, 7]을 활용하여 소프트웨어 버그를 더욱 효율적으로 찾아내고자 하였다.

웹 브라우저의 경우 자바스크립트를 해석하고 실행하는 자바스크립트 엔진에서 많은 버그가 발생하였기 때문에 자바스크립트 엔진을 대상으로 한 퍼징에 대해 다양한 연구[1, 3, 6, 7]가 선행되었다. 그러나 선행된 연구에서 제시된 퍼징 기법들은 코드 분석과 퍼징을 분리하고 있으므로 이미 발견된 취약점이나 소스 코드 분석 등 경험으로 도출한 정보를 퍼징에서 활용할 수 없다. 더불어 퍼징을 수행하면서 경계값 도달 여부를 확인할 수 없기 때문에 경계값에 대해 충분한 퍼징을 수행할 수 없는 한계점이 있다.

본 논문에서 제시하는 퍼징 방법론은 AST를 확장하여 스크립트 해석기의 문법 검사를 통과할 수 있는 자바스크립트 소스 코드를 생성한

본 연구는 한국연구재단 지원사업(2017R1C1B2003957) 및 2018년도 과학기술정보통신부의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00420, API 호출 단위 자원 할당 및 사용량 계량이 가능한 서버리스 클라우드 컴퓨팅 기술 개발)
† 주저자: 세종대학교 정보보호학과 학부과정 (me@shc.me, y2sm2n@gmail.com)
* 교신저자: 세종대학교 정보보호학과 교수 (woongbak@sejong.ac.kr)

다. 또한, 이를 실행하는 자바스크립트 엔진의 코드 커버리지를 극대화하여, 더 높은 확률로 취약점에 도달하는 것을 목표로 한다. 이를 위해 코드 커버리지 극대화 과정에 휴리스틱 기법을 활용하여 경험에 의해 도출한 정보를 퍼징에서도 활용할 수 있도록 하는 방법과 이러한 유도로 인해 얻게 되는 효과에 대해 설명하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 선행되었던 자바스크립트 퍼저들의 특징과 한계 및 코드 커버리지에 대해 살펴본다. 3장에서는 기존 퍼저들의 한계점을 개선한 새로운 퍼징 방법론에 대해 제시하고, 마지막으로 4장에서 결론과 향후 연구 방향에 관하여 서술한다.

II. 관련 연구

본 장에서는 자바스크립트 엔진을 대상으로 퍼징을 수행하며 해석기의 문법 검사를 효과적으로 통과하는 방법이 제안된 연구 중 대표적인 두 퍼저에 대해 소개하고, 코드 커버리지를 넓히기 위해 수행되었던 연구를 소개한다.

2.1 기존의 자바스크립트 퍼저

Jsfunfuzz[7]는 대표적인 생성 방식의 퍼저로서, Firefox 브라우저의 자바스크립트 엔진을 테스트하기 위해 개발되었다. Jsfunfuzz는 자바스크립트 언어의 문법 규칙을 내장하고 있으며, 이를 바탕으로 해석기의 문법 검사를 통과하는 올바른 구조의 자바스크립트 소스 코드를 생성한다. 이를 통해 해석기를 통과할 수 있지만, 직접 정의한 문법 관계를 바탕으로 코드를 생성하기 때문에 이를 일반화하여 자바스크립트를 제외한 다른 언어에 적용하기에는 다소 어려움이 있다는 한계가 있다. 또한, JSfunfuzz는 생성된 자바스크립트 소스 코드의 실행 결과를 이후 진행되는 퍼징에 활용할 수 없으며, 무작위로 생성하기 때문에 연속된 조건이 만족해야해야 도달 가능한 루틴에 대해서는 매우 낮은 확률에 의존할 수밖에 없게 된다. 이처럼 JSfunfuzz는 경계값 문제를 해결하지 못하고 있다.

이러한 JSfunfuzz의 일반화 문제와 경계값 문제를 개선하기 위해 Langfuzz[6]는 사용자의 입력을 변조하는 방법으로 경계값 문제를 해결하였으며, 특정 언어에 구속되지 않으면서도 일반적인 해결책을 찾기 위해 노력하였다. Langfuzz는 특정 언어의 소스 코드가 주어졌을 경우, 이를 파편화하고 학습하여 특정 언어의 문법을 스스로 알아가도록 설계되었다. 또한, 크래시를 발생시켰던 코드는 이후에도 크래시를 발생시킬 확률이 높다는 가정하에 다시 한번 학습을 진행한다. 이처럼 Langfuzz는 학습한 정보들을 바탕으로 스스로 코드를 생성하고 변조하면서 입력값을 생성한다. 이를 통해 일반화된 퍼저를 제작하였지만 복잡한 코드를 생성할 경우 무의미한 코드가 될 확률이 높아지기 때문에 작은 코드를 생성하는 한계가 있으며, 좁은 범위의 경계값에 대해서만 퍼징을 수행하게 된다.

2.2 기존의 코드 커버리지 퍼저

AFL(American Fuzzy Lop)[8]은 유전 알고리즘을 적용하여 코드 커버리지 확대를 지향하는 대표적인 퍼저이다. 퍼징을 수행할 대상의 소스 코드의 모든 분기점마다 진입 여부를 확인할 수 있는 코드를 삽입하고, 퍼징을 수행하며 프로그램의 실행을 추적하는 방법으로 퍼징 대상의 수행 흐름을 수집한다. AFL은 변조 작업을 수행하면서, 전반적인 코드 흐름이 유일하더라도, 새로운 상태 전이를 발생하지 않는 입력값은 버리는 과정을 통해 코드 커버리지를 확대한다. 이러한 확대 방식은 실제로 코드 커버리지를 확대할 수는 있지만, 경계값에 도달하였는지를 확인할 수 없으므로 경계값에 대해 충분한 퍼징을 수행할 수 없다는 한계점이 있다.

2.3 정리

기존의 연구를 통해 그동안 문법 기반 퍼징[3, 4, 6, 7]과 코드 커버리지와 관련된 연구[8, 9]가 활발하게 진행됐음을 확인할 수 있었다. 하지만 대다수 자바스크립트 퍼저에 대한 연구는 해석기의 문법 검사를 통과하는 방법론에 초점을 두어 연구가 진행되어 왔기 때문에 자바스크립트 엔진의 내부 로직에 대한 퍼징에는

다소 미흡한 점이 있었으며, 별도의 소스 코드 분석 행위가 퍼징에 영향을 줄 수 없었다. 따라서 본 연구에서는 휴리스틱 기법을 활용한 부분 재생성 과정을 통해 코드 커버리지를 극대화함으로써, 선행 분석 결과에 의해 도출한 정보를 퍼징에서도 활용할 수 있도록 하여 코드 커버리지 확대 과정에 방향성을 유도하는 방법을 제안한다.

III. HastFuzz 작동 원리

퍼징의 경우 임의적인 입력을 생성하기 때문에 경계값 조건을 찾기 어렵다는 단점이 있었다. 이러한 문제를 해결하기 위해 LangFuzz를 비롯한 대부분의 퍼저는 기존의 사용자 입력에 변화를 일으키는 방식의 접근을 제안하였다. 그러나 이러한 접근은 소프트웨어에 대한 분석과 퍼징의 과정을 분리하기 때문에, 기존의 알려진 취약점이나 소스 코드 분석을 통해 도출한 정보를 퍼징에서 활용할 수 없다는 한계가 있었다.

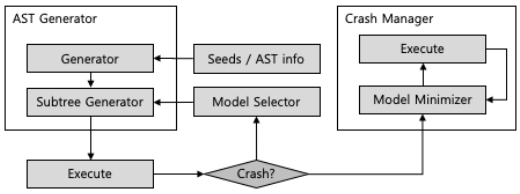


그림 1 HastFuzz 동작 순서

이에 본 연구에서 제안하는 HastFuzz는 통계적인 AST 생성으로 경계값 조건 문제를 해결하고, 코드 커버리지에 기반한 우선순위에 따라 선택된 모델을 부분 재생성하며 퍼징을 수행한다. 이때, 부분 재생성과 모델 선택 과정에 휴리스틱 기법을 적용하여 특정 코드에 대해 더욱 충분한 퍼징을 수행하도록 유도한다. [그림 1]은 제안하는 퍼징 방법론을 도식화한 것으로 이를 통해 소스 코드 분석 등을 비롯하여 선행 분석을 통해 도출한 정보를 퍼징에 적용할 수 있다.

3.1 AST 생성

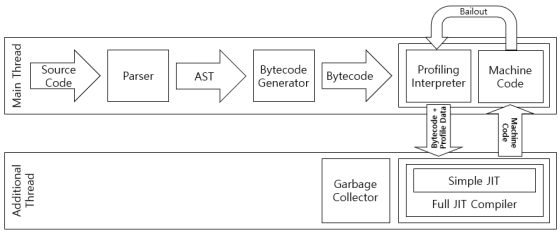


그림 2 ChakraCore의 JavaScript 엔진 구조

브라우저의 자바스크립트 엔진에 대한 퍼징을 효율적으로 수행하기 위해서는 먼저 자바스크립트 엔진의 구조를 파악할 필요가 있다. 자바스크립트 코드가 엔진에서 실행되기 위해서는 [그림 2]와 같은 실행 과정[9]을 거친다. 자바스크립트 엔진의 해석기는 내장된 문법 검사를 통해 입력된 자바스크립트 소스 코드의 실행 가능 여부를 판단하기 때문에, 생성되는 모든 자바스크립트 소스 코드는 이를 만족해야 한다. 이에 본 연구는 이를 효과적으로 만족시키기 위해 AST 확장을 통한 소스 코드 생성 방법을 제안한다.

3.1.1 AST 노드

AST는 소스 코드를 트리 형태로 표현한 구조이며, 이를 바탕으로 실제 실행에 필요한 바이트 코드를 생성한다. AST는 노드와 간선 형태로 표현되며, 각 노드는 종류에 따라 표현식과 설정해야 하는 속성이 결정된다. 각 속성에는 할당 가능한 자식 노드의 유형이 문법 규칙에 의해 정해져 있으며, 생성된 AST는 자바스크립트 소스 코드로 직렬화하는 과정에서 표현식을 참조한다.

3.1.2 AST 노드 관계 정의

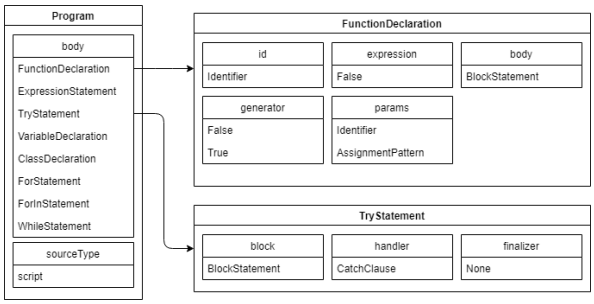


그림 3 AST에서 노드 간의 관계 표현 예시

문법 규칙에 따른 AST 노드의 상하 관계는 [그림 3]과 같이 나타낼 수 있다. AST의 최상

위 노드로 생성되는 'Program' 유형의 노드에는 'body', 'sourceType' 속성이 있으며, 그 중 'body' 속성에는 8종류의 노드가 할당될 수 있음을 나타내고 있다. 이러한 AST 노드 간의 상하 관계 정보는 AST Parser를 이용하여 이미 작성된 자바스크립트 코드를 통해 학습할 수 있다.

HastFuzz는 미리 정의된 AST 노드 관계 정보를 토대로 자식 노드를 선택 생성하고, 자식 노드 또한 이를 반복하여 재귀적으로 트리를 확장해나간다. 이러한 방법을 통해 올바른 문법 구조를 가진 AST를 생성할 수 있다.

3.2 코드 커버리지 극대화

HastFuzz는 AST를 자바스크립트 소스 코드로 직렬화하여 자바스크립트 엔진의 입력으로 실행하여 코드 커버리지를 측정한다. 측정된 코드 커버리지와 AST 쌍을 모델이라 하며, 이러한 모델의 AST에 부분적인 재생성을 하여 새로운 코드 커버리지가 측정되는 모델을 발견해 나간다. HastFuzz는 이러한 과정을 반복하여 코드 커버리지를 극대화 할 수 있다.

3.2.1 코드 커버리지 해시값

HastFuzz는 AST를 자바스크립트 엔진의 입력으로 실행한다. 이때 실행된 모든 분기 주소와 분기별 실행 횟수를 코드 커버리지 정보로 측정한다. 후술할 연산을 효율적으로 처리하기 위해서는 커버리지 정보를 압축할 필요가 있다. 따라서 모든 데이터는 해시함수로 압축하여 관리하며 이러한 압축된 코드 커버리지를 코드 커버리지 해시값으로 정의한다.

3.2.2 AST 부분 재생성

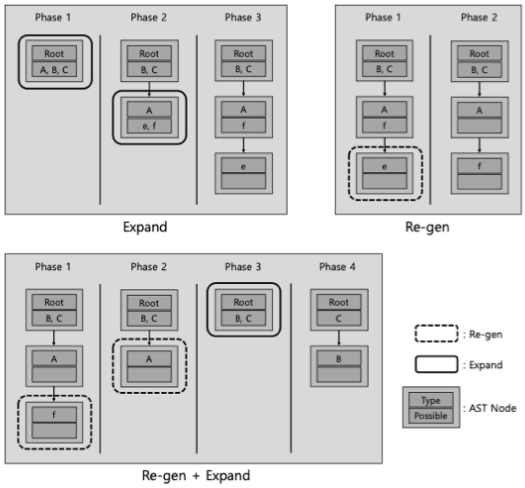


그림 4 상황별 AST 부분 재생성 절차

HastFuzz는 기존 모델의 AST를 부분적으로 재생성하여 새로운 코드 커버리지 해시값이 측정되는 자식 모델을 찾는다. 이때 AST의 반복적인 재생성에도 새로운 AST 구조를 만들어 내는 것을 보장하기 위하여 [그림 4] 같이 각 AST 노드는 할당되었던 자식 노드 유형을 저장하여 노드 확장 단계에서 기존의 노드를 할당하지 않도록 한다.

3.2.3 모델 관리

새로운 코드 커버리지를 효과적으로 발견하기 위해서는 재생성할 모델의 우선순위를 정해야 한다. 본 연구에서는 발견횟수가 가장 적은 모델을 우선 선택하도록 하여 퍼징을 진행하였다.

$$P_1(M) = COUNT(M)$$

퍼징을 수행한 모델M의 우선순위는 위 함수와 같이 동일 모델의 개수로, 최솟값 우선순위 큐에 의해 관리를 하게 된다.

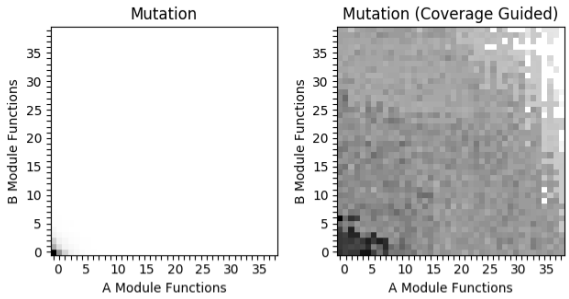


그림 5 코드 커버리지 히스토그램 1

[그림 5]은 간소화된 두 모듈에 대해 퍼징을 16만 회 수행하였을 때 최종적으로 도달한 코드 커버리지를 나타낸 것으로, 각 40개의 경계값 조건의 실행 여부를 측정하였다. 좌측의 히스토그램은 모델 관리 없이 단일 모델에 대해 퍼징을 수행하였다. 우측의 히스토그램은 발견 횟수가 가장 적은 모델을 우선 선택하여 퍼징을 수행하였다. 두 히스토그램을 통해 코드 커버리지 기반의 모델 관리로 더욱 넓은 범위의 코드에 도달할 수 있음을 알 수 있다.

3.3 휴리스틱 기법의 적용

모델의 우선순위 산정과 AST 노드 확장 단계에 휴리스틱 기법을 활용하여 커버리지 극대화 과정에서 특정 실행 분기로 방향성을 부여할 수 있다. 대부분의 분석가들은 이미 알려진 취약점이나 소스 코드 분석을 통해 특정 기능과 상황에서 취약점이 자주 발생할 수 있다는 것을 경험적으로 유추할 수 있다. 이러한 경험에 의한 정보를 퍼징에 반영하여 커버리지 극대화 과정에 방향성을 부여하는 방법을 제안한다.

3.3.1 코드 커버리지 휴리스틱

모델 선택 단계에서, 자바스크립트 엔진의 특정 루틴을 실행한 모델에 높은 우선순위를 부여한다. 이를 통해 특정 경계값 조건에서 더 많은 퍼징을 수행하도록 유도할 수 있다. 이때 측정되는 루틴은 중첩될 수 있다.

$$H_{cov}(M) = \sum_{k=1}^{n} isTriggered(F_k)$$

$$P_2(M) = COUNT(M) - H_{cov}(M)^2$$

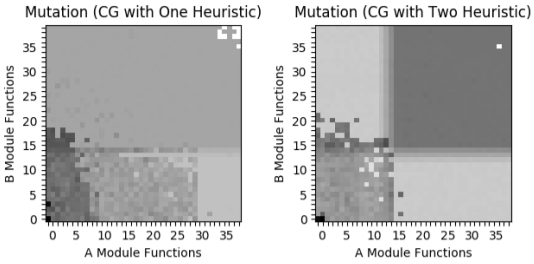


그림 6 코드 커버리지 히스토그램 2

[그림 6]은 간소화된 두 모듈에 대해 퍼징을 12만 회 수행하였을 때 최종적으로 도달한 코드 커버리지를 나타낸 것으로, 좌측 히스토그램은 총 4개(B모듈 4개 분기)의 커버리지 측정을, 우측 히스토그램은 총 9개(A모듈 5개 분기, B 모듈 4개 분기)의 커버리지 측정을 우선순위 산정에 활용한 결과이다.

3.3.2 AST 휴리스틱

노드 확장 및 재생성 단계에서 AST 노드의 속성 할당 시에 참조하는 AST 관계 정보에 가중치를 부여하는 방법을 통해 특정 기능과 조건에 확률적으로 더 적은 횟수의 퍼징으로 도달하도록 유도를 할 수 있다. 이는 코드 커버리지 휴리스틱과 함께 활용할 경우 좀 더 가시적인 결과를 확인할 수 있다.

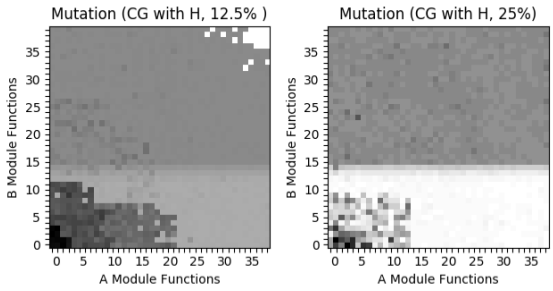


그림 7 코드 커버리지 히스토그램 3

[그림 7]은 간소화된 두 모듈에 대해 퍼징을 12만 회 수행하였을 때 최종적으로 도달한 코드 커버리지를 나타낸 것이다. 우측 히스토그램에 해당하는 테스트에서 선택된 모델에 대해 퍼징을 수행할 때, B 모듈의 코드 커버리지에 영향을 줄 수 있는 도메인을 우선 선택하도록 가중치를 부여하였다. 이로 인해 코드 커버리지 휴리스틱을 통해 유도하려던 특정 경계값 조건에 좀 더 적은 횟수의 퍼징으로 도달하게 된 것을 확인할 수 있었다.

IV. 결론 및 추후 연구

본 논문에서는 휴리스틱을 이용하여 코드 커버리지 확장에 방향성을 가지도록 유도하는 퍼징 방법론을 제안하였다. 기존의 자바스크립트 퍼저들은 해석기의 문법 검사를 통과하기 힘들거나, 이를 통과하더라도 퍼징 결과값을 활용하지 않는 등의 비효율적인 부분이 존재하였다. 또한, 임의의 코드를 생성하여 테스트를 수행하는 접근은 대상 소프트웨어에 대한 분석 행위

와 퍼징 과정을 분리하기 때문에 기존에 알려진 취약점이나 소스 코드 분석을 통해 도출한 정보를 퍼징에서 활용할 수 없다는 한계가 있었다.

이러한 한계점들을 개선하기 위해 본 논문에서 제안한 HastFuzz는 노드 간의 상하 관계를 만족하는 AST를 생성함으로써 해석기의 문법 검사를 통과하고, 휴리스틱 기법을 활용하여 코드 커버리지 극대화에 방향성을 부여하는 접근은 분리되어 있었던 소프트웨어 분석과 퍼징의 과정을 결합하여 취약점 탐색의 효율을 높여줄 것으로 기대한다.

하지만 HastFuzz는 코드를 생성하고 실행하는 과정의 실행 속도 문제와 코드 커버리지를 측정하는 과정에서 오버헤드가 발생할 수 있다는 한계가 있다. 이를 위해, 실행 속도 개선 방안에 대한 추가 연구와 코드 커버리지 확장을 위한 휴리스틱 기법의 효율적인 적용 방안 대한 추가 연구를 진행할 계획이다.

[참고문헌]

[1] TGuo, Tao, et al. GramFuzz: Fuzzing testing of web browsers based on grammar analysis and structural mutation. Informatics and Applications (ICIA), 2013 Second International Conference on. IEEE, 2013.

[2] Dewey, Kyle, Jared Roesch, and Ben Hardekopf. Language fuzzing using constraint logic programming. Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM, 2014.

[3] Godefroid, Patrice, Adam Kiezun, and Michael Y. Levin. Grammar-based whitebox fuzzing. ACM Sigplan Notices. Vol. 43. No. 6. ACM, 2008.

[4] Godefroid, Patrice, Michael Y. Levin, and David Molnar. SAGE: whitebox fuzzing for security testing. Queue 10.1 (2012): 20.

[5] Rebert, Alexandre, et al. Optimizing Seed Selection for Fuzzing. USENIX Security Symposium. 2014.

[6] Holler, Christian, Kim Herzig, and Andreas Zeller. Fuzzing with Code Fragments. USENIX Security Symposium. 2012.

[7] RUDERMAN, J. Introducing jsfunfuzz. Blog Entry. http://www.squarefree.com/2007/08/02/introducing-jsfunfuzz/, 2007.

[8] M. Zalewski. American Fuzzy Lop. http://lcamtuf.coredump.cx/afl/.

[9] ChakraCore Architecture-Overview, https://github.com/Microsoft/ChakraCore/wiki/Architecture-Overview

효율적인 악성코드 분석을 위한 에뮬레이션 기반 동적 분석 도구 구현 및 검증

김민재*임명은*주하종*염흥열**

*순천향대학교 정보보호학과

Implementation and verification of an emulation based dynamic analysis tool for effective malware analysis

Min Jae Kim*, Myeong eun Im*, Ha Jong Joo*, Heung Youl Youm**

*Department of Information Security Engineering, SoonChunHyang University.

요 약

코드 난독화(Code Obfuscation)는 프로그램의 코드를 일부 또는 전체를 읽기 어렵게 변환하여 역공학(Reverse Engineering)을 방해하는 기술을 말한다. 일반적으로 이러한 기술은 소프트웨어 보호를 목적으로 사용되었지만, 악성코드에서 사용되기 시작하면서 제작자들이 안티 바이러스(Anti-Virus) 소프트웨어의 탐지를 피하고 악성코드 분석가의 역분석을 어렵게 하기 위해 많이 사용되고 있다. 코드 난독화 기법 중에는 자체적으로 데이터 인코딩을 통해 API에 대한 직접적인 호출을 숨기는 방식이 존재한다. 본 논문에서는 에뮬레이션 기반의 분석도구를 구현하고 이를 수치화한 뒤 검증하여 효율적인 악성코드 분석 방안에 대하여 제안을 하고자 한다.

I. 서론

최근 악성코드는 랜섬웨어, 파밍, 피싱 등 여러 형태로 유포되고 있으며, 유포 방식 또한 이메일, 사회 공학적 공격 기법(Social Engineering Attack), 드라이브 바이 다운로드(Drive-by-Download) 기법 등 다양한 형태를 보이고 있다.[1] 유포되는 악성코드들 중 자체 난독화 또는 인코딩을 수행하여 파일의 분석을 어렵게 하는 악성코드가 다수 존재한다. 따라서 이러한 형태의 악성코드에 대해 악성코드 분석가는 분석하기 힘든 어려움을 가지고 있다.

악성코드의 분석 도구로 자주 사용되는 IDA (Interactive Disassembler)는 파일에 대한 디스어셈블링(Disassembling)을 통해 기계어 코드로 변환하여 어셈블리어 소스를 생성하고 이에 대한 디버깅을 가능하게 하는 툴이다.[2] 그 중 IDA 툴에서 제공하는 기능 중 하나인 IDAPython은 Python 플러그인의 제공을 통해 조금 더 분석을 용이하게 한다.[3] 해당 기능을 이용하면 IDA에서 Python 스크립트 파일을 실행 가능하며, 이러한 스크립트 작성을 통해 문자열을 통해 자체 난독화 및 인코딩된 악성코드에 대해 동적분석처럼 악성코드를 직접 실행하지 않아도 난독화를 해제 할 수 있다.

본 논문에서는 IDAPython 기능을 통해 직접 악성코드를 실행하지 않으면서 악성코드의 자체 난독화 및 인코딩을 해제하여 효율적인 악성코드 분석을 위한 분석도구를 구현하는 과정을 보여주고 이에 대한 검증을 하며 이후 구성은 다음과 같다. 2장에서는 악성코드 내 난독화와 관련된 이전 연구 사례와 코드 난독화와 바이너리 인코딩 기법에 대해 살펴본다. 이후 3장에서는 실제로 시스템을 설계하였고 시스템의 수행 결과를 검증하였다.

II. 관련 연구

2.1 이전 연구 사례

이전의 사례를 살펴보면, 해외에서는 Tomer Teller의 'API-Trigger' 기반의 메모리 덤프를 통한 자동화된 악성코드 분석 연구[4]와 국내에서 이를 보완하여 설계한 'Mem-shot'이라는 난독화된 악성코드 분석 시스템 설계 및 구현[5]에 대한 연구등이 존재한다. 이들은 공통적으로 별도의 'Cuckoo Sandbox'라는 샌드박스 분석 시스템을 통해 동적으로 난독화를 해제하는 방안을 설계하고 구현하였다. 앞선 연구들에서는 별도의 가상 시스템을 이용해야 했기 때문에 성능상의 한계점이 있었고, 분석가의 로컬 환경에서 직접적으로 분석하지 못한다는 한계점이 있다. 본 논문에서는 별도의 가상환경을 구성하지 않더라도 로컬 환경에서 분석하여 결과를 도출할 수 있도록 효율적인 악성코드 분석 시스템을 제안하였고, 이어지는 장에서 이에 대한 설계와 검증을 하였다.

2.2 코드 난독화

코드 난독화(Code Obfuscation)는 프로그램의 코드를 일부 또는 전체를 읽기 어렵게 변환하여 역공학(Reverse Engineering)을 방해하는 기술을 말한다.[6] 난독화가 적용된 프로그램은 일반적인 프로그램을 분석하는 것보다 훨씬 많은 비용이 들기 때문에 최근의 악성코드들은 자체 난독화 기술을 적용해 유포되고 있다. 코드 난독화의 기술은 크게 4가지의 종류가 있으며, 구획 난독화(Layout obfuscation), 데이터 난독화(Data obfuscation), 집합 난독화(Aggregation obfuscation), 제어 난독화(Control obfuscation)로 나눠진다.[7] 본 논문에서는 데이터 난독화에 대해 다룬다.

2.3 바이너리 인코딩 기법

인코딩이란 정보의 형태나 형식을 변환하는 처리 기법을 말하며[8], 대표적인 예시로는 자주 볼 수 있는 Base64 인코딩 기법이 존재한다. 일반적으로 인코딩은 문자나 기호를 컴퓨터에서 표현할 목적으로 사용하거나 네트워크 기반 통신의 암호화를 위해 사용한다. 이러한 기법을 악성코드에서는 내부 작업을 숨기기 위해서 사용하는데, 바이너리 내부에서 사용하는 문자열을 인코딩하여 저장하고 이를 사용하기 직전에 특정 디코드 함수를 통해 본래의 문자열을 디코딩하여 사용하는 인코딩 기법을 사용한다.

III. 에뮬레이션 기반 시스템 설계

3.1 접근 방법

그림 1은 'Backdoor:Win32.Akdoor.C1354644'(Ahnlab)라는 탐지명을 가진 악성코드에서 인코딩된 함수명을 디코딩해 각 변수에 저장하는 루틴이다.

```
v154 = 20685;
LODWORD(v3) = sub_180001680(v1, &v146, 34i64);
v42 = -2075943050;
v43 = -867645576;
qword_180026970 = v3;
v44 = -915055511;
v45 = -174486764;
v46 = 373580717;
v47 = 891036123;
LODWORD(v4) = sub_180001680(v2, &v42, 24i64);
qword_180026780 = v4;
v105 = -989670339;
v106 = -1086615760;
v107 = -1307693800;
v108 = -1795624339;
v109 = 2083390156;
v110 = 389966480;
v111 = 1211038293;
v112 = -25624;
LODWORD(v5) = sub_180001680(v2, &v105, 30i64);
```

[그림 1] 악성코드 내 인코딩 된 함수명

그림 1에서 해당 악성코드는 API 함수를 사용하기 위해 인코딩 되어있는 API 이름을 디코딩하고, 디코딩된 함수 이름을 통해 함수 주소를 구한다. 따라서 프로그램을 실행시킨 후, 디코딩 루틴이 실행된 후의 메모리를 확인하면 디코딩된 함수명을 확인할 수 있다. 그러나 악성코드는 실행 시 악성코드에 감염되므로 로컬 환경에서 실행시킬 수 없다는 제약이 존재한다.

본 논문에서는 이러한 디코딩 루틴을 에뮬레이터로 실행시키는 방식을 제안한다. 이러한 방식은 실제 환경이 아닌 에뮬레이터 상에서 프로그램을 실행하기 때문에 악성코드에 감염될 위험이 없으며, 서로 다른 악성코드에 대해 디코딩 루틴을 별도로 분석하지 않아도 되기 때문에 디코딩 루틴 혹은 디코딩 키가 달라지더라도 알고리즘을 수정할 필요가 없다.

추가적으로 IDAPython 기능을 이용해 디코딩된 모든 값을 분석 정보로 저장함으로써 분

석에 소요되는 시간을 크게 줄였다.

3.2 시스템 설계

본 논문에서는 악성코드에서 인코딩 된 문자열을 디코딩하는 루틴을 자동화하도록 시스템을 설계하고 구현하였다. 구현에 사용된 시스템 환경과 툴은 다음과 같다.

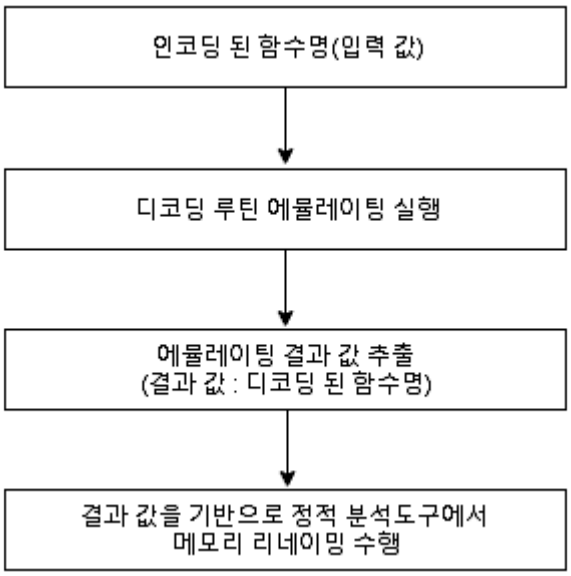
- 분석환경

(1) OS : Windows 7 SP1 x64 bit

(2) Language : Python 2.7.15[9]

(3) Programs : Triton[10], IDA Pro

악성코드에서 실행하는 디코딩 루틴의 주소 정보를 이용해 에뮬레이터에서 해당 주소와 함께 인코딩 된 문자열을 인자로 함께 실행시킨 후 디코딩 된 문자열을 반환하는 방식의 시스템이다. 디코딩 된 문자열을 반환한 후에는 IDA의 rename 기능을 통해 이후 코드에서 직관적으로 디코딩 된 문자열을 볼 수 있도록 하였다. 또한 추가적으로 프로그램의 검증을 위해 자체적으로 여러 악성코드 샘플들을 수집하였다. 해당 샘플들은 본문에서 언급한 바이너리 인코딩 기법을 사용하며, 특정한 디코딩 함수를 통해 문자열을 복호화하여 사용하는 특징이 있다. 아래 그림 2는 전체 시스템의 도식도이다.



[그림 2] 전체 시스템 도식도

IV. 수행 결과

본 장에서는 설계된 프로그램 검증을 위해 수집한 샘플에 대하여 에뮬레이팅을 수행한 결과 값에 대해 수치화하여 보여주고자 한다. 또한 제안 방법을 검증하기 위해 다음의 과정을 통해 진행하였다.

- 검증방법

a. 디코딩 함수의 주소값 추출 및 에뮬레이팅

b. 에뮬레이팅 결과 값 추출

c. 결과 값 기반 메모리 리네이밍

- 악성코드 샘플(MD5)

탐지명 : Backdoor:Win32.Akdoor.C1354644 (Ahnlab)

Case 1 : 1609bfd16a255cad5acbdd97df671558

Case 2 : a1e399523c45ff2f13f82db3158ceee1

Case 3 : 78673b473fdf62b4983e4092b238715e

Case 4 : cc9e9f0b736a8530cece7be560c8c5d1

Case 5 : 63a3fb12a161e231b633dd24b8052525

Case 6 : d09396a6d83c07795258274e6781e727

Case 7 : 1216da2b3d6e64075e8434be1058de06

Case 8 : 8739b39b7d935e38160f68b43863d795

```
qword_180026840 = sub_180001680(v1, (const WCHAR *)&v53, 0x12u);
v106 = -29760244;
v107 = -2113491439;
v108 = -1909322440;
v109 = -2115119543;
v110 = 1765477317;
v111 = 288909458;
v112 = 759954005;
qword_180026798 = sub_180001680(v2, (const WCHAR *)&v106, 0x1Cu);
v81 = -1317623229;
v82 = -716054178;
v83 = -981905307;
v84 = -1062429948;
v85 = 862016156;
v86 = 571160268;
v3 = sub_180001680(v2, (const WCHAR *)&v81, 0x18u);
```

[그림 3] 에뮬레이팅 수행 전 함수

위의 그림 3은 에뮬레이팅 수행 전 악성코드 내 인코딩 된 함수들의 모습이다.

```
shutdown_rr = sub_180001680(v1, (const WCHAR *)&v53, 0x12u);
v106 = -29760244;
v107 = -2113491439;
v108 = -1909322440;
v109 = -2115119543;
v110 = 1765477317;
v111 = 288909458;
v112 = 759954005;
gethostbyname_rr = sub_180001680(v2, (const WCHAR *)&v106, 0x1Cu);
v81 = -1317623229;
v82 = -716054178;
v83 = -981905307;
v84 = -1062429948;
v85 = 862016156;
v86 = 571160268;
v3 = sub_180001680(v2, (const WCHAR *)&v81, 0x18u);
```

[그림 4] 에뮬레이팅 수행 후 함수

위의 그림 4는 디코딩 루틴을 에뮬레이팅 수행 후 결과 값을 이용해 정적 분석도구인 IDA

내에서 함수 명을 메모리 리네이밍한 것이다. 앞선 그림 3에 대비하여 난독화되었던 변수들이 해제되어 shutdown, gethostbyname과 같은 API 형태로 나타나 이후 악성행위에서 사용된다.

[표 1] 분석 결과

	디코딩 함수 호출 횟수	디코딩 성공률(용어 수정)
Case 1	130	100%
Case 2	130	100%
Case 3	130	100%
Case 4	130	100%
Case 5	130	100%
Case 6	130	100%
Case 7	130	100%
Case 8	130	100%

본 논문에서는 총 8개의 샘플 케이스에 대해서 시스템을 수행하였다. 모든 케이스는 각각 130개의 디코딩 함수를 호출하였고 본 논문의 시스템을 수행한 결과 모든 디코딩 함수를 에뮬레이팅 수행하여 함수명을 가시적으로 나타내었다.

V. 결론

본 논문에서는 악성코드에 사용된 코드 난독화 기법 중 바이너리 인코딩 기법에 대해 조사하고, 에뮬레이팅을 통한 효율적인 악성코드 분석에 대한 방법을 제안하고 이를 검증하였다. 또한, 앞서 언급했던 이전 연구의 단점인 가상환경에서의 악성코드 난독화 해제 방안의 한계점이었던 성능 문제와 로컬 환경에서 직접적으로 분석 불가의 한계점을 보완하였다. 따라서 본 논문에서 제안한 방법을 이용하면 바이너리 인코딩 기법이 사용된 악성코드에 대한 분석에 있어 인코딩 된 함수명이 API 형태로 나타나기 때문에 분석시간을 단축시켜 효율적인 악성코드 분석에 도움을 줄 수 있으리라 예상한다.

[참고문헌]

[1] 백재종. 2016. 악성코드 인젝션 사이트 탐지를 통한 방어효율 향상방안. 한국정보통신학회논문지. 20(7): 1290-1295

[2] IDA, https://www.hex-rays.com

[3] IDAPython, https://github.com/idapython/src

[4] Tomer Teller, Enhancing Automated Malware Analysis Machines with Memory Analysis Blackhat 2014

[5] 최상훈, 박기웅. 2016. Mem-Shot: 악성코드 난독화 분석을 위한 API-Trigger 기반의 메모리 덤프 시스템 설계 및 구현. 한국차세대컴퓨팅학회논문지. 12(2): 23-32

[6] 전준수, 한태숙. 2012. 가상화 기법으로 난독화된 실행 파일의 동적 분석 방법. 정보과학회논문지. 40(1): 61-71

[7] 이병용, 최용수. 2008. Obfuscation 기술의 현황 및 분석과 향후 개발 방향. 보안공학연구논문지. 5(3): 219-228

[8] 김선범. 2018. 딥러닝을 이용한 문서의 인코딩 및 언어 판별. 한양대학교 대학원 석사학위논문.

[9] Python, https://www.python.org/

[10] F. Saudel, J. Salwan.. 2015. Triton: A Dynamic Symbolic Execution Framework, SSTIC. In Symposium sur la s´ecurit´e des technologies de l'information et des communications. 31-54.

민간 클라우드 도입 장애요인 분석을 통한 국방 클라우드 도입 전략 도출

박준규* 전우진* 이상훈** 박기웅†

*세종대학교 시스템보안연구실 **국방과학연구소 †세종대학교 정보보호학과

Deployment Strategies of Military Cloud by Analyzing Obstacles to Deployment of Commercialized Cloud Services

Jun-Gyu Park* Woo-Jin Jeon* Sang-Hoon Lee** Ki-Woong Park†

*SysCore Lab., Sejong University

**Agency for Defense Development

† Dept. of computer and information security, Sejong University

요 약

클라우드 컴퓨팅은 인터넷 기술을 활용하여 사용자 요구에 따라 컴퓨팅 자원(네트워크, 서버, 스토리지, S/W)을 서비스로 제공하는 컴퓨팅으로 많은 관심을 받고 있다. 최근 국방 분야에서도 이와 같은 패러다임에 맞춰 클라우드 컴퓨팅 기술에 대한 도입을 추진하고 있다. 기존 연구는 주로 국가 정책이나 공공 기관 중심의 발전 방안을 주로 다루었지만 산업의 근간이 되는 기업 입장에서의 연구는 부족하였다. 따라서 본 논문에서는 국내 기업의 클라우드 도입 장애요인을 서비스 가용성 및 보안 관점에서 분석하여 해당 장애요인을 극복하는 성공적인 국방 클라우드 도입을 위한 전략을 제시하고자 한다.

I. 서론

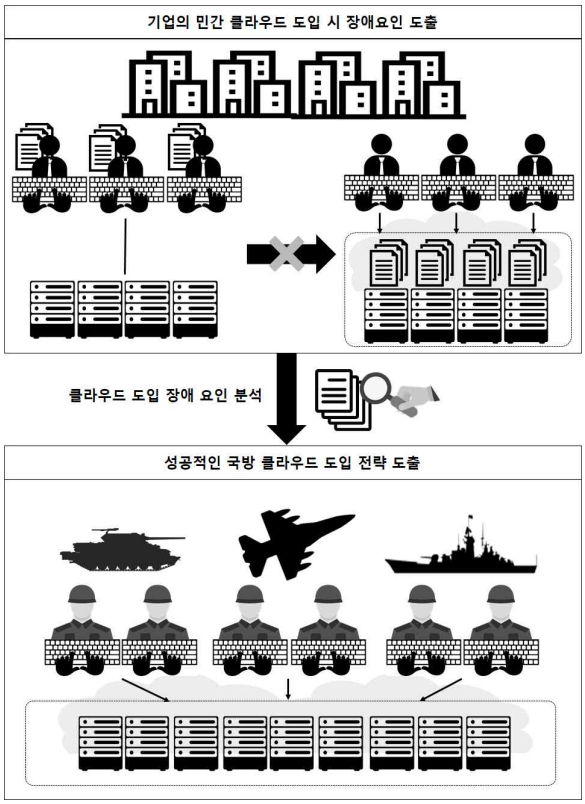
클라우드 컴퓨팅(Cloud Computing, 이하 '클라우드'로 표기함)은 '인터넷 기술을 활용하여 사용자 요구에 따라 컴퓨팅 자원(네트워크, 서버, 스토리지, S/W)을 서비스로 제공하는 컴퓨팅'이다[1]. 클라우드는 기업 내·외부에 대규모의 IT자원을 확보한 뒤, 인터넷을 통해 접근할 수 있게 하여 규모의 경제를 통해 비용 효율화가 가능하다. 또한, IT자원의 수요가 발생한 시점 혹은 수분 내에 IT자원 확보가 가능하여 클라우드 사용자는 신속한 사업 대응이 가능하게 된다. 다양한 IT자원뿐만 아니라 애플리케이션 및 서비스 등을 통해 기업에게 큰 부담으로 작용하였던 IT에 대한 부담을 축소시켜 기업 및 기관에서는 핵심 사업 및 업무에 대한 집중이 가능하게 되었고, 이러한 이유로 클라우드는 현대 핵심 IT 전략이 되었다.

최근 국방 분야에서도 이와 같은 패러다임에 맞춰 클라우드 컴퓨팅 기술에 대한 도입을 추진하고 있다. 안보경영연구원(SMI)에서 2016년에 진행한 '국방 클라우드 컴퓨팅 운영환경 구축방안 연구[2]'를 통해 국방정보자원 운용의 효율화를 위해 클라우드 도입의 필요성을 인식하여 육·해·공군 전산소를 통합한 국방통합데이터센터(DIDC)를 설립하였으며, 전군 공통의 특정 시스템을 대상으로 일부 IaaS(Infrastructure as a Service) 형태의 클라우드 서비스를 제공하고 있는 것을 알 수 있다. 또한, 2013년에 발표한 '국내 클라우드 정책 분석 및 발전방향에 관한 연구[3]'에서는 다양한 프레임워크를 통해 정부의 클라우드 정책을 분석하고, 문제점을 제시하였으며 우리나라의 클라우드 확산 정책이 서비

† 교신저자: 박기웅 (세종대학교 정보보호학과 교수)

이 연구는 국방과학연구소의 국방 지휘통제 통합·연동 기반기술 특화연구실 과제의 지원(UD180012ED) 및 2018년도 과학기술정보통신부의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2018-0-00420, API 호출 단위 자원 할당 및 사용량 계량이 가능한 서버리스 클라우드 컴퓨팅 기술 개발).

스 소비자가 아닌 서비스 공급자 중심이고, 민간 중심이 아닌 정부 중심이라는 점을 지적하였다. 이를 통해 기존 연구는 주로 국가 정책이나 공공 기관 중심의 발전 방안을 주로 다루었지만 산업의 근간이 되는 기업 입장에서의 연구는 부족한 것을 알 수 있다.



[그림 1] 국방 클라우드 도입 전략 도출 과정

2009년 한국클라우드산업협회(KACI) 민영기의 '클라우드 서비스 활성화를 위한 장애요소 및 대응방안[4]'에서는 클라우드 서비스 활성화 저해요인 중 하나로 클라우드 서비스의 안정성에 대한 우려를 꼽았다. 이는 클라우드를 도입하려는 기업과 국방 분야에 장애요인으로 작용하여 클라우드 도입을 저해하고 있다[5].

클라우드 도입을 저해하는 주요 요인 중 또 다른 하나는 보안이다. 시장조사기관인 IDC가 2016년에 진행한 클라우드 보고서(Cloud Going Mainstream)[6]에는 응답기업의 33%와 1천명 이상 규모의 기업 50%가 보안을 클라우드 도입의 장벽으로 제기하였다. 또한, 2018년 과학기술정보통신부에서 발표한 '2018년 공공부문 클라우드 컴퓨팅 수요조사 결과[7]'에서는 1225개 기관을 대상으로 클라우드 수요조사를 실시하였고, 공공부문의 클라우드 미도입 사유로 보안 우려 및 규제가 24.4%를 기록하였다. 이를 통해 기업 및 공공 분야의 클라우드 도입 장애요인으로 보안에 대한 우려가 존재하는 것을 알 수 있으며, 이는 정보 자원의 중요도가 높은 국방 분야에도 해당 되는 요인이다. 위의 장애요인 외에도 기업의 민간 클라우드 도입 시 발생하는 다양한 장애 요인들이 존재한다.

본 논문에서는 국방 클라우드 도입과 관련성이 높고 기술적으로 해결 가능한 장애 요인들에 대하여 해결 방안을 제시하기 위해 이에 해당하는 장애요인인 서비스 가용성 및 보안 문제를 다룬다. 또한, 정부의 입장이 아닌 클라우드 수요자(사용자) 중심으로 현재 상황을 이해하기 위해 [그림 1]과 같이 국내 기업들의 민간 클라우드 도입을 저해하는 요인을 분석하고, 이를 통해 국방 분야에서 성공적인 클라우드 도입을 위한 방안을 제시하고자 한다.

논문의 구성은 다음과 같다. 2장에서는 국내 기업의 클라우드 도입 장애요인을 서비스 가용성 및 보안 관점에서 분석한다. 3장에서는 분석된 장애요인을 극복하는 성공적인 국방 클라우드 도입을 위한 전략을 제시한다.

II. 기업의 민간 클라우드 도입 시 장애 요인 분석

2.1 서비스 가용성 문제

기업에서 클라우드 도입 후 서비스 장애 발생 시 고객들에 대한 안정적인 서비스 제공 불가능 및 기업 내 업무 처리 지연 등의 문제가 발생한다. 고객들에게 안정적인 서비스를 제공하지 못하는 것은 기업의 신뢰도 및 이익과 직결되는 문제이다. 이에 따라 기업에서는 기존 시스템에서 클라우드로 전환 후 발생할 수 있는 서비스 장애에 대한 우려로 인해 클라우드를 도입하지 않는 결론에 도달하게 된다.

2.2 보안에 대한 우려

클라우드에서 가상화 기술과 하나의 소프트웨어를 여러 테넌트(사용자)가 공유해서 사용할 수 있도록 하는 멀티테넌시 기술은 핵심 기술이다. 하지만, 이러한 기술을 사용함으로 인하

여 인증 및 접근제어의 복잡도가 상승한다. 이에 따른 관리 부주의 및 설정 오류 등으로 클라우드 이용자의 정보가 유출되는 문제가 발생할 수 있고, 이는 클라우드 보안의 핵심 이슈이다[8]. 특히, 기업에서 경영정보 및 개인정보와 같은 민감한 정보들이 잘못 관리되어 보안 사고가 발생하면 기업에게는 큰 타격이 발생한다.

III. 성공적인 국방 클라우드 도입 전략

UC 버클리대학교에서 연구한 '09년 클라우드 컴퓨팅 확산 시 예측되는 10가지 장애요인과 해결방안[9]' 결과 중 서비스 가용성에 해당하는 해결방안과 2017년 소프트웨어정책연구소(SPRi)에서 발표한 '클라우드 보안의 핵심이슈와 대응책[8]'에서의 보안 문제 해결방안은 아래 표 1에 나타나있다. 하지만, 민간과 국방의 상이한 특성으로 인해 민간에서의 해결방안을 국방에 완벽히 적용시키는 것은 불가능하다. 따라서 본 장에서는 민간에서의 해결방안을 토대로 성공적인 국방 클라우드 도입 전략을 제시한다.

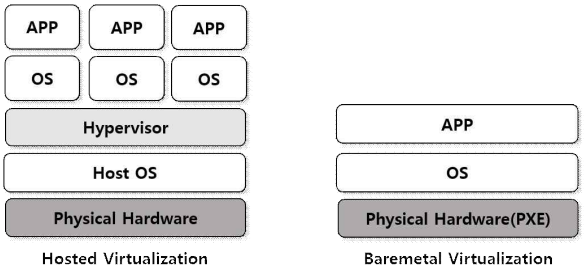
장애요인	민간에서의 해결방안	국방에서의 해결방안
서비스 가용성	다수의 클라우드 공급자 활용	- 베어메탈 방식을 이용한 하이브리드 클라우드 도입
보안에 대한 우려	기존의 보안 방식의 재구성을 통한 방어 체계 구축	

[표 1] 클라우드 컴퓨팅 확산 시 기술적 장애요인과 민간 및 국방에서의 해결방안

3.1 베어메탈 방식을 이용한 하이브리드 클라우드 도입

본 논문에서의 하이브리드 클라우드(Hybrid Cloud)란 [그림 2]와 같이 하드웨어의 리소스를 공유하는 호스트 기반 가상화 방식과 리소스를 공유하지 않는 격리된 환경에서의 단독 인프라가 제공되는 베어메탈 방식을 혼용하여 구성하는 클라우드 방식이다.

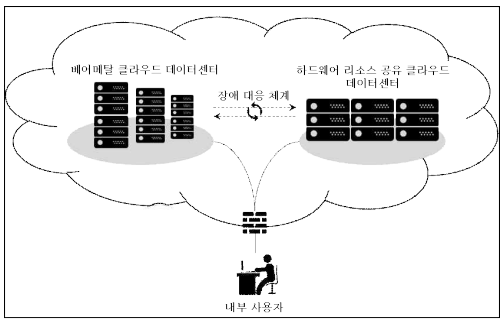
베어메탈이란 PXE(Preboot Execution Environment)를 활용하여 네트워크로 운영체제를 부팅할 수 있게 해주는 환경이다. PXE란 BIOS 및 NIC(Network Interface Card)를 통해 디스크 대신 네트워크에서 컴퓨터를 부트스트랩을 할 수 있도록 해주는 기술이다. 네트워크를 통해 시스템을 부팅 할 수 있는 기능을 사용하여 서버를 배포 및 관리한다. 또한, 사용자에게는 하드웨어의 리소스를 공유하지 않는 격리된 환경에서의 단독 인프라가 제공된다. 이러한 베어메탈 방식을 활용한 하이브리드 클라우드를 이용할 경우, 클라우드 도입을 저해하는 요인인 서비스 가용성과 보안에 대한 우려를 해결하는 것이 가능하다.



[그림 2] 호스트 기반 가상화 방식과 베어메탈 방식

3.1.1 물리적으로 구분된 특성으로 서비스 가용성 문제 해결

'K-ICT 클라우드 컴퓨팅 활성화 계획[10]'에 따르면 중앙행정기관, 지자체, 공공기관 대부분 정보 자원의 중요도가 높으므로 민간 클라우드보다 정부 클라우드 및 자체 클라우드를 이용하도록 하고 있다. 이로 인해 높은 중요도를 가지고 있는 군의 정보자원 특성상 자체 클라우드를 구축하여야만 한다. 통합데이터센터를 구축하여 자체 클라우드를 이용할 경우, 서비스 장애 발생 시 집중화된 시스템으로 인해 모든 시스템이 마비될 가능성이 존재한다. 하지만, 하이브리드 클라우드를 도입할 경우 [그림 3]과 같이 데이터센터 간 물리적으로 분리된 특성으로 인해 서비스 장애 발생 시 가용성 확보가 가능한 이점이 있다. 물론, 서로 다른 위치의 데이터센터 간 장애 대응 체계가 구축되어 있어야한다.



[그림 3] 하이브리드 클라우드 구성도

3.1.2 격리된 환경으로 보안에 대한 우려 감소

베어메탈 방식을 이용할 경우 격리된 환경에서의 단독 인프라가 제공되기에 하드웨어 리소스를 공유하지 않으므로 정보가 유출될 가능성이 낮다. 가령, 군에서 호스트 기반 가상화 방식을 이용할 경우 각기 다른 성격을 가진 기관의 유출되어서는 안 될 정보가 관리 부주의 및 설정 오류 등으로 유출될 가능성이 있다. 하지만, 군의 정보자원 중 더욱 높은 중요도를 가진 정보자원과 관련된 군 내부 기관 및 업무 수행의 경우 베어메탈 방식을 이용하여 타 기관 및 업무와의 격리를 통해 정보 유출 가능성을 낮추어 보안에 대한 우려를 감소시키는 것이 가능하다.

IV. 결론

본 논문에서는 국내 기업 관점에서 민간 클라우드 도입을 저해하는 요인을 분석하고 이를 통해 성공적으로 국방 분야에 클라우드를 도입하기 위한 전략을 제안하였다.

우리나라는 클라우드 산업이 활성화 되기 위한 우수한 ICT 인프라를 보유하고 있으며, 정부의 관련 법과 제도 제정으로 지속적으로 클라우드 산업 규모가 성장하고 있다. 하지만, 클라우드 도입을 저해하는 서비스 가용성 문제 및 클라우드 보안에 대한 우려가 존재하고 있다. 이와 같은 문제를 해결하기 위해 본 논문에서는 국내 기업 관점에서 민간 클라우드 도입을 저해하는 요인을 분석하고 이를 통해 성공적인 국방 클라우드 도입 전략으로 베어메탈 방식을 이용한 하이브리드 클라우드를 도입하여 서비스 가용성 문제와 보안에 대한 우려를 감소시키도록 제안하였다.

추후 연구에서는 더욱 다양한 측면에서 클라우드 도입 장애 요인을 분석하여 성공적으로 국방 분야에 클라우드를 도입하기 위한 전략에 대한 연구를 수행할 것이다.

[참고문헌]

[1] P.MELL, T.Grance. “The NIST definition of cloud computing”, 2011.

[2] 안보경영연구원(SMI), “국방 클라우드컴퓨팅 운영환경 구축방안 연구”, 2016.

[3] 백승익, 신지연, 김종우, “국내 클라우드 정책 분석 및 발전방향에 관한 연구”, 한국전자거래학회지 제18권 제3호, 2013.

[4] 민영기, “클라우드 서비스 활성화를 위한 장애요소 및 대응방안”, 한국정보통신기술협회 저널 125호, 2009.

[5] 김성태, “클라우드 컴퓨팅의 동향과 군 도입 시 고려사항(1)”, 한국국방연구원 주간국방논단 통권 제1386호, 2011.

[6] IDC(International Data Corporation), “Cloud Going Mainstream: All Are Trying, Some Are Benefiting; Few Are Maximizing Value”, 2016.

[7] 과학기술정보통신부, 행정안전부, “2018 공공부문 클라우드 컴퓨팅 수요조사 결과”, 2018.

[8] 정성재, 배유미. "클라우드 보안 위협요소와 기술 동향 분석" 보안공학연구논문지 10.2 , 2013.

[9] M. Armbrust and Armando Fox, and et al., "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley Technical Report, No.UCB/EECS-2009-28, 2009.

[10] 미래창조과학부, “K-ICT 클라우드 컴퓨팅 활성화 계획”, 2017.

PDF 악성코드 유형분류를 위한 API 시퀀스 정제 방안 연구

김경한, 최보민, 강홍구, 박준형*

*한국인터넷진흥원

A study on API sequence refining method for categorizing PDF malware

Kyeonghan Kim, Bomin Choi, Hongkoo Kang, Junhyung Park*

*Korea Internet & Security Agency.

요 약

문서형 악성코드의 증가와 더불어 유사/변종 악성코드 공격이 끊이지 않고 있는 가운데, 악성코드 공격에 대한 효율적인 대응을 위하여 머신러닝을 적용한 악성코드 분류기술에 대한 연구가 활발히 이루어지고 있다[1][2][3][4][5]. 그러나, 대부분의 연구는 지도학습에 기반하며, 전문가에 의해 라벨이 생성되지 않은 악성코드를 분류하는 방법에 대한 연구는 미비하다. 본 논문에서는 악성코드의 행위정보를 기반으로 대량의 악성코드를 빠르게 분류 할 수 있는 행위기반 악성코드 프로파일링 시스템을 소개하고, 해당 시스템에서 PDF 악성코드 유형을 분류하기 위해 사용하는 비지도 학습 기반 라벨 생성을 위한 API 시퀀스 정제방안을 제시한다.

I. 서론

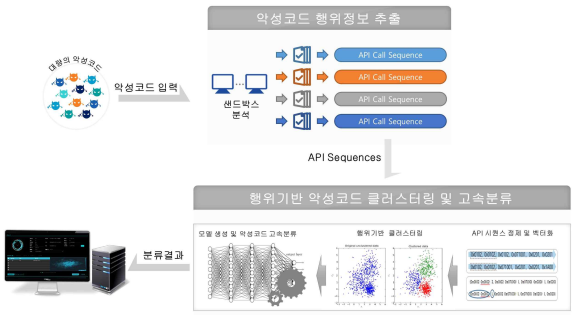
글로벌 보안제품 성능 평가 기관인 AV-TEST의 보고서에 의하면, 악성코드 발생량은 지난15년간 증가해왔고, 2018년 1월부터 10월까지의 발생량은 1억2천만 건에 달한다. 이는 일평균 약40만 건의 위협이 발생하고 있음을 보여준다[6]. 최근 악성코드 공격이 급증하는 가운데, 문서형 악성코드 또한 증가추이를 보이고 있다[7][8].

시간 및 비용측면에서의 공격 효용성을 추구하는 악성코드 제작/공격자들은 주로 기존의 악성코드를 재활용하거나 동일한 제작도구를 활용한다[9]. 이러한 악성코드 공격에 대응하기 위해서 대량의 악성코드를 빠르게 분류하는 기술이 요구되고 있다. 현재 한국인터넷진흥원(이하 KISA)에서는 악성코드의 행위정보를 이용하여 유사/변종 악성코드를 신속하게 분류할 수 있는 행위기반 악성코드 프로파일링 시스템을 개발하고 있다. 본 논문에서는 해당 시스템을 소개하고, 시스템에서 PDF 악성코드를 분류하기 위해 사용하는 비지도 학습기반 API시퀀스 정제 방안에 대하여 소개한다.

II. 행위기반 악성코드 프로파일링 시스템

KISA에서는 유사/변종 악성코드들의 행위가 유사하다는 특징을 고려하여 유사/변종 악성코드를 빠르게 분류하는 행위기반 악성코드 프로파일링 시스템을 개발하고 있다.

행위기반 악성코드 프로파일링 시스템은 샌드박스를 통해 악성코드의 실행중에 호출되는 API시퀀스를 추출하여, 인덱스 해싱등의 전처리 과정을 통하여 행위벡터로 변환한다. 변환된 행위벡터를 대상으로 악성코드간의 유사도를 측정하여 비지도 학습기반 클러스터링을 통해 라벨을 생성하고, 생성된 라벨을 기반으로 뉴럴 네트워크를 학습하여 모델을 생성한다. [그림 1]은 행위기반 악성코드 프로파일링 시스템의 개념도를 나타낸다[10].



[그림 1] 행위기반 악성코드 프로파일링 시스템 개념도

III. PDF 악성코드 API 시퀀스 정제 방안

문서형 악성코드는 바이너리 자체만으로 악성 여부를 구분하기 어려우며, 문서 실행 과정에서 내부 악성 스크립트나 프로그램의 구조적 취약점들을 이용하여 악성행위를 시도한다. 또한, 문서 편집기나 뷰어 프로그램은 문서형 악성코드를 실행하기 위한 필수 요소이다.

위와 같은 이유로 문서형 악성코드를 행위기반으로 분류하기 위해서는 뷰어와 문서 편집기 실행을 통한 API 호출 분석이 필요하다. 즉, 정상 문서와 악성 문서에서 사용하는 API 목록을 비교하여 악성 API 호출을 필터링 함으로써 행위 기반의 악성 문서 분류가 가능하다.

3.1 텍스트마이닝 기법을 통한 PDF 악성코드 호출 API 특징 분석

본 논문에서는 PDF 악성코드 4,156건을 대상으로 호출하는 API 시퀀스 특징을 분석한다. API 시퀀스 특징분석에는 문서 및 문서군내에서의 핵심어를 도출하는 텍스트마이닝 기법인 TF-IDF(Term Frequency-Inverse Document Frequency)기법을 사용한다. TF값이 높을수록 한 문서 안에서 특정 API의 중요도가 높음을 의미하고, IDF값이 낮을수록 문서군 안에서 특정 API의 중요도가 높음을 의미한다. 즉, IDF값이 1일 경우 특정 API를 모든 PDF 악성코드에서 호출했음을 의미한다. [그림 2]는 PDF 악성코드 4,156건에서 호출한 75종의 API에 대하여 TF-IDF값을 계산한 결과이다. 호출된 75종의 API는 파일 제어 함수 24종, 프로세스 제어 함수 16종, 레지스트리 제어 함수 11종, 기타 함수 24종으로 구성된다.

API 명	TF-IDF(평균)	TF-평균	IDF
NtAllocateVirtualMemory	0.705612623	0.705612623	1
NtOpenFile	0.456495082	0.456384831	1.000241575
NtCreateFile	0.215633669	0.215633669	1
pdf_unescape	0.098047777	0.057545446	1.703832081
LdrLoadDll	0.085294997	0.085274397	1.000241575
NtWriteFile	0.070505902	0.065652711	1.073922185
NtDelayExecution	0.067689915	0.048672594	1.390719292
CoInitializeEx	0.039442682	0.039433156	1.000241575
CoUninitialize	0.038695024	0.038685678	1.000241575
CoCreateInstance	0.02535209	0.021489989	1.179716258
pdf_eval	0.015803078	0.007272287	2.173054765
NtProtectVirtualMemory	0.009930115	0.005149295	1.92844155
__exception__	0.006842549	0.001905006	3.591878769
NtResumeThread	0.004083743	0.00377877	1.08070718
CreateProcessInternalW	0.003527869	0.002233451	1.57955973
...	...	...	...
OleInitialize	0.00122022	0.000486235	2.509527002
CreateThread	3.95432E-07	5.12283E-08	7.719013154
GetForegroundWindow	3.95432E-07	5.12283E-08	7.719013154
GetNativeSystemInfo	3.95432E-07	5.12283E-08	7.719013154
GetUsernameW	3.95432E-07	5.12283E-08	7.719013154
GlobalMemoryStatusEx	3.95432E-07	5.12283E-08	7.719013154
LoadStringW	3.95432E-07	5.12283E-08	7.719013154
NtCreateThreadEx	3.95432E-07	5.12283E-08	7.719013154
NtQuerySystemInformation	3.95432E-07	5.12283E-08	7.719013154
OpenSCManagerW	3.95432E-07	5.12283E-08	7.719013154
OpenServiceW	3.95432E-07	5.12283E-08	7.719013154
RegQueryValueExA	3.95432E-07	5.12283E-08	7.719013154
SetFilePointer	3.95432E-07	5.12283E-08	7.719013154
SetFilePointerEx	3.95432E-07	5.12283E-08	7.719013154
StartServiceW	3.95432E-07	5.12283E-08	7.719013154

[그림 2] PDF 악성코드 호출 API 대상 TF-IDF 결과

일반적으로 TF-IDF의 값이 높을수록 문서군 내에서의 핵심어가 될 확률이 높아 [그림 2]에서 상단에 위치한 API들이 PDF 악성코드를 분류하는 핵심어가 되어야하지만, 문서형 악성코드의 실행을 위해서는 뷰어 및 문서 편집기의 실행이 필수라는 점을 감안해야 한다. 즉, 문서형 악성코드는 실제 악성행위를 하는 부분은 취약점의 트리거 혹은 스크립트 부분에 해당하는 극히 일부분이고, 뷰어 및 문서 편집기가 실행되는 과정에서 호출되는 상투적인 API가 대다수이다. 이를 고려하면, [그림 2]에서 상위부분에 해당하는 IDF값이 1에 근사하고 TF값이 높은 API는 뷰어 및 문서 편집기에서 호출하는 상투적인 API라고 간주할 수 있다. 따라서, PDF 악성코드를 분류하기 위해서는 뷰어 및 문서 편집기를 통해 상투적으로 호출되는 API를 정제해야 하며, 행위기반 악성코드 프로파일링 시스템에서는 [그림 2]의 상위 10종의 API를 정제 대상으로 한다.

```
1  System Call Tracer for Windows XP - Windows 10. (Version: 0.9.3.0)
2  Copyright (c): Pankaj Garg <pankaj@intellectualheaven.com>.
3  All rights reserved.
4
5  Tracing command: ["C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe"]
6  ADOBE_READLOGGER_CMD:COMMENT:BeforeOROADOBE_READLOGGER_CMD:COMMENT:AfterOROPathAddBackslashW(1cbf8bc, 0, 76b3b130, 5c005c) = 1cbf962
7  QueueUserWorkItem(767c0964, 4638d0, 0, 4638d0) = 1
8  RegKrnInitialize(75820000, 2, 0, 2f3f9cc) = ffffff01
9  TlsGetValue(13, 2f3f9e0, 6c1622a3, a67f43f3) = 0
10 TlsGetValue(13, 0, 6c162c89, dc68205f) = 0
11 GetModuleHandleA(6c1c49ec, 0, 6c162c89, dc68205f) = 761a0000
12 GetProcAddress(761a0000, 6c1c49fc, 0, 6c162c89) = 7741cd10
13 DecodePointer(dc68205f, 0, 6c162c89, dc68205f) = 761f1e16
14 TlsSetValue(13, 761f1e16, 2f3f9e0, 6c1622a3) = 1
15 TlsGetValue(13, 3, a67f43f3, 2f3fa5c) = 761f1e16
```

[그림 3] straceNT를 이용해 캡처한 Adobe Reader API 함수 호출 시퀀스

```
1  ADOBE_READLOGGER_CMD:COMMENT:BeforeOROADOBE_READLOGGER_CMD:COMMENT:AfterOROGetProcessHeap
2  HeapAlloc
3  InterlockedDecrement
4  GetProcessHeap
5  HeapFree
6  Ord1d7
7  Ord278
8  Ord278
9  Ord278
10 GetCurrentThread
```

[그림 4] 수집한 API 호출 시퀀스 중 중복된 함수명 제거

공문서 실행한 경우	정상 파일 실행한 경우	공통 발견 API	PDF 악성코드 호출 API 75종
	AcquireSRWLockExclusive		
	AcquireSRWLockShared		
ActivateActCtx	ActivateActCtx	ActivateActCtx	
	AddAtomW		
	AnyLinkedFonts		
	AppendMenuW		
	AssocCreate		
BaseDllFreeResourceId	BaseDllFreeResourceId	BaseDllFreeResourceId	
BaseDllMapResourceIdW	BaseDllMapResourceIdW	BaseDllMapResourceIdW	
BitBlt	BitBlt	BitBlt	
CalcMenuBar	CalcMenuBar	CalcMenuBar	
CallWindowProcW	CallNextHookEx	CallWindowProcW	
	CallWindowProcW		
	ChangeWindowMessageFilterEx		
CharNextW	CharNextW	CharNextW	
	CharPrevW		
...	...	...	...
ClientToScreen	ClientToScreen	ClientToScreen	
CloseHandle	CloseHandle	CloseHandle	
CloseThemeData	CloseThemeData	CloseThemeData	
	CoCreateGuid		CoCreateInstanceEx
	CoCreateInstance		CoCreateInstance
	CoInitialize		CoGetClassObject
	CoInitializeEx		CoInitializeEx
	CoLockObjectExternal		CoInitializeSecurity
CombineRgn	CombineRgn	CombineRgn	
	CompareStringA		
CompareStringW	CompareStringW	CompareStringW	
	ConvertDefaultLocale		
CopyRect	CopyRect	CopyRect	
	CoRegisterInitializeSpy		
	CoTaskMemAlloc		
CoTaskMemFree	CoTaskMemFree	CoTaskMemFree	
	CoUninitialize		CoUninitialize
	CreateActCtxW		
CreateBitmap	CreateBitmap	CreateBitmap	
CreateCompatibleBitmap	CreateCompatibleBitmap	CreateCompatibleBitmap	
CreateCompatibleDC	CreateCompatibleDC	CreateCompatibleDC	
CreateDIBitmap	CreateDIBitmap	CreateDIBitmap	CreateDirectoryW
...	...	...	...
CreateDIBSection	CreateDIBSection	CreateDIBSection	CreateProcessInternalW
CreateEventA	CreateEventA	CreateEventA	CreateThread

[그림 5] 공문서, 정상문서, PDF악성코드 대상 호출 API 시퀀스 비교 결과

3.2 PDF 악성코드 정적분석을 통한 정제방안 타당성 검증

3.1에서 제시한 PDF 악성코드가 호출하는 API 정제 방안의 타당성을 검증하기 위하여 악성코드 행위기반 프로파일링 시스템에서 사용하는 뷰어 프로그램을 대상으로 공문서, 정상문서에서 호출하는 API 시퀀스를 분석한다. 뷰어 프로그램의 API 함수 호출 로깅은 [그림 3]과 같이 straceNT를 통하여 캡처하였으며, [그림 4]와 같이 수집한 API 호출 시퀀스 중 중복을 제거한다. 수집한 API 호출 시퀀스는 [그림 5]와 같다. [그림 5]의 녹색은 공문서와 정상 문서에서 상투적으로 호출하는 API를 의미하고, 황색은 정상문서와 악성코드에서 모두 호출한 API, 암적색은 악성코드에서만 호출한 API를 의미한다. [그림 2]의 API 정제 대상인 상위10종의 API중 상위 7종의 API는 녹색으로 표기되며, 나머지 3종의 API는 CoInitializeEx, CoUninitialize, CoCreateInstance로 황색으로 표기되나 Adobe Reader에서 사용하는 객체, 스크립트 관련함수이므로 정제대상에 포함시키는 것이 타당하다.

[표 1] API 정제 전 후 생성된 클러스터 신뢰도

구분	API 정제 전	API 정제 후
클러스터수	385개	376개
미분류 개수	14개	5개
클러스터 신뢰도(평균)	98.6%	99.8%

IV. 결론

본 장에서는 3장에서 제시한 API정제 방안을 행위기반 프로파일링 시스템에 적용시켜 PDF 악성코드를 분류한 결과를 나타낸다. [표 1]은 PDF 악성코드 공통호출 API 정제 전후 비지도 학습기반 클러스터링을 통하여 생성된 라벨(클러스터)의 신뢰도를 보여준다. API정제 결과 생성된 라벨의 개수는 385개에서 376개로 감소하고 신뢰도는 약 1.2%증가한 것으로 보아 클러스터간의 응집도가 향상되었음을 알 수 있다. 또한, 미분류 악성코드 개수 또한 14개에서 5개로 감소하여 뷰어 및 문서 편집기에서 공통으로 호출하는 통상적인 API를 정제한 것이 효용성이 있음을 보였다.

ACKNOWLEDGMENT

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2016-0-00081, 악성코드 全 생명주기 통합 프로파일링 및 공격그룹 식별 기술 개발)

[참고문헌]

[1] P. Li, L. Liu, et al., On challenges in evaluating malware clustering. In Proceedings of 13th International Symposium on Recent Advances in Intrusion Detection, RAID 2010, pp. 238-255, 2010.

[2] C. Ravi and R. Manoharan, "Malware detection using windows api sequence and machine learning," International Journal of Computer Applications, vol. 43, Issue 17, pp. 12-16, April 2012.

[3] A. Fujino, et al., "Diccovering Similar Malware Samples Using API Call Topics," 2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC2015, January 2015.

[4] I. K. Cho, et al., "Malware Analysis and Classification Using Seqyence Alignments," Intelligent Automation & Soft Computing, vol.22, Issue 3, pp. 371-377, 2016.

[5] L. Liu, et al., "Automatic malware classification and new malware detection using machine learning," Frontiers of Information Technology & Electronic Engineering, vol. 18, Issue 9, pp. 1336~1347, September 2017.

[6] AV-TEST, https://www.av-test.org.

[7] SOPHOS, "The Rise of Document-based Malware," (https://www.sophos.com/en-us/security-news-trends/the-rise-of-document-based-malware.aspx).

[8] AhnLab ASEC Report, https://www.ahnlab.com/kr/site/securityinfo/asec/asecView.do?groupCode=VNI001&seq=27109.

[9] HELPNETSRCURITY, "Why hackers reuse malware," (https://www.helpnetsecurity.com/2017/11/20/hackers-reuse-malware/)

[10] H. Kang, et al., "Design and Implementation of Profiling System based on Malware Behavior," ICONI, pp.395-400, 2017.

탈중앙화 항공 체크인 시스템을 위한 스마트 계약

김혜빈*, 박지선*, 신상욱**

*부경대학교 대학원 정보보호학협동과정

**부경대학교 IT융합응용공학과

Smart Contract for Decentralized Airport Check-In System

Hye-Bin Kim*, Ji-Sun Park*, Sang Uk Shin**

*Interdisciplinary Program of Information Security, Graduated School, Pukyong National University

**Dept. of IT Convergence and Application Eng., Pukyong National University

요 약

현재 공항에서 이루어지는 체크인 시스템은 서버-클라이언트(Server-Client) 구조로 동작하여 체크인 요청에 대한 응답 처리를 한다. 해당 구조는 중앙 서버가 단일실패지점이 되는 취약점이 존재하기 때문에 전체 시스템 중단으로 인한 체크인 서비스 불가능에 대한 위험이 항상 존재한다. 또한 승객 예약 정보를 작성하는 데 있어서도 데이터 오류와 위·변조의 위험이 존재한다.

따라서 공항 체크인 시스템에 존재하는 단일 실패지점에 대한 위협을 줄이고, 승객의 항공권 예약 정보 기록 과정의 자동화를 통한 데이터 정확성을 확보하기 위한 시스템이 필요하다. 본 논문에서는 대표적인 탈중앙화 시스템인 블록체인을 기반으로 하는 공항 체크인 모델과, 해당 모델이 사용할 스마트 계약 내용에 대하여 제안한다.

I. 서론

현재 공항에서 이루어지는 체크인 시스템은 웹 페이지로부터 요청이 들어오면 항공사 중앙 서버와 연결된 컴퓨터 예약 시스템(CRS)에 저장된 승객 예약정보인 PNR(Passenger Name Record)의 내용을 검증하여 결과를 웹 서버가 응답하는 방식으로 이루어진다[1]. 시스템은 서버-클라이언트(Server-Client) 구조로 동작함으로써 중앙 서버가 단일실패지점(Single Point of Failure)이 되는 취약점을 갖고 있다. 따라서 중앙 서버가 마비되면 그를 기반으로 하는 서비스들이 적시에 제공되지 못한다. 체크인 서비스도 이런 경우에 해당됨에 따라, 위와 같은 문제가 발생하면 탑승 예약자들의 불편을 초래할 수밖에 없다.

또한 예약정보를 포함한 PNR은 프로그램을 이용하여 담당자가 직접 작성하는 것이다[2]. 만일 작성과정에서 오류가 생긴다면 데이터 정확성을 확보하지 못하며, 악의적인 담당자에 의한 데이터 위·변조 문제도 충분히 가능성 있다. 따라서 이러한 문제점을 보완하기 위해 단일 실패지점을 없앤 탈중앙화 구조를 채택함과 동시에 저장된 데이터들의 정확성과 무결성을 보장할 수 있는 모델이 필요하다. 본 논문에서는 대표적인 탈중앙화 시스템인 블록체인을 기반으로 하는 체크인 모델의 구성요소와 예약된 항공권에 대한 PNR을 설정하여 블록체인에 저장하거나 체크인 요청에 대한 응답 처리 기능을 이행하는 스마트 계약의 정의 방법에 대하여 제안한다.

II. 관련연구 : 스마트 계약

블록체인(Blockchain)은 수많은 트랜잭션들을 모아 하나의 블록으로 형성하여 타임스탬프 값과 함께 체인 구조로 연결한 탈중앙화 데이터베이스이다[3]. 블록의 헤더에 이전 블록의 해시값이 포함이 되어있기 때문에 블록 내의 필드

값이 1비트라도 변경되면, 연결된 모든 블록의 해시 값이 변경된다. 또한 별도의 중앙 서버가 없이 P2P 네트워크에 참여하는 분산된 노드들이 합의 알고리즘을 기반으로 하여 동일한 장부를 공유하고 있기 때문에 변경의 결과를 즉시 열람할 수 있으므로 블록의 위·변조가 어렵다. 그리고 한번 체인에 추가되면 위의 특징들로 인해 내용이 보존된다. 따라서 높은 안전성, 불변성, 무결성을 보장한다.

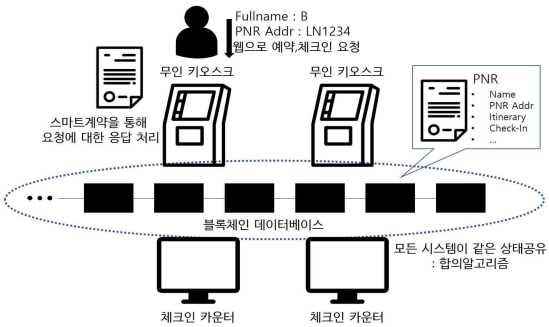
이더리움 스마트 계약(Ethereum Smart Contract)은 블록체인을 기반으로 하는 금융거래, 부동산 계약, 공증 등 다양한 형태의 계약을 체결하고 이행하는 것으로, 2013년 Vitalik Buterin에 의해 널리 알려지게 되었다. 개발자가 직접 계약의 내용과 조건을 코딩할 수 있는 것이 특징이며 그에 따라 다양한 스마트 계약이 지금까지 개발되어왔다. 이더리움에서는 튜링 완전한 언어인 솔리디티(Solidity)를 이용하여 계약을 구축할 수 있다.

또한 블록체인 기반 탈중앙화 어플리케이션 DApp(Decentralized Application)을 이더리움 플랫폼 상에 설계하여 기존의 여러 중앙 집중형 시스템을 탈중앙화 구조로 대체할 수 있다[4]. 이는 웹, 모바일 어플리케이션 등 다양한 형태로 구현될 수 있다.

III. 스마트 계약을 이용한 제안 모델

3.1 시스템 구성요소

본 절에서는 스마트 계약을 이용한 블록체인 기반 공항 예약, 발권 시스템의 구성요소에 대해 논한다. [그림 1]은 제안하는 시스템 구성요소 간에 일어나는 전체 프로세스를 나타낸 것이다.



[그림 1] 제안 시스템의 구성요소

블록 합의 과정에 참여하는 노드들로는 항공사별로 설치된 카운터 PC들과 공항의 셀프 체크인 키오스크들이 해당 된다. 그리고 각 PC에 구동되는 웹 어플리케이션이 바로 DApp이다. 이들은 항공사에서 정한 규칙을 바탕으로 정의된 스마트 계약에 따라 생성되는 트랜잭션을 검증한 다음 합의 알고리즘을 통해 블록에 포함시킨다. 결과적으로 모든 노드는 동일한 내용의 예약 데이터들이 저장된 장부를 공유한다.

항공권에 대한 예약정보의 기록이나 공항 체크인 요청에 대한 응답은 스마트 계약 코드가 DApp에 참조되어 실행됨으로써 이루어진다. 계약의 내용에는 입력받은 데이터들을 바탕으로 PNR을 설정하는 함수와 체크인 요청에 대하여 검증하는 함수, 그리고 승객 예약정보가 저장될 PNR 데이터 구조체가 포함된다. 계약을 구성하는 함수에 대한 자세한 내용은 다음 3.2절에서 논의한다.

3.2 스마트 계약을 이용한 동작 프로세스

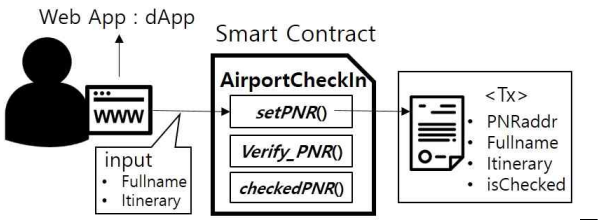
본 절에서는 스마트 계약 내의 체크인 기능과 관련된 함수들의 내용에 대해 논한다. 동작 프로세스를 알아보기 위한 계약의 개발환경은 이더리움 리믹스 솔리디티(Ethereum Remix-Solidity)이다. 이더리움은 기본적으로 공개 블록체인(Public Blockchain)을 채택하기 때문에, 누구나 스마트 계약에 접근할 수 있다. 이에 무분별한 접근을 방지하기 위해 함수의 접근 권한을 다르게 하여 스마트 계약을 호출한 노드의 역할을 일정 부분 제어할 수 있다.

스마트 계약 내에 정의된 함수의 역할은 크게 두 가지로 나뉜다. 첫 번째로, 항공권을 구매할 때 승객 예약정보 PNR을 생성하여 기록하는 함수와 보조적 함수인 승객 예약정보 식별번호(PNR address) 구성 함수가 있다. 두 번째로는 앞서 논의한 승객 예약 정보 기록 함수에서 생성된 PNR을 바탕으로 체크인 요청이 들어왔을 때 내용을 검증하여 결과를 응답하는 함수가 있다.

```
struct PNR{ //passenger name record
    string PNRaddr;
    string Fullname;
    bytes32 Itinerary;
    bool isChecked;
    uint256 Deadline;
}
mapping(string=>PNR) PNRlist;
```

[그림 2] PNR 내용을 저장하는 구조체

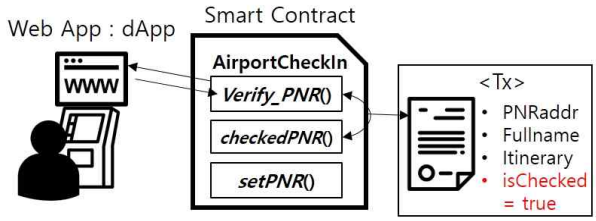
기존 시스템에서와 같이 사용자가 항공권을 구매하면, 항공권의 예약정보가 PNR 데이터로 생성이 된다. [그림 2]는 예약 정보가 저장될 구조체 필드 *PNR*에 관한 것이다. *PNR*에는 항공권을 예약한 사람의 영문이름인 *Fullname*, 여정의 내용을 해시함수로 처리한 *Itinerary*, 해당 PNR에 해당하는 항공권의 발권 여부를 나타내는 *isChecked*, 항공권의 유효기간을 의미하는 *Deadline*필드로 이루어져있다. *PNRaddr*필드와 *isChecked*필드는 PNR의 유효성을 검증할 때 사용된다. *PNRlist*는 string형태의 *PNRaddr*값이 key가 되는 *PNR* 구조체 배열이다. 배열엔 모든 PNR 데이터가 저장된다.




```
function setPNR(string _fullname, bytes32 _itinerary){
    _pnraddr = generatePNRNumber();
    PNRlist[_pnraddr].PNRaddr = _pnraddr;
    PNRlist[_pnraddr].Fullname = _fullname;
    PNRlist[_pnraddr].Itinerary = _itinerary;
    PNRlist[_pnraddr].isChecked = false;
    PNRlist[_pnraddr].Deadline =  calcDeadLine();
    generatePNR(_pnraddr,_fullname);
}
```

[그림 3] PNR 생성 과정과 계약코드

[그림 3]은 PNR을 생성하는 함수에 대한 코드이다. 항공권 구매완료 시에, 우선 *generatePNRNubmer*함수를 호출한다. 이는 기존의 시스템에서 승객 예약 번호를 생성할 때의 규칙이 정의된 함수이다. 함수를 통해 탑승 예약자가 입력한 데이터를 바탕으로 영문과 숫자 조합의 6자리 일련번호를 생성하여 *_pnraddr* 값에 저장한다. 그 다음 입력받은 영문이름, 항공권에 대한 여정의 해시 값을 각각의 구조체 필드로 설정하여, 예약 정보 트랜잭션을 생성한다. *setPNR*함수가 호출되고 난 후 트랜잭션이 블록에 포함되었을 때, *isChecked* 필드가 갖는 기본 값은 *false*이다. 이는 승객 예약정보에 해당하는 항공권이 아직 발권이 되지 않았음을 의미한다.




```
function Verify_PNR(string _pnraddr)returns(bool){
    if(stringCompare(PNRlist[_pnraddr].PNRaddr,_pnraddr) == false)
        return false;
    else{
      if(Verify_expiriation(PNRlist[_pnraddr].Deadline) != true)
        return false;
    }
    return true;
}
```

[그림 4] 체크인 요청 검증과정과 계약코드

[그림 4]는 체크인에 관여하는 노드들 즉, 키오스크나 카운터 PC가 웹 체크인 페이지로부터 체크인 요청을 받았을 때 사용하는 *Verify_PNR* 함수 코드이다. *PNRlist* 배열 내에 입력받은 예약번호 *_pnraddr*와 그를 key값으로 하는 value 원소인 *PNRlist[_pnraddr].PNRaddr* 값이 서로 일치하면 승객 예약정보의 존재 유무가 확인이 된다.

또한 함수 *Verify_expiration*을 통하여 승객 예약 정보가 가진 유효기간에 대한 검사도 해야 한다. *PNR* 구조체의 *Deadline*필드는 임의로 항공기 출발 시각에 24시간을 더한 시각으로 계산이 되도록 함수를 따로 두어 설정을 해두었다. 해당 승객 *_pnraddr*에 해당하는 *PNR*의 *Deadline*필드가 함수 호출시점을 기준으로 유효기간 내에 있다면 *true*를, 그렇지 않으면 *false*를 반환 받음으로써 확인할 수 있다. 위의 두 함수로 예약 정보에 대한 검증이 *true*를 반환받음으로써 완료되면, *checkedPNR*함수로 좌석지정 또는 발권 기능을 수행한 후 체크인 완료 트랜잭션을 생성한다.

IV.분석

기존 체크인 시스템에서 사용하던 승객 예약 정보인 PNR은 프로그램을 이용하여 직접 작성을 했다. 사람이 하는 작업임에 따라 오류가 생기거나 의도적인 위·변조 문제가 있을 수 있다. 이러한 작성 과정을 스마트 계약의 코드로 구현해두면, 동작 조건을 충족할 때 입력받은 데이터들을 바탕으로 자동화되어 작성이 된다. 따라서 데이터의 정확성을 높이고, 임의적인 위·변조를 방지한다. 또한 프로그램을 사용하는 데 있어서 사용되던 시스템 유지비도 줄일 수 있다.

또한 네트워크에 참여하는 즉 항공사 카운터 PC와 셀프 체크인 키오스크들이 노드가 되어 동일한 예약 데이터들에 대한 장부를 공유하고 있다. 기존에 항공사 중앙서버나 공항의 전산망이 마비가 될 때, 웹 체크인을 하는데 어려움이 있었다면 제안 모델에서는 참여 노드중 일부가 마비되더라도, 남은 노드들만으로 전체 공항 내의 시스템 유지가 가능하기 때문에, 체크인 서비스가 불가능할 확률이 줄어든다.

또한 예약 데이터가 스마트 계약을 통하여 생성된 후 블록체인에 저장되기 때문에, 블록체인의 특성상 데이터를 임의로 위·변조 할 수 없다. 또한 스마트 계약을 통해 자동화된 프로세스를 수행함으로써, 항공권 예약과 발권에 대한 부인 방지 기능도 추가된다.

제안 모델이 채택할 수 있는 블록체인의 형태는 공항을 사용하는 항공사들이 참여하는 컨소시엄형 블록체인(Consortium Blockchain)이다. 일반적인 공개 블록체인이 누구에게나 열람, 합의 과정 참여권한이 허가되는 반면, 사설 블록체인(Private Blockchain)과 위에 서술한 컨소시엄형 블록체인은 네트워크 내에서 미리 선정된 노드들에게만 그 권한이 부여된다[5]. 따라서 항공사 내의 선정된 노드들만이 승객의 예약 데이터를 다루기 때문에, 누구에게나 무분별한 공개되는 것을 방지하여 보안성을 높일 수 있다. 또한 정해진 몇 개의 노드들만이 합의 과정에 참여함으로써 합의 속도를 높인다. 이는 실제 체크인 시스템이 가져야할 실시간성과 부합한다.

V. 결론

본 논문에서는 탈중앙화 구조로 승객 예약 데이터 생성과, 그를 이용한 체크인 시스템 과정을 나타낸 스마트 계약에 대해 논하였다. 블록체인을 기반으로 한 탈중앙화 구조를 전체 시스템에 도입함으로써, 단일 실패지점에 대한 위협을 낮추고 스마트 계약을 통하여 체크인 데이터가 검증, 생성되어 블록체인에 등록이 됨으로써, 악의적인 위·변조로부터 안전하다는 장점이 있다. 또한 예약, 발권을 자동화함으로써 정확성, 신뢰도를 높일 수 있다. 따라서 제안된 본 모델은 논의된 장점들을 특징으로 하여 기존의 시스템을 보완 할 수 있을 것이다.

Acknowledgement

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임. (No.2017-0-00156, 블록체인의 정보보호 프레임워크 및 평가 방법 개발)

[참고문헌]

[1] Mehovic, Farid. "System for propagating, retrieving and using transaction processing facility airline computerized reservation system data on a relational database processing platform." U.S. Patent No. 6,122,642. 19 Sep. 2000.

[2] Wikipedia, Passenger Name Record, Retrieved from URL: https://en.wikipedia.org/wiki/Passenger_name_record

[3] Romano, Diego, and Giovanni Schmid. "Beyond Bitcoin: A Critical Look at Blockchain-Based Systems." Cryptography 1.2 (2017): 15.

[4] Buterin, Vitalik. "A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum White Paper." 2014.

[5] Zheng, Zibin, et al. "An overview of blockchain technology: Architecture, consensus, and future trends." Big Data (BigData Congress), 2017 IEEE International Congress on. IEEE, 2017.

자바스크립트 난독화 도구 분석 및 평가*

신동순, 김형식
성균관대학교 전자전기컴퓨터공학과

On the evaluation of JavaScript obfuscation tools

Dongsoon Shin, Hyoungshick Kim
Department of Electrical and Computer Engineering, Sungkyunkwan University.

요 약

웹 브라우저에서 사용되는 자바스크립트는 쉽게 코드를 확인 할 수 있다. 악의적인 사용자는 웹 브라우저를 통해서 해당 웹 페이지의 자바스크립트를 분석하여 XSS(Cross Site Scripting)와 같은 공격을 시도 할 수 있다. 이런 보안 문제를 해결하기 위해서 소스 코드를 난독화 하여 소스 코드를 보호한다. 자바스크립트 난독화 도구들이 많이 생김으로써 어떤 난독화 방법을 사용해야 하는지 가장 효율적으로 난독화 할 수 있는 연구는 많이 진행 되었다. 하지만 난독화 도구에 대한 평가는 연구가 많이 되지 않았고 아직까지 문제로 남아 있다. 본 논문에서는 온라인에서 사용되는 자바스크립트 난독화 도구 9개를 선택하여 분석하고 난독화 도구의 크기 증가량을 측정 하였다. 결과로는 JSfuck 난독화 도구가 기존 코드에 비해서 54배가 증가 하여 크기 증가량이 가장 높은 난독화 도구라는 것을 밝혔다.

I. 서론

자바스크립트는 웹에서 사용되는 스크립트 언어로써 PHP (Hypertext Preprocessor), HTML (Hyper Text Markup Language)과 함께 사용되어 일반적인 웹 페이지를 구성한다. 웹 페이지에서 사용된 스크립트 및 소스 코드는 웹 브라우저를 통해서 컴파일 되어 실행된다. 이때 웹 브라우저 악의 적인 사용자는 해당 웹페이지의 소스 코드를 보고 공격을 할 수 있다. 웹 개발자들은 자신들이 개발한 코드를 보호하기 위하여 난독화를 사용하여 자신의 코드를 보호하였다.

초기의 단순한 난독화 방식은 공백을 모두 지워 버리는 방법이 소개 되었고 이를 파훼하기 위하여 공격자들은 다시 공백을 복구하는 툴을 개발하였다. 이렇게 공격자와 방어자의 싸움으로 현재의 난독화 방법은 처음보다는 굉장히 강력해졌다. 강력해 짐과 동시에 여러 난독화툴이 나오기 시작하여 어떤 난독화툴을 사용해야 나의 코드를 더 효율적으로 보호 할 수 있는지가 필요 하였다. 여러 난독화 툴은 각각의 장점과 단점을 갖고 있지만 아직까지 제대로 된 평가 방법은 갖추어져 있지 않다.

본 논문은 자바스크립트 난독화 도구를 평가하기 위하여 9개의 자바스크립트 난독화 도구를 분석 하였고 평가 하였고 평가 결과 JSfuck이 크키 증가량이 58배로 가장 컸다.

본 논문의 구성은 2장에서 관련 연구를 설명하고 3장과 4장에 걸쳐 자바스크립트 난독화 도구를 분석하고 평가한다. 5장에서는 난독화 크기 증가량의 평가 결과를 설명하고 6장에서 향후 계획을 설명하며 7장에 결론을 설명한다.

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음(IITP-2018-2015-0-00403)

[표 1] 자바스크립트 난독화 툴 분석 결과

Tool name/Feature	Parameter	Function	Value	String	Flow	Signature
Obfuscator.io	o	o	o	x	o	x
JavaScript2img	o	o	o	o	o	x
BeautifyTools	x	x	x	x	o	o
JavaScript Obfuscator	o	x	x	o	o	x
Aaencode	o	o	o	o	o	x
JSfuck	o	o	o	o	o	x
Esoteric JavaScript Obfuscator	o	o	o	o	o	o
Dean Edwards Packer	x	x	x	x	o	o
UglifyJS	o	x	o	x	x	x

II.관련 연구

기존의 난독화 방법의 연구는 지속적으로 이루어 지고 있다. 하지만 난독화를 평가하는 방법에 대해서는 많은 연구가 이루어지지 않았다.

[1]은 자바스크립트 소스 코드의 의미를 이용하여 난독화된 자바스크립트 소스 코드를 탐지한다.

본 논문은 [2]을 이용하여 자바스크립트 난독화 도구에 적용하여 도구를 평가 하였다. [1]에서 난독화 도구를 평가하기 위하여 난독화 방법에 따라서 분류하고 평가한다. 정확성, 시간, 효율성 등으로 난독화 도구를 평가한다.

III.자바스크립트 난독화 도구 분석

[표 1]은 자바스크립트 난독화 도구 및 특징을 보여준다. 난독화 도구의 특징은 Parameter, Fucntion, Value, String, Flow, Signature로 분류 하였다.

Parameter는 함수에 사용된 Parameter가 난독화가 되었는지를 나타낸다. 예를 들어 foo(int number) 라면 number를 다른 문자를 사용하여 난독화를 진행한다.

Function은 함수명이 난독화가 되었는지를 나타낸다. 예를 들어, foo 라는 함수명이 존재할 때 난독화를 한다면 다른 문자 또는 특수문자를 사용하여 난독화 한다.

Value는 코드에서 사용된 변수명들을 난독화를 하였는지를 나타낸다. 변수명 역시 다른 문자열 또는 특수문자로 난독화를 진행한다.

String은 코드에 존재하는 모든 문자열을 난독화를 했는지 나타낸다. 예를 들어 str = "abc" 라면 abc를 다른 문자열로 난독화 시키지만 출력은 abc 그대로 출력이 되는지 나타낸다.

Flow는 코드의 전체적인 흐름이 난독화 되는지를 확인한다. 코드를 확인 하였을 때 함수 또는 변수가 어떠한 방법으로 사용되는지 알 수 있다면 Flow는 난독화 되어 있지 않다.

Signature는 코드가 난독화 된 후 해당 난독화 도구의 Signature를 남기는지 남기지 않는지 판단한다.

obfuscator.io[3는 함수의 인자와 어떤 함수가 사용 되었는지 보기 어렵게 만들어준다. 하지만 문자열들과 함수명을 난독화 시켜주지는 않는다. 기존의 코드에서 새로운 변수와 함수가 생성되고 여러 가지 연산들이 추가 된다.

javaScript2img[4]는 함수와 모든 인자들을 완전하게 재배치하고 문자열도 다른 형태로 만들어준다. 또한 모든 코드들은 자바스크립트의 이미지 함수로 대체 된다.

BeautifyTools[5]는 obfuscator.io와 비슷하게 함수의 인자와 어떤 함수가 사용되었는지 난독화를 시켜준다. 하지만 처음 함수의 인자값이 BeautifyTools의 시그니처인 p,a,c,k,e,d 로 암호화 되기 때문에 BeautifyTools로 난독화를 했다는 사실을 쉽게 알 수 있다.

JavaScript Obfuscator[6]는 함수의 인자와 함수의 흐름을 난독화 한다. 다른 난독화 방법과는 다르게 모든 문자열을 헥사 값으로 나타낸다. 함수 명은 난독화 하지는 않지만 함수에 사용된 문자열은 헥사값으로 나타내기 때문에 어떤 문자열을 사용했는지 알수 없다.

aaencode[7]는 일본식 이모티콘 형태로 모든

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4  <script type="text/javascript">
5      document.write("Hello, World! This is my
6  first JavaScript.");
7  </script>
8  </body>
9  </html>
```

[그림 1] 원본 코드

코드와 문자열들을 특수기호로 만들어 코드가 어떻게 동작되는지도 모두 난독화되어 전혀 알 수 없다.

JSfuck[8]은 aaencode와 비슷하게 모든 코드와 문자열들을 특수문자로 난독화를 진행한다. 총 6개의 특수문자인 [,], (,), !, +를 사용하여 모든 소스 코드와 문자열들을 난독화 한다.

Esoteric JavaScript Obfuscator[9]는 JSfuck과 비슷한 형태로 난독화를 진행한다. 알파벳 a-z와 공백에 해당하는 문자를 특수문자인 (,), [,], >, <, ^, |, %, &, ., !, \, /, =, -, +, ~과 숫자 1을 조합하여 나타낸다. 역시 모든 코드와 문자열을 난독화 하기 때문에 기존의 코드를 알아내기는 아주 힘들다.

Dean Edwards Packer[10]는 BeautifyTools와 비슷한 난독화를 진행한다. 함수의 이름과 파라미터 코드의 흐름을 난독화 하지만 문자열을 난독화 하지는 않는다. 또한 앞부분의 시그니처인 p,a,c,k,e,r 라는 시그니처가 남기 때문에 어떤 난독화 도구를 사용했는지 알 수 있다.

UglifyJS[11]는 Obfuscator.io와 비슷하게 난독화를 진행한다. 함수명, 코드 흐름, 변수명은 난독화를 진행하지만 문자열은 난독화를 진행하지 않는다.

자바스크립트 난독화 도구 분석 결과로는 JavaScript2img, Aaencode, JSfuck 난독화 도구가 가장 많은 부분을 난독화 시키며 Signature를 남기지 않는다. 그러므로 앞서 언급한 난독화 도구의 강도(Strength)가 훨씬 높다.

IV.자바스크립트 난독화 도구 평가

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4  <script type="text/javascript">
5  ...
6  [][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])
7  [!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]
8  +(!![]+[])[+!
9  ...
10 </script>
11 </body>
12 </html>
```

[그림 2] JSfuck으로 난독화된 코드

자바스크립트 난독화 툴이 얼마나 효율적인지 판단하는 방법은 여러 가지가 존재한다. 세 가지 방법으로, 1) 난독화 된 코드가 공격자에 의해서 분석 될 때 공격자의 분석을 얼마나 어렵게 하는지, 2) 공격자가 난독화된 코드를 분석할 때 걸리는 시간이 얼마나 걸리는지, 3) 공격자가 난독화된 코드를 얼마나 정확하게 원본으로 복구 할 수 있는지가 존재한다. 본 논문에서는 2)에서 사용되는 시간의 효율성에 대해서 측정하였다. 시간의 효율성은 소스 코드의 크기에 비례하여 증가하기 때문에 난독화 도구의 크기 증가량을 측정한다.

크기 증가량은 *Size*로 나타내며 기존의 코드의 크기를 *Ori(x)*, 난독화 되어 바뀐 코드의 크기를 *Obf(x)*라고 한다. (1)은 기존의 코드 크기를 분모로 갖고 난독화 되어 바뀐 코드의 크기를 분자로 나타내어 난독화 되어 바뀐 코드의 크기가 기존의 코드 크기에서 얼마나 더 많이 커졌는지 알 수 있다.

$$Size = \frac{Obf(x)}{Ori(x)} \quad (1)$$

(1)을 이용하여 [표 1]의 난독화 도구 들을 측정하였다. 측정 환경은 i5-6600 CPU, RAM 16GB, SSD 512GB 이며, 사용된 OS는 Windows 10 Professional에서 측정 하였다.

기존의 소스 코드는 [그림 1]을 사용 했으며 JSfuck으로 난독화된 소스 코드의 예는 [그림

2]와 같다.

앞서 분석한 자바스크립트 난독화 도구 중 Signature를 남기지 않으면서 모든 Feature를 난독화 하는 도구는 JavaScript2img, Aaencode, JSfuck 이였으므로 해당 도구들의 크기 증가량을 비교했다. [표 2]는 3개의 도구에 대해서 (1)을 적용하여 나타낸 점수로써 기존 코드보다 얼마나 더 많이 크기가 증가 했는지 알 수 있다.

V.평과 결과

본 논문에서 정의한 가장 강력한 자바스크립트 [그림 1]의 소스 코드는 1KB 으로 측정 되었고 난독화 도구는 JavaScript2img, Aaencode, JSfuck 였고 JavaScript2img는 기존 코드에서 8배, Aaencode는 6배, JSfuck은 54배가 증가하여 JSfuck이 가장 많이 증가하였다. 다시 말해, JSfuck으로 난독화 되었을 때 분석에 가장 오래 걸린다.

[표 2] 자바스크립트 난독화 툴 평가 결과

Tool name	Size increase
Origin	1 KB
JavaScript2img	8 KB
Aaencode	6 KB
JSfuck	54 KB

VI.향후 계획

다음 연구로는 자바스크립트의 난독화 도구를 평가하는 지표중 하나인 코드 크기 증가량이 난독화 성능에 얼마나 영향을 미치는지 확인이 필요하다. 또한 실제 난독화 도구의 성능을 평가하기 위하여 난독화된 코드가 자동화된 복호화 도구에 의해서 얼만큼 강한지 등을 정형화된 표로 만들어 어떤 난독화 도구가 가장 효율적인지를 판단이 필요하다.

VII. 결론

본 논문에서는 온라인에서 쉽게 구할 수 있는 자바스크립트 난독화 도구에 대해서 분석하고 난독화 도구로 난독화된 코드와 원본 코드의 크기를 비교하여 크기 증가량을 측정 하였다. 측정된 결과에 따르면 JSfuck이 기존 코드보다 54배 증가하여 가장 크기 증가량이 높은 자바스크립트 난독화 도구다.

[참고문헌]

[1] Lu, Gen, and Saumya Debray. "Automatic simplification of obfuscated JavaScript code: A semantics-based approach." 2012 IEEE Sixth International Conference on Software Security and Reliability. IEEE, 2012.

[2] Viticchié, Alessio, et al. "Assessment of source code obfuscation techniques." Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on. IEEE, 2016.

[3] Javascript Obfuscator Tool https://obfuscator.io/

[4] Javascript2img tool http://javascript2img.com/

[5] Beautiful tools Javascript obfuscator http://beautifytools.com/javascript-obfuscator.php

[6] Javascript Obfuscator http://javascriptobfuscator.com/

[7] Aaencode Demo http://utf-8.jp/public/aaencode.html

[8] JSFuck http://www.jsfuck.com/

[9] Esoteric Javascript Obfuscator https://bl.ocks.org/jasonsperske/5400283

[10] Dean Edwards Packer http://dean.edwards.name/packer/

[11] Uglify-js https://www.npmjs.com/package/uglify-js

LDA를 이용한 사회공학 공격 산출방안

임승호* 김지원*

*육군사관학교 컴퓨터과학과

tmdgh5503@kma.ac.kr admin486@kma.ac.kr

A Study on the Estimation of Social Engineering Attacks Using Latent Dirichlet Allocation(LDA)

Seing-ho Lim* Jee-won Kim*

*Department of Computer Science, Korea Military Academic.

요 약

네트워크상의 기술적인 취약점을 이용한 공격 이외에도 사람들 간의 기본 신뢰를 기반으로 정보를 얻는 비(非)기술적 침입 수단인 사회공학 공격이 증가하고 있다. 사회공학 공격은 비교적 적은 노력으로 광범위한 정보를 얻을 수 있다는 점에서 공격자에게 효율적인 공격수단으로 점차 확대되고 있다. 그럼에도 불구하고 기존 연구는 공격 유형에 대한 산출에 대한 연구가 부족한 실정이다. 이에 본 연구에서는 사회공학 공격기법을 체계적으로 모델링 한 기존 연구들을 LDA(Latent Dirichlet allocation)를 이용하여 사례를 분석하는 방안을 제시하여 기존 모델의 활용성을 높이고자 한다.

I. 서론

과거 2011년 북한 김정일 국방위원장의 사망 소식이 국내·외적으로 큰 관심사로 떠올랐다. 사람들은 김정일 사망 및 시신과 관련된 링크를 클릭하고 동영상을 재생했지만 내용은 허위였고 악성파일만이 전파되었다. 2016년 모기업에서는 개인정보 담당자의 가족을 사칭하여 악성코드가 첨부된 이메일을 보내 회사고객 1,000만 여명의 정보가 탈취당하였다. 이처럼 사람을 속여 자신들의 원하는 방향으로 움직이게 하거나[1] 사람을 속여 민감한 정보를 누설하게 하거나, 적으로 하여금 정보와 연관된 자원의 보안경계(security perimeter)를 우회할 수 있도록 행동함으로써 공격 목표를 완성할 수 있도록 하는 공격을[2] 사회공학 공격이라고 한다.

사회공학 공격은 물리적인 보안 외에도 인간적인 보안이 중요하다는 것을 시사하며 그 공격 발생 빈도가 점차 증가하고 있다. 그럼에도 불구하고 아직 이에 대한 연구는 활발히 이루어지지 않고 있다가 최근 연구에서 심리적 영역을 중심으로 사회공학 공격을 접근하는 연구가 이루어지고 있다[3][4]. 또한, 사회공학 공격 방법을 모델링하여 제시하는 연구[5]에 의하면 모델링 후 검증 방법이 사람이 직접 분석한 사례를 기반으로 검증방법을 제시하였다면 본 연구에서는 데이터 전처리를 통한 토픽 모델링 분석방법을 제안하여 기존 연구의 모델링의 활용성을 높이고자 한다.

II. 공격의 단계

일반적으로 공격자는 일련의 단계를 거쳐 공격하는 패턴을 보이는데 반드시 이러한 과정을 거치는 것은 아니나 대부분 아래와 같은 일련의 단계를 거쳐 공격 한다.

2.1 기술적 공격단계

Fig 1.과 같이 기술적 공격 단계가 이루어지

며, 첫 번째 단계는 정보수집 단계로 목표 시스템의 다양한 정보를 수집하는 단계이며, 두 번째 단계인 목표접근 단계는 해킹 등을 통해 대상 시스템에 접근할 수 있는 수단과 방법을 찾아 불법접근 시도하는 단계이다. 사이버 위협도구들을 이용해 실질적인 피해를 가하는 세 번째 단계가 위협실시 단계이며, 마지막 단계는 흔적제거 단계로 과시를 하던지 혹은 들키지

Fig 1. 기술적 공격의 단계[6]

않게 다음공격을 위해 백도어(backdoor)를 설치하는 단계를 말한다.

2.2 사회공학 공격 단계

기존 연구[7]에서 연구한 사회공학 공격단계도 기술적 공격단계와 유사하다. 목표 대상에 대한 정보를 수집하는 단계로 대상의 가족, 개인활동, 사회활동을 포함한 모든 정보를 수집. 정보의 양과 질에 따라 성공여부가 결정하는 정보수집 단계에서부터, 공격자가 요구하는 것을 의심 없이 쉽게 수용할 수 있도록 신뢰를 쌓는 과정인 관계 및 라포형성 단계, 적극적인 공격단계와 공격을 받았다는 사실을 인지하지 못하도록 공격활동의 정보나 증거를 정리하는 실행단계로 Fig 2.와 같다.

Fig 2. 사회공학 공격의 단계[7]

2.3 사회공학 사이버작전의 단계[5]

최근 연구된 사회공학 사이버전의 모델링을 보면 Fig 3.과 같으며 이 모델은 기존 사회공학 공격의 수행단계와 유사하다. 정보를 수집한 후 심리학의 인지이론을 적용해 관심영향 심리기제와 인지판단 심리기제를 적용하여 라포를 형성하여 행동 강화 심리기제로 공격과 실행을 한다. 이 연구 모델의 특징으로는 이 모든 단계는 기술적 공격 수행단계의 첫 번째 단계인 정보수집 단계에서 이루어진다는 것과 기존 사회공학 공격모델과 다르게 방략 선택단계가 있다는 것이다. 공격자는 정보 수집을 통해 어떤 공격이 성공할 것인지를 미리 생각하는 방략을 선택 한 후 해당 방략에 맞게 라포를 형성한다고 제시하고 있다. 이와 같이 모델링한 단계를 거친다면 공격자의 입장에서는 목표달성에 유리할 것으로 판단된다.

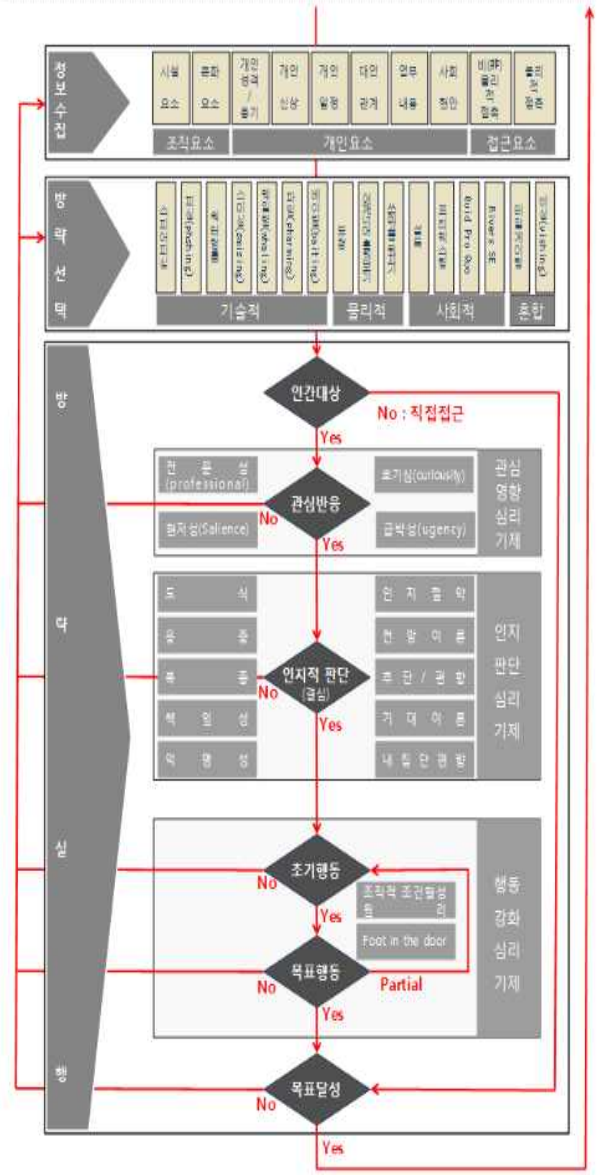


Fig 3. 사회공학 사이버전 공격 모델링[5]

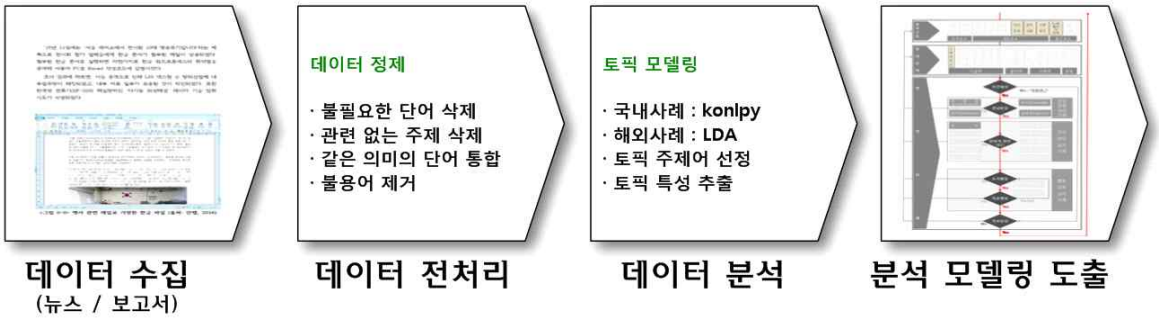


Fig 4. LDA를 이용한 사례분석 처리 단계

III. 사회공학 공격유형 산출방안

위에서 제시한 사회공학 공격단계의 검증방법은 대부분 사례를 통한 방법이나 설문조사를 통한 방법으로 검증 및 대안을 제시하였다. 설문을 이용하여 검증하는 방법은 설문지의 항목과 모집단의 성격에 따라 공격의 한계와 결과의 차이가 존재한다. 그러므로 대부분의 연구는 사례를 통해 단계나 모델링을 검증하는 방법을 이용하였다.

사례분석 방법을 적용하는 것이 사회공학 공격기법이 실생활에서 어떻게 적용하여 공격되는지 보다 더 체계적으로 이해할 수 있기 때문이다. 이러한 사례분석 검증방법은 기사나 보고서를 통해 사람이 직접 검색하여 찾은 후, 분석대상을 해당 목적에 맞게 분류 후 사회공학과 연계된 핵심단어를 추출하여 각 단계에 적용하는 방법으로 진행된다.

그러나 이와 같은 방법은 사회공학으로 해당되는 기사나 보고서를 일일이 찾아야 하고 그런 다음 요약을 통해 핵심단어를 추출하여 단계에 적용하다 보니 시간이 다소 걸리며, 핵심단어를 놓치거나 혹은 오역이 발생할 있다. 이러한 문제로 사례를 검증하는 과정이 사람이 아닌 프로그램을 통해 검증한다면 핵심단어를 놓치거나 오역의 건수가 줄어들 수가 있을 것이다.

토픽 모델링(Topic Modeling)은 문서 집합의 추상적인 "주제"를 발견하기 위한 통계적 모델 중 하나이다. 텍스트 본문의 숨겨진 의미구조를 발견하기 위해 사용되는 텍스트마이닝 기법으로 기계 학습 및 자연언어 처리의 한 분야이다. 이와 같은 방법을 이용하여 핵심단어를 추출하면 단어의 신뢰도가 향상될 것이다. 토픽 모델링 방법 중에서도 문서가 토픽들로부터 생성된 단어로 이루어진다는 것을 가정을 하고 확률모델을 통해 문서의 토픽과 토픽에 분포하는 단어들을 분석하는 방법인 잠재 디리클레할당(Latent Dirichlet Allocation, LDA)방법을 사용하여 공격유형을 산출하고자 한다. 이 방법은 문서안의 토픽 하나하나가 디리클레 분포[1]를 따르며 문서를 토픽에 할당한다는 의미를 가진다. 잠재 디리클레 할당(LDA) 방법을 이용한 사회공학 공격유형 산출방안은 다음과 같은 방법으로 프로그램 되어 핵심 단어를 추출하여 결과의 모델링에 적용된다. 첫 번째 단계는 문서에 핵심 단어가 몇 개가 있는지를 결정하는 것이다. 그런 다음 데이터 전처리 과정을 거쳐야 한다. 데이터 전처리 과정은 처음에는 사람이 직접 처리해야 하지만 그 과정이 반복될수록 사람이 개입되지 않고 데이터 전처리를 할 수 있다. 세 번째 과정은 데이터를 분석하는 과정이다. 문서마다 토픽이 어떻게 분포되어 있는지, 토픽마다 단어 분포가 어떤지 디리클레 분포에 따라 정의하며 문서의 제일 첫 단어부터 끝 단어까지 현재의 위치의 단어가 어떤 토픽과 관계있는지를 정하고 현재 위치의 토픽에 따라 단어의 분포를 결정하고 그 중에서 가장 높은 확률의 단어를 선택 하는 방법으로 분석한다. 한국어는 한국어 형태소 분석기(konlpy)

1) 연속 확률분포의 하나로, k차원의 실수 벡터 중 벡터의 요소가 양수이며 모든 요소를 더한 값이 1인 경우에 확률값이 정의되는 연속확률분포

를 사용하며 데이터를 분석하며 영어로 된 데이터는 명사와 동사 원형만 걸러낸 후에 LDA 사용하며 분석한다. 일련의 전제과정을 요약하면 Fig 4.와 같은 방법으로 진행된다.

IV. 결 론

사회공학 기법은 공격 대상이 기술의 취약점이나 시스템이 아닌 바로 사람의 심리를 이용한 공격이기에 완벽하게 방어하기에는 다소 무리가 있기에 꾸준히 증가하고 있는 공격 형태이다. 본 연구에서는 이러한 사회공학 공격방법도 기술의 의한 공격단계와 마찬가지로 일련의 단계를 거쳐서 공격하는 것으로 보고 이를 검증하는 방법으로 LDA를 이용한 사례분석방법을 제안하였다. 제안한 방법을 이용한다면 기존의 사례분석 방법보다 핵심단어를 추출하는 데에 있어 방법과 시간이 절약될 것으로 예상되며, 핵심단어의 데이터를 지속적으로 추출한다면 확장된 사회공학 공격방법을 예상하거나 새로운 공격 시나리오를 산출 할 수 있을 것으로 기대된다.

[참고문헌]

[1] Chi. M. "Security Policy and Social Media Use", 2011 The SANS Institute, 2014.

[2] Social Engineering: I-E based Model for Human Weakness for Attack and Defense Investigations, 2017.

[3] 박재혁, 이재우, 인간의 감정상태를 이용한 사회공학 기법 연구, 정보보호학회, 2015.08

[4] 정고은, 김영래, 민윤기, 심리적 트리거를 적용한 보이스피싱 사례분석-사회공학을 중심으로, 사회과학연구, 2017. 10.

[5] 강정호, 신규용외 5명, 사회공학기법과 사이버작전과의 연계성 연구, 사이버사령부, 2017. 11.

[6] 육군사관학교 컴퓨터과학과, 공군사관학교, 해군사관학교, 육군 3사관학교, 사이버전 개론 2nd Ed., 양지사, 2016. 02.

[7] 김홍진 외 3명, 사회공학 취약점 진단에 대한 프레임워크와 자동화 도구에 관한 연구, 한국IT서비스학회, 2014. 11.

[8] Wenjun Fan, Kevin Lwakatare and Rong Rong, "Social Engineering: I-E based Model of Human Weakness for Attack and Defense Investigations," Computer Network and Information Security, 1-11, Jen 2017

[9] John Orbell, Robyn M. Dawes, "A "Cognitive Miser" Theory of Cooperators' Advantage", The American Political Science Review, Vol. 85, No. 2, pp. 515-528, Jun 1991

음성인식 기반 사용자 인증 기술 동향 조사

김베드로, 김형식
성균관대학교 전자전기컴퓨터공학과

Survey on Voice Recognition Based User Authentication

Bedeuro Kim, Hyoungshick Kim

Department of Computer Science and Engineering,
Sungkyunkwan University

요 약

모바일 기술의 대중화와 센싱 기술의 발전으로 인해 생체 정보를 이용한 다양한 인증방식이 소개되고 있다. 본 논문에서는 모바일 환경에서 사용자의 편의성과 보안성을 만족시킬 수 있는 음성인식 기반의 사용자 인증 방식의 연구동향에 대해 분석하였다. 이를 통해 음성인증 방식 연구에서 고려해야할 요소와 앞으로 나아가야할 연구 방향에 대해 모색해보고자 한다.

Ⅰ. 서론

온라인 서비스에 대한 사용자 인증 방식으로 패스워드, 공인인증서와 같은 인증 기법들이 사용되어 왔다. 하지만 이러한 인증 기법들은 사용자의 기억에 대한 의존도가 높고, 본인 확인 정보가 담긴 매체를 항상 지니고 다녀야하기 때문에 사용자의 편의성을 낮추는 문제를 지니고 있다.

이를 해결하기 위한 방안으로 사용자의 고유한 생체 정보를 활용한 생체 기반 인증방식이 주목을 받고 있다. 특히 모바일 기기의 대중화와 센싱 기술의 발전으로 사용자의 생체 정보를 언제 어디서든 쉽게 확인할 수 있게 되면서 생체 기반 인증방식에 대한 관심이 높아지고 있다. 홍채, 지문, 정맥 등 다양한 생체 정보가 존재하지만, 그 중에서도 음성은 모바일 환경에서 사용자의 편의성을 높이면서도 보안성을 유지할 수 있는 사용자 정보로서 각광을 받고 있다.

음성인식 기반 사용자 인증 기술이 주목을 받고 있다. 음성인식 기술은 크게 화자식별(Speaker Identification)과 화자검증(Speaker Verification)으로 나눠 설명할 수 있다.

- 화자식별: 등록된 여러 화자의 음성 가운데서 입력으로 들어온 음성과 일치하는 화자를 찾는 기술.
- 화자검증: 화자 한명에 대한 음성정보에 대해 실제 화자와 일치하는지 판단하는 기술.

더 나아가 입력되는 음성 내용에 대한 의존 유무에 따라 문장 독립(text-independent)방식과 문장 종속(text-dependent)방식으로 분류할 수 있다.

본 논문에서는 음성인식 기반 사용자 인증 기술과 관련한 연구 동향에 대해 살펴보고, 향후 음성인증 방식에서 요구되는 연구방향에 대해 논의할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 문장 독립과 문장 종속 방식에 대해 기술할 것이다. 3장에서는 화자식별 관련 기술의 연구 동향을 살펴보고, 4장에서는 화자검증에 대한 연구를 알아볼 것이다. 마지막으로 5장에서는 본 논문에서 조사된 연구 동향을 정리하여 결론을 제시한다.

Ⅱ. 문장 종속과 문장 독립방식

음성인식 시스템에 영향을 주는 요인으로 화자의 음성적 특징 외에도 음성의 내용이 시스템의 성능에 주요 요인으로 작용할 수 있다. 이는 문장 독립과 문장 종속 방식으로 나눌 수 있으며 특징은 아래와 같다.

2.1. 문장 종속 방식

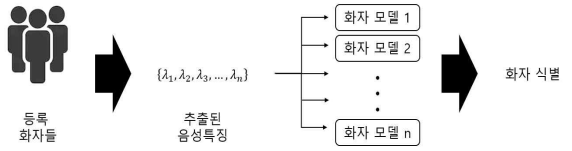
화자인식 시스템을 통해 등록한 음성의 문장과 실제 화자임을 검증하는 테스트에서 입력한 음성의 문장이 반드시 일치해야하는 방식을 의미한다[1]. 이는 사용자가 화자인식 시스템을 사용하기 이전에 말해야할 문구를 미리 숙지해야하기 때문에 편의성이 낮아지는 문제를 가지고 있다. 하지만 주어진 문장만을 사용하기 때문에 입력되는 문장의 양이 적어지고 신뢰성있는 모델을 만들 수 있다는 장점을 가지고 있다.

2.1. 문장 독립 방식

문장 종속 방식과 반대로 화자인식 시스템에 등록된 음성의 문장형태와 새로 입력한 음성의 내용이 서로 달라도 되는 방식이다[2]. 시스템에 대한 화자의 자유도가 문장 종속 방식에 비해서는 비교적 높다는 장점을 가지고 있다. 하지만 화자 인식 과정에서 입력되는 음성 이외에도 내용에 대한 고려가 필요하기 때문에 시스템의 cost complexity가 높아지는 단점이 있다.

Ⅲ. 화자식별

화자식별은 다수의 화자들로부터 입력된 음성 정보를 기반으로, 입력으로 주어진 음성이 어떤 화자의 것인지를 찾아내어 구분하는 기술을 말한다. 화자식별 시스템의 동작과정은 [그림 1]과 같다.



[그림 1] 화자식별 시스템 동작과정

먼저 특정 시간동안 여러 화자들로부터 얻어진 음성 정보(발성)에 대해 각 사용자별 음성의 특징을 추출하게 된다. 이렇게 추출된 음성 특징 데이터를 기반으로 화자모델을 훈련시켜 화자식별 시스템을 구축한다. 이를 이용하여 특정 화자의 음성이 입력으로 들어오면 다수의 화자들 중에서 특정 화자가 누구인지 식별을 하게 된다. 이는 실시간 처리를 필요로 하는 폰뱅킹, 내비게이션과 같은 서비스에 적용될 수 있다. 하지만 시스템에 등록된 화자 이외에 사용자 음성을 입력으로 이용하게 되면 등록된 음성 중 가장 유사한 음성을 화자로 인식하는 보안 문제가 발생한다는 단점을 가지고 있다. 화자식별에 대한 일반적인 접근 방법으로는 데이터 분포로부터 화자의 특성을 나타낼 수 있는 가우시안 혼합 모델 기반 방식과 분류 모델 중 하나인 서포트 벡터 머신 기반 방식이 존재한다[1, 2]. 이 장에서는 화자식별에 이용되고 있는 방법들에 대해 알아보고자 한다.

3.1. GMM방식의 화자식별

시스템에 등록된 여러 화자들로부터 추출한 음성 특징 벡터에 대해 최대우도법(Maximum Likelihood)을 적용하여 가장 높은 확률을 나타내는 화자를 선별하는 방식이다.

화자들의 음성에서 특징 벡터를 추출하는 방식으로는 주로 MFCC (Mel-Frequency Cepstral Coefficients) [2]기법이 사용되며, 이외에도 LPCC (Linear Predictive Cepstral Coefficients) [3] 또는 LPCs (Linear Prediction Coefficients) [4] 방식들이 사용되고 있다. MFCC는 1Khz 이상의 주파수를 인식할 수 없는 사람의 청각적 특성과 더불어 인간의 귀가 지닌 임계 대역폭의 변화에 초점을 둔다. 이처럼 사람의 신체적 특성에 기반을 둔 MFCC은 음성 인식에서 적합한 특징 추출방식으로 여겨지고 있다[5].

음성에 대한 특징 벡터를 이용하여 화자를 식별하기 위해서는 각 화자에 대한 통계모델이 필요로 되며 GMM (Gaussian Mixture Model)이 보편적으로 사용되고 있다. 이 모델을 적용하여 적합한 음성 특징 벡터를 선별하기 위해서 최대우도법 (Maximum Likelihood)과 E-M (Expectation-Maximization)알고리즘이 사용된다. 하지만 신뢰성있는 결과 값을 추정하기 위해서는 각 화자에 대해 많은 훈련 데이터가 있어야 하는 단점을 가지고 있다. 이를 보완한 모델로 GMM-UBM (Gaussian Mixture Model-Universal Background Model)이 있다. 불특정한 다수의 화자들로부터 수집된 배경화자들의 음성 데이터를 이용함으로써 다량의 데이터에 대한 훈련을 필요로 하는 GMM의 단점을 보완하였다.

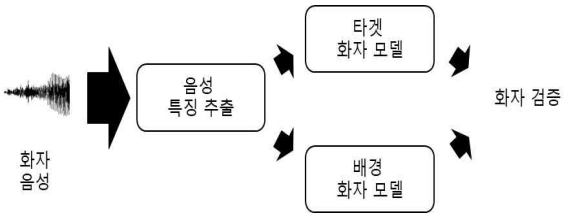
3.2. SVM 방식의 화자식별

SVM (Support Vector Machines)이 소개된 이후로 강인한 음성 특징을 추출하여 화자를 식별하는 연구가 진행되고 있다[6]. 이와 관련하여 기존의 GMM방식에 SVM을 적용한 GSV - SVM (Gaussian Mixture Models Super Vecto r-Support Vector Machine)방식이 고안되었다. 이는 훈련에 필요한 데이터의 양적 문제를 해결한 GMM-UBM과 분류의 정확도가 높은 SVM의 장점을 혼합하여 화자식별의 성능을 개선하고자 하였다. 하지만 기본적인 SVM은 이진 분류에 특화되어 있기 때문에 다수의 클래스를 분류하는데 있어 한계를 가지고 있다. 이러한 문제점을 해결하기 위하여 이진 분류 방식을 멀티 클래스 분류 영역으로 확장한 방법들이 적용되었다. 대표적으로 One-against- the - Rest와 One - against - One

방식이 존재한다[7]. One-against-the - Rest를 적용한 음성 분류기는 K개의 서로 다른 멀티 클래스가 존재할 때, 특정 클래스 하나와 나머지 K-1개의 클래스에 대한 이진 분류기를 전체 클래스 K만큼 생성하게 된다. 새롭게 입력된 화자의 음성을 테스트셋(Test set)으로 사용하여 클래스 스코어가 가장 높은 화자를 입력된 음성의 화자로 인식하게 된다. 이에 반해 One - against - One은 전체 클래스 중 2개의 클래스를 선택하여 분류기를 생성하게 된다. 마찬가지로 새로운 입력 음성을 기반으로 클래스 스코어가 가장 높은 화자를 입력된 음성의 화자로 인식하게 된다.

Ⅳ. 화자검증

화자검증은 입력된 음성이 특정 화자의 음성인지 아닌지를 판별하기 위해 사용되는 기술이다. 화자검증에 이용되는 모델 기반 방식의 화자검증 시스템은 [그림 2]와 같다.



[그림 2] 화자검증 시스템 동작과정

화자식별 방식과 마찬가지로 입력된 화자의 음성으로부터 음성 특징을 추출하게 된다. 다음으로 시스템 상에 저장되어 있는 타겟 화자의 음성과 비교하여 얼마나 유사한지를 계산하게 된다. 계산 결과가 일정 기준 값 이상이면 화자임을 확인하여 사용자가 서비스를 이용할 수 있게 된다. 화자검증에 대한 성공은 타겟 화자에 대한 승인 혹은 거짓 화자에 대한 미승인으로 판단되며, 반대 경우에 대해서는 화자검증이 실패한 것으로 본다.

화자검증 시스템의 정확성은 입력된 화자의 음성과 등록된 화자 사이의 차이에서 발생하며 이와 관련하여 일반적으로 두 가지의 요인이 존재한다. 첫째로 입력된 화자가 말한 음성과 시스템에 등록된 화자가 말한 음성의 문장 구성이 다른 경우이다. 두 번째로는 화자의 음성에 대한 정보가 불분명한 경우이다. 예를 들어 주변의 잡음으로 인해 화자의 음성이 잘 들리지 않거나 짧은 녹음 시간으로 인해 화자의 발언 정보가 적을 때이다. 현재까지 잡음에 대한 연구는 많이 이루어지고 있지만, 짧은 음성 정보에 대한 인식연구는 적은 편이다. 이 장에서는 짧은 발성에 대한 화자검증 연구에 대해 살펴본다.

4.1. DNN방식의 화자검증

최근 들어 DNN (Deep Neural Network)을 적용한 화자인식이 기존의 GMM 모델에 비해 높은 향상 보이면서 이에 대한 관심이 높아지고 있다[10]. 특히 길이가 짧은 화자 음성에 대해 DNN을 적용하여 인식의 정확도를 개선하고자 하는 연구가 이루어지고 있다.

McLaren 등은 i-vector 추출 과정에서 기존의 GMM-UBM 대신 DNN을 적용한 화자 모델을 제안하였다. 제안기법에 쓰인 i-vector는 화자의 발성 단위에 따라 음성 데이터를 처리하는 방식으로 화자인식 분야에서 특징 추출방식으로 이용되고 있다. Variani 등[11]은 i-vector 대신 DNN을 이용해 추출된 d-vector를 특징 추출 방식으로 사용하였다. 새로운 화자가 음성을 입력하게 되면 트레이닝 된 DNN 모델을 통해 맨 마지막 층에서 화자의 특징 벡터가 추출되며 이를 기존에 등록된 화자들의 특징 벡터들과 비교하여 화자를 검증하게 된다. 제안 방식에 대한 실험은 문장 종속적인 짧은 음성 데이터를 이용해 진행되었으며, 그 결과 d-vector가 i-vector에 비해 높은 성능을 가지는 것으로 확인되었다. 또한 DNN기반의 화자검증 시스템이 잡음에 강인하며 낮은 오거부율 (타겟 화자를 타 화자로 인식할 확률)을 가지는 것을 발견하였다. Lee 등[13]은 DNN모델 중 오토인코더(Auto-Encoder)를 활용한 음성 특징 추출 방식을 제안하였다. 오토인코더는 입력과 출력층의 개수가 동일한 특징을 가지고 있으며, 학습을 통해 입력과 출력층의 모양을 동일하게 만들어가는 과정에서 히든 레이어에 입력에 대한 유의미한 특성값이 생성되게 된다. 제안 논문에서는 오토인코더의 특징을 활용하여 타켓 화자의 음성을 훈련시키는 과정에서 히든 레이어를 통해 타켓 화자의 음성 특징인 i-codes를 추출해낸다. 이를 입력으로 들어오는 미지정 화자의 음성 특징값과 비교하여 화자를 검증하였다.

Ⅴ. 결론

본 논문에서는 음성인식 기술에 대한 연구동향을 살펴보았다. 먼저 음성인식을 구성하는 화자식별과 화자검증 방식에서의 연구들에 대해 알아보았다. 연구동향에 대한 분석결과, 화자검증 방식에서 음성 특징을 추출하기 위한 다양한 방식들이 연구되고 있음을 확인하였다. 또한 화자검증 시스템에서 입력된 음성의 길이가 짧을 경우, 이를 인식하기 위해 DNN 모델이 주로 사용되고 있음을 알 수 있었다.

이러한 연구동향을 바탕으로 추후 여러 DNN 모델을 적용한 음성 특징 추출 방식에 대한 연구나 입력되는 화자의 음성적 특징(길이, 구성)에 상관없이 화자를 인식할 수 있는 시스템에 대한 연구가 가능할 것으로 보인다.

감사의 글

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-01369, O2O 서비스를 위한 무자각 증강인증 및 프라이버시가 보장되는 블록체인 ID 관리 기술 개발)

[참고문헌]

[1] Heigold, G., Moreno, I., Bengio, S., and Shazeer, N., "End-to-end text-dependent speaker verification," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5115-5119, 2016.

[2] Kinnunen, Tomi, and Haizhou Li. "An overview of text-independent speaker recognition: From features to supervectors," Speech communicatio n, Vol. 52, No. 1, pp. 12-40, 2014.

[3] D. A. Reynolds and R. C. Rose, "Robust text independent speaker identification using Gaussian mixture speaker models," IEEE Transaction Speech Audio Process. Vol. 3, no. 1, pp. 72-83, 1995.

[4] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," IEEE Signal Process, Vol. 13, pp. 308-311, 2006.

[5] J. R. Deller, Jr., J. H. L. Hansen, and J. G. Proakis, "Discrete-Time Processing of Speech Signals," IEEE Press, 2000.

[6] L. R. Rabiner and R. W. Schafer, "Digital Processing of Speech Signals," Prentice-Hall, 1978.

[7] H. Hermansky, "Perceptual Linear Pred ictive (PLP) Analysis of Speech," Journal of Acoustic Society America, Vol. 87, No. 4, pp. 1738-52, 1990.

[8] "Mel-frequency cepstrum," Wikipedia, acce ssed 26 June 2015.

[9] He, L., Zhang, W. Q., Shan, Y., and Liu, J. "Channel compensation technology in differential GSV-SVM speaker verification system," IEEE Conference on Asia Pacific Circuits and Systems, 2008.

[10] Weston, Jason, and Chris Watkins. "M ulti-class support vector machines," Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

[11] Lei, Y., Scheffer, N., Ferrer, L., and McLa ren, M. "A novel scheme for speaker recognition using a phonetically-a ware deep neural network," IEEE Inter national Conference on Speech and Signal Processing, pp. 1695-1699, 2014.

[12] Variani, E., Lei, X., McDermott, E., Lope z-Moreno, I., and Gonzalez-Dominguez, J.. "Deep neural networks for small footprint text-dependent speaker verification," IEEE Inter national Conference on Speech and Signal Processing, Vol. 14, pp. 4052-4056, 2014.

[13] Lee, H. S., Lu, Y. D., Hsu, C. C., Tsao, Y., Wang, H. M., and Jeng, S. K., "Discriminative autoencoders for speaker verification," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5375-5379. 2017.

An Investigation and Analysis of Blockchain Use Cases for Digital Forensics

Sandi Rahmadika and Kyung-Hyune Rhee*

Interdiscplinary Program of Information Security, Graduate School PKNU
*Department of IT Convergence and Application Engineering
Pukyong National University, Republic of Korea
e-mail: sandika@pukyong.ac.kr, khrhee@pknu.ac.kr

Abstract

Blockchain technology has a significant impact on various fields of science. It has also changed the way people transact. The nature of blockchain can be used in forensic science to build decentralized systems in terms of managing the digital forensics data. The problems arise when various parties get permission to access digital forensics data due to it can directly damage the integrity of the data. Therefore, blockchain can be used as a solution to address the problems. In this paper, we thoroughly review the prominent existing model of blockchain in the digital forensic domain. We emphasize several points based on the selected literature reviews. In the last session, we present our view regarding the relationship between blockchain and digital forensics. This paper is expected to become a simple reference for stakeholders for developing blockchain technology in digital forensics domain.

I. Introduction

The integrity of a digital forensic data is very essential for law enforcement because it is directly related to the authenticity of evidence. In the current digital forensics system, there are still some issues including management of evidence [1]. In this sense, the digital evidence may be accessed by several different parties in the investigation so that if it is not periodically controlled, the evidence may be changed or not genuine either intentional or not.

Blockchain for digital forensics must satisfy the following key features [2]:

- **Integrity**: the evidence should not be changed or altered by the parties.
- **Traceability**: the data of digital forensic must be traceable to the owner.
- **Tamper-proof**: Blockchain provides this feature which is very difficult to damage stored data (in form of blocks).
- **Verifiability**: All processes that occur must be confirmed by the system.
- **Authentication**: For all parties involved, their identity must be known and have access rights to manage data.

In this paper, we present the blockchain model in the digital forensic domain. We thoroughly select the model from various literature reviews. This study aims to determine the extent to which blockchain can be applied in the forensic domain. Blockchain technology for digital forensics must be mature enough before being applied in the actual system considering that forensic data is essential for law enforcement.

This research is expected to be a consideration for blockchain developers in the digital forensic area to be used as a reference. We explain some of the advantages of blockchain in digital forensics, drawbacks, obstacles, and others. In addition, we also emphasize several points to be discussed as the first step for system development.

II. Blockchain and Its Applications

Blockchain technology offers to radically transform the way parties exchange their digital asset securely without having to trust the third party nor a central authority [3]. In

theory, since there is no third party presence in the system [4], blockchain can reduce operating costs in transactions and eliminate errors that can be made by the middleman.

Since its emergence, blockchain has been widely applied in various disciplines such as healthcare, financial, energy, manufacturing, banking and finance, government and public growth and to name a few. With the features offered, blockchain is now penetrating into the forensic area so that research on blockchain for forensics has also been done. Therefore, the characteristics of blockchain are very suitable to be applied to the forensics domain to manage digital evidences.

By design, blockchain is inherently resistant to modification of the data. Blockchain properties can be used to secure data stored by cryptographic techniques. Data that has been stored on the blockchain network is almost unchangeable by the attacker.

III. The Presence of Models

In this section, we describe the presence of models for blockchain and digital forensics. The model has been carefully selected by considering a few points.

B-CoC is a blockchain-based Chain of Custody for evidences management in digital forensics which is proposed by Bonomi et al. [2] in 2018 as shown in Figure 1. The model was created with the aim of maintaining the integrity of the evidence due to the evidences are used by many parties involved. B-CoC uses the Istanbul Byzantine Fault Tolerance (IBFT) consensus which tolerates at most f faulty validators out of a total of $n = 3f + 1$ [5]. The command is executed using Ethereum platform and the smart contract on solidity language. It has four features, namely *CreateEvidence (ID, description), Transfer (ID, newoner), RemoveEvidence (ID), and GetEvidence (ID).*

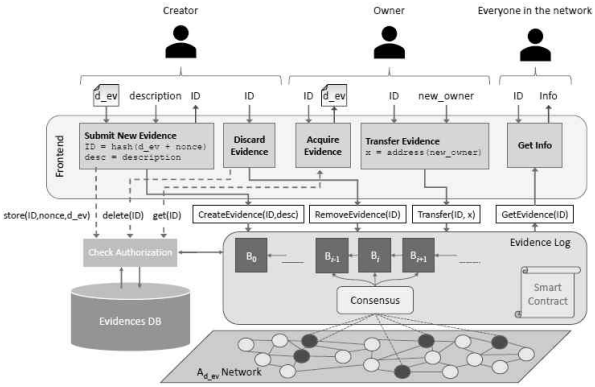


Figure 1. Blockchain-based chain of custody

In order to execute a command with 0 bits the data length required 170207 *gas (tx)*, while for the data length of 1024 bits requires around 897367 *gas (tx)* per transaction. Furthermore, for blockchain growth rate per year is around 3.09 GB with the maximum workload (n = 1000000) which is acceptable given the capacities of today's storage devices.

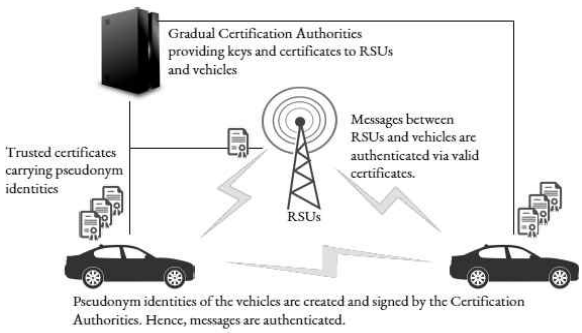


Figure 2. The model of VPKI

Block4Forensic - an integrated lightweight blockchain framework for forensic applications of connected vehicles which is proposed by Cebe et al [1]. The system is applied to the private blockchain to manage membership establishment and privacy. In other words, blockchain is used to provide comprehensive forensic services for accident investigation. The integration of the Vehicular Public Key Management (VPKI) model (see Figure 2) is also applied to support the process.

BC-DataLogging - the blockchain model based data logging and integrity management

system for cloud forensic is proposed by Park et al [6]. The purpose of this study is to solve the problem of data integrity in cloud services used in forensic areas.

Truth-based System - the way to improves cyber forensics and cybersecurity through blockchain technology and truth based systems [7]. Keyless Security Infrastructure (KSI) is used to ensure validation of virtual machine gold copy operating system and forensics toolkit. Whilst, it also used to ensure forensic workstation calibration and ensure no change to client data during the process.

IV. Discussion and Some Remarks

Based on several studies described in the previous section, blockchain technology implemented in digital forensics is used to maintain the integrity of evidence although in the form of different models and platforms. It is possible because the tamper-proof nature of the blockchain means that the data that has been verified and stored cannot be changed by the attacker unless the attacker controls more than half of the total network.

Apart from the advantages possessed by blockchain, this technology must mature enough in all aspects such as security, privacy, authentication, storage capacity and so on. Security properties are essential since directly related to sensitive data that is accessed by law enforcement agencies.

One way that can be used to disguise digital forensic on the blockchain network is to use the ring confidential transaction technique as shown in Figure 3 and algorithm to adding a new member that can be seen in Algorithm 1 which explains the basic procedures that can be used to create ring confidential transactions in digital forensics and procedures for adding new members to the group. Intuitively, this technique aims to hide the value of digital evidence by combining the total value of the previous transaction. By doing so, the observer cannot tell the information about the current transaction.

Figure 3. The forensic transaction on RCT group

A single of digital evidence can be accessed by various law enforcers who are members of the same blockchain network which is known in advance. Therefore, the integrity of the data must be maintained. If the evidence can be changed intentionally or not, it certainly disturbs the ongoing process.

Algorithm 1: Digital Forensics RCT

```
Procedure Create_RCT for digital forensic in BC
    if ∀tx BC use CRS then return True
    Curr.Value ← Value of ∀tx (in a group)
    RingCT ← chooseprev.txs ∈ RCTgroup
    Get MsgEvidence ∈ Auth. Parties
    Sign σ ← MsgEvidence, Prv.Key
    CurrMsg ← Sign σ || RingCT
end Procedure
Procedure AddNewMembers:
    Members_n ← NewOutput_txs
    Update RCT ← Members_n ⊕ RingCT
end Procedure
```

We conclude some points that must be considered before implementing blockchain in the forensic area which is based on several studies that previously presented.

- **Strong authentication scheme**. The system must be able to provide a strong authentication scheme. This is related to how each party proves that they have the right to access digital forensic data.

- **Identity management**. The identity of law enforcement must be known before joining the permission blockchain network. as well as for transactions that have occurred where each transaction output can be used as multiple inputs for the next transaction (see Algorithm 1).

  Procedure AddNewMembers:
  Members_n ← NewOutput_txs
  Update RCT ← Members_n ⊕ RingCT
  end Procedure

- **Consensus selection** must be appropriate by considering the type of system, parties, number of validators, and who is directly involved into consensus.

- **On-off chain storage**. Data storage systems are essential in the blockchain for digital forensic system. The capacity of a evidence in digital form plays an important role in the storage.

	On chain data	Off chain data
Data types	Standardized data fields containing summary information in text form (e.g. age, gender)	Expansive evidence details (e.g. notes) and abstract data types (e.g. MRI images)
Pros	Data is immediately visible for all connected organizations, making in the single source of truth.	Storage of any format and size of data
	Constrained in the type and size of data that can be stored	• Data is not directly and immediately visible • Requires off chain micro services

- **Scalability**. Developers are important to carefully consider who is directly involved in the system. Thus the type of blockchain, the type of consortium, and the ability of the system can be estimated so as to create an efficient system.

There are many ways that can be used to improve the security of a forensic system such as RingCT technique as we described in this paper. Blockchain in the forensic area cannot stand alone, it requires mature connections with various other technologies such as IoT and its devices. In addition, additional security systems are needed to ensure that the system is completely safe.

V. Conclusion

We have presented the opportunities of blockchain in the digital forensics domain. We also present several points that we consider important for the developer before the blockchain is applied to the actual forensic system. Overall, blockchain technology can be said to be suitable for managing digital forensic data with guaranteed data integrity even though the data has been accessed by various parties. For future work, we will provide a blockchain model to manage digital evidence based on related research.

Acknowledgment

This research was supported by the MSIT(Ministry of Science, ICT),Korea, under the ITRC(Information Technology Research Center) support program (IITP-2018-2015-0- 00403) supervised by the IITP(Institute for Information &communications Technology Promotion) and partially supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) (No. NRF-2018R1D1A1B07048944)

[References]

[1] Cebe, Mumin, et al. "Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles." arXiv preprint arXiv:1802.00561 (2018).

[2] Bonomi, Silvia, Marco Casini, and Claudio Ciccotelli. "B-CoC: A Blockchain-based Chain of Custody for Evidences Management in Digital Forensics." arXiv preprint arXiv:1807.10359 (2018).

[3] Toyoda, Kentaroh, et al. "A novel blockchain based product ownership management system (POMS) for anti-counterfeits in the post supply chain." IEEE Access 5 (2017): 17465-17477.

[4] https://www.jbs.cam.ac.uk/faculty-research/ Accessed: October 19th, 2018

[5] Gramoli, Vincent. "From blockchain consensus back to byzantine consensus." Future Generation Computer Systems (2017).

[6] Park, Jun Hak, Jun Young Park, and Eui Nam Huh. "Blockchain based data logging and integrity management system for cloud forensics" Computer Science & Information Technology: 149.

[7] Zatyko, Ken "Improving Cyber Forensics and Cybersecurity through blockchain technology with truth based systems" International Symposium on Forensic Science Error Management, July (2015).

식품 추적 및 제어를 위한 블록체인 솔루션

쵸느에진랏*, 이경현**
*부경대학교 정보보호학협동과정
**부경대학교 IT융합응응공학과

A Blockchain solution for food traceability and control

Cho Nwe Zin Latt*, Kyung-Hyune Rhee**
*Interdisciplinary Program of Information Security, Graduate School, Pukyong National University
**Department of IT Convergence and Application Engineering Pukyong National University

Abstract

A number of high profile recalls traced to imported tainted ingredients and contaminated seafood, domestically produced fresh vegetables, and found beef. Many of the food recalls have been linked to safety and quality problems in the supply chain and media attention has focused mostly on the international supply. Today a new technology called the blockchain which is a ground-breaking innovation in decentralized information technology presents a whole new approach. We outline that mainly in the paper to solve about food fraud problem by using blockchain technology.

I. Introduction

Nowadays, food safety has drawn growth of academic and commercial concerns. Good traceability systems help to minimize the production and distribution of unsafe or poor quality products, thereby minimizing the potential for bad publicity, liability and recalls. Food and food production is one of the largest industries in the world. It is also the most fragmented industry with production scattered all over the world.

The food supply chain [1] also becomes more global over times. Keeping control of the supply chain of food is therefore a costly and difficult task. The dependence on trust in third party operations, ethics in production, transportation, to name a few areas, is evident. Stamps and documentation, IT-systems, certificates, food origin, mixing of food, the use of chemicals etc. are areas where fraud or ignorance can create problems on a large scale. At worst, it can cause health problems, even deaths. Traceability has become increasingly important.

The Global Food Traceability [2] Center is intended to assist companies to better understand and implement ways to track and trace of the paths of products through the food chain, to improve food safety and security and to avoid or litigate devastating public health and economic impacts. However, we want to emphasize that food traceability is about more than recalls. Being able to ascertain the origin of products, ingredients and their attributes, from the farm through food processing to retail, food service and the consumer is growing in importance. In our eating habits along with more stringent

government regulations and requirements have resulted in new food handling, preparation, storage and production practices. Whether users are in food production, storage, packaging or preparation a comprehensive food safety management system in critical to the success of your food and beverage business. Beyond basic food regulations and acceptable workplace practice, user needs contingency plans for potentials crises such as product recalls or forborne illness outbreaks. We hope that our system could make food traceability from "farm to fork" become a reality, and rebuild public confidence in the food supply chain.

The paper is organized as followed. In section 2 presents that blockchain solution in the food industry. In section 3 discussion about food fraud problem and the section 4 is conclusion about this paper.

II. Blockchain solution in the food industry

Blockchain technology [3] helped to reduce the paperwork required to process the containers. Those sorts of transposition documents, such as the bill of lading can easily be tampered with our copied, making the supply chain vulnerable to criminals who can replace goods with counterfeit products. IBM has also been on a busy streak signing deals across various industries to explore blockchain as a use case and pilot new projects, especially with stakeholder involved in the food supply chain. By bringing together the different stakeholder of the value chain and coupling it wit Blockchain, in Figure 1, IBM [4] is creating the opportunity align trust and information sharing between farmers, processor, distributors and retailer make up the long supply chain. The complexity of the food supply chain and the decentralized nature truly requires a decentralized solution such as Blockchain.

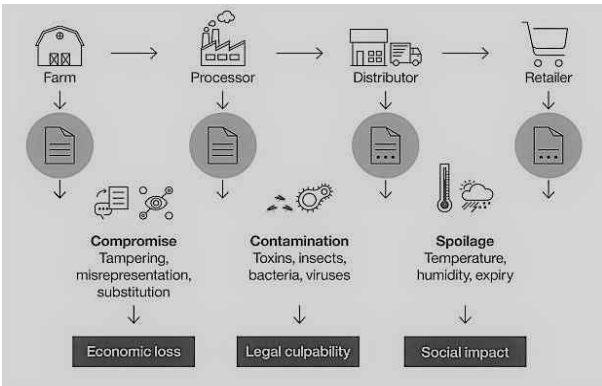


Fig.1. Blockchain technology allows better information management along the value chain.

III. Discussion

By using blockchain technology in the food industry, we can solve about food fraud problems. If something happened in food, can solve immediately. User can find quickly the base of problem. Therefore, we will discuss about blockchain technology how to solve in food fraud problem, blockchain solution, security issue and privacy and smart contract. We presented perspective basically published paper as follow:

(1) The Problem

The problem is in the last decade we have seen the consumer progressively returning to healthier and more sustainable food habits. This cultural evolution is part of a greater understanding of the world. However, high quality food is often target of frauds and attempts, especially in markets from where the food product is produced.

(2) Blockchain solution

Forget products not only cause economic losses, but they can also have an impact on

the original product brand image. As a solution, allows manufacturers to guarantee the authenticity of their products. This allows importers to be sure that the products they are buying are in fact authentic. RFID [5] code or QR codes are applied on every single food or liquor item. Every RFID code or QR code are unique and linked to an online smart contract [6] system that stores it on a blockchain-based, decentralized database. The final consumer, when buying the products, can access the Authenticity Certificate by scanning the QR code with their smart phone that the item is authentic and the code was not scanned any other time, who the manufacturer is, when the product was purchased from the manufacturer, and who is the authorized distributor or importer. Users can solve with these methods in being food fraud problem.

(3) Security Issues

In the blockchain each node has a complete copy of the digital ledger. These can independently work to verify the transaction [7]. If the nodes don't agree, then the transaction is canceled. This system keeps the ledger tidy. Additionally, it makes it nearly impossible to commit a food fraudulent transaction and cryptographic keys used in blockchain, each keys are long, complex and virtually impossible to decipher unless you have to view it. The two-key system makes transaction even more secure in food industry and it does without sacrificing the transparency that makes the system so unique. Therefore, blockchain could create huge progress in food safety and reduce costs and saving lives.

(4) Integrity

There is a way to increase trust in the food industry. Blockchain solutions are already up and running in other industries and they are ready to apply in the food industry. With a blockchain track and trace solution, it was traced in seconds. When problems something occurs in the food, the restaurant or stores that served the food are often left wringing their hands. promising to get to the bottom of it but blockchain track and trace [8] will help them immediately track affected items to their origins locating the issue quickly so they can remove the contaminated products from menus, shelves and supply chains. That is the integrity of blockchain in food industry.

(5) Smart Contract

Smart contracts [9] are also a strength of blockchain. Attaching terms and conditions to transactions that are automated, so a computer can execute the terms such as transportation temperature sampling data specifications, thus preventing a shipment from being finalized and send out by a supplier if it did not meet buyer requirements. Blockchain also offers democratization of information with real-time visibility of the chain to all parties in a transaction at the same item. This is what can enable quick recalls.

In order that solution for the supply chain problems, the system should satisfy all of requirements. Those requirements are: Blockchain, Security Issues, Integrity and Smart Contract.

IV. Conclusion

The conclusion from this paper is that the prospect for blockchain solutions withing food traceability and control are substantial. Blockchian offers an immutable, auditable

platform to record data among a range of disconnected parties, with the promise of reduced traceability time, better supply chain collaboration and possible brand strengthening. However, many challenges in implementation are still to be overcome, such as how to translate posted information in the system and whether companies are willing to take the leap into transparent operations. In turns out that Bitcoin is not the only application for blockchain technology. It may prove useful in a number of other areas such as financial services, personalized health and food provenance.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2018R1D1A1B07048944)

[References]

[1] Tian, Feng. "An agri-food supply chain traceability system for China based on RFID & blockchain technology." Service Systems and Service Management (ICSSSM), 2016 13th International Conference on. IEEE, 2016.

[2] Bosona, Techane, and Girma Gebresenbet. "Food traceability as an integral part of logistics management in food and agricultural supply chain." Food control 33.1 (2013): 32-48.

[3] Pilkington, Marc. "11 Blockchain technology: principles and applications." Research handbook on digital transformations (2016): 225.

[4] "China: IBM And Walmart Use Blockchain Technology To Track Pork". 2017.Fleischwirtschaft.De. http://www.fleischwirtschaft.de/economy/news/China-IBM-and-Walmart-use-blockchain-technology-to-track-pork-33638.

[5] Hong, I-Hsuan, et al. "An RFID application in the food supply chain: A case study of convenience stores in Taiwan." Journal of food engineering 106.2 (2011): 119-126.

[6] Tian, F., 2016, June. An agri-food supply chain traceability system for China based on RFID & blockchain technology. In Service Systems and Service Management (ICSSSM), 2016 13th International Conference on (pp. 1-6). IEEE.

[7] Tse, Daniel, et al. "Blockchain application in food supply information security." Industrial Engineering and Engineering Management (IEEM), 2017 IEEE International Conference on. IEEE, 2017.

[8] Underwood, Sarah. "Blockchain beyond bitcoin." Communications of the ACM 59.11 (2016): 15-17.

[9] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." white paper (2014).

머신러닝에 기반한 PE패턴과 악성코드 시각화 분석을 활용하는 분석 방법에 대한 연구

염성규*, 신동일*, 신동규*

*세종대학교 컴퓨터공학과

A Research on Static Analysis method Exploiting PE Pattern and Malware Visualization Analysis based on Machine Learning

Seong-Kyu Yeom*, Dong-Il Shin*, Dong-kyoo Shin*,

*Department of Computer Engineering, Sejong University

요 약

ICT 기술이 발달함에 따라 오픈소스, 해킹 도구 등을 이용한 악성코드가 빠르게 증가함과 동시에 다양한 변종이 탄생하고 있으나 분석 및 분류를 진행하기 위한 전문가가 부족하다. 따라서 악성코드 분석을 자동화하려는 연구가 활발하게 이루어지고 있다. 본 논문에서는 기계학습을 이용하여 정적 분석 자동화 기법에 대하여 연구한다. 머신러닝에 기반한 PE 패턴 분석과 악성코드 파일의 시각화 분석을 통하여 비교 분석하고 조합하여 악성코드를 분류한다. 본 연구 방법을 사용함으로서 PE파일부터 쉘코드까지 다양한 악성코드를 분류할 수 있다.

I. 서론

ICT 기술이 발달함에 따라 언제 어디서나 원하는 정보에 접근하는 등의 일상생활에 편리함과 조직의 업무 효율성 제고에 긍정적인 영향을 미치고 있다. 하지만 한편으로 정보 유출, 웹사이트 훼손, 서비스 거부, 랜섬웨어, 좀비PC 등의 부정적인 영향 또한 증가하고 있다[1]. 또한 과거에 비해 현재의 악성코드는 난독화, 패킹 등으로 코드를 분석하기 어렵게 하고 다양한 기능, 모듈화, 지능화 등으로 점점 복잡해지고 있다. 그리고 오픈 소스 활성화로 다양한 변종 탄생과 전문지식 없는 사람도 해킹 도구를 사용하여 간단하게 악성코드를 제작 할 수 있게 되었다[2].

이러한 이유들로 종류가 다양해지고 수도 급증한 악성코드를 사람이 일일이 대응하는 것은 어려워졌다. 이에 따라 악성코드를 자동으로 분석 및 탐지, 분류하려는 연구들이 많이 진행되었다.

악성코드를 분석하는 방법에는 동적 분석 방법과 정적 분석방법이 있다. 대부분 동적 분석의 경우 샌드박스 기술을 이용한 바이너리 파일 실행을 통한 행위 분석 방식의 동적 분석 기술에 기반한 신종 악성코드 분석 및 탐지 기술에 의존하고 있는 경향이 있다[3]. 그러나 동적 분석은 악성코드를 직접 실행해야 한다는 단점이 있다. 이와 같은 문제점들로 동적 분석 자동화 시스템만으로는 효과적인 악성코드 분석 및 분류가 어렵다. 이에 악성코드를 정적 분석을 자동화 할 수 있는 기술에 대한 연구와 시스템이 필요하다.

본 논문에서는 기계학습과 딥러닝을 이용하여 악성코드 정적 분석 분류 시스템 제작한다. 2장에서는 현재 악성코드 정적 분석을 통해 분류하는 기술에 대한 관련 동향 및 기술에 대하여 정리하고 3장에서는 시스템 구조 설계 및 동작에 대해 설명하며 4장에서는 실험 및 성능 분석 결과를 제시하고 4장에서 시스템의 의의와 결론을 제시한다.

II. 관련연구

악성코드 정적 분석에 대한 연구가 다양하게 진행되고 있다. 프로그램에서 사용하는 API(Application Programming Interface)를 추출하여 해당 API가 악성코드에 발생하는 확률에 따라 가중치를 산정하여 분류하는 연구[4], 이진 파일 전체를 문자열로 바꾸어 N개의 연속된 요소를 추출하는 N-gram 기법으로 정상 파일과 악성코드 분류에 적용하는 연구[4], 바이너리 파일의 구조에 대하여 패킹과 난독화의 흔적을 정적 검사를 통한 위험 등급을 산정하여 악성 여부를 판단하는 연구, PE(Portable Executable) 헤더 및 섹션 정보에서 뽑은 정보를 바탕으로 악성코드를 분류하는 연구[5], 바이너리 파일 자체를 시각화 하여 CNN(convolutional neural network) 또는

RNN(Recurrent neural network)으로 분류하는 연구 등이 있다[6,7]. 연구 동향 중 악성코드에서 API가 발생하는 확률에 따라 가중치를 산정하는 경우 정적 분석과 동적 분석을 같이 진행하기 때문에 기본적으로 연산량이 크다. 또한 최근의 악성코드들은 API 탐지 기법을 회피하기 위하여 사용하지 않는 API를 추가하거나 대체 API를 사용한다. 이러한 점들로 악성코드의 탐지에 사용되는 API의 기준이 모호해진다는 단점이 있다. N-gram 기법을 사용한 악성코드 분류 기법은 악성코드 상에서 의미 없는 코드를 제거하여야 되고 패턴화된 공격 규칙을 이용하므로 변형된 악성코드 탐지에 취약한 단점이 존재한다. PE 헤더 및 섹션 정보에서 뽑은 정보를 바탕으로 악성코드를 분류하는 방법은 PE구조를 가진 파일들만 분류가 가능하다는 점이다. 하지만 악성코드는 웹쉘 등의 소스코드로만 이루어진 경우도 있다.

본 논문에서는 PE헤더에서 추출한 파일에 대한 55개의 특징과 엔트로피, 패커 등 추가정보를 제공해줄 13개의 특징을 추출하는 방법과 PE구조를 가지고 있지 않아도 분류가 가능하도록 바이너리 자체를 이미지로 시각화하여 분류하는 방법을 합쳐 악성코드 분류 시스템을 구현한다.

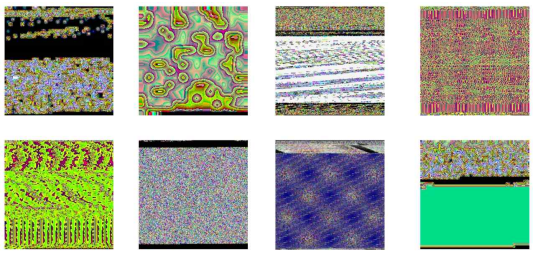
III. 악성코드 정적 분석 방법의 설계 및 구현

악성코드 분류 시스템은 정적 분석을 자동화하여 파일 자체의 특징으로 악성코드를 탐지 하도록 설계되었다. 탐지 시스템의 입력 데이터는 악성코드, 정상 파일, 소스 파일 등의 모든 입력을 데이터로 받는다. 아래 그림1은 전체 시스템의 구조도 이다.

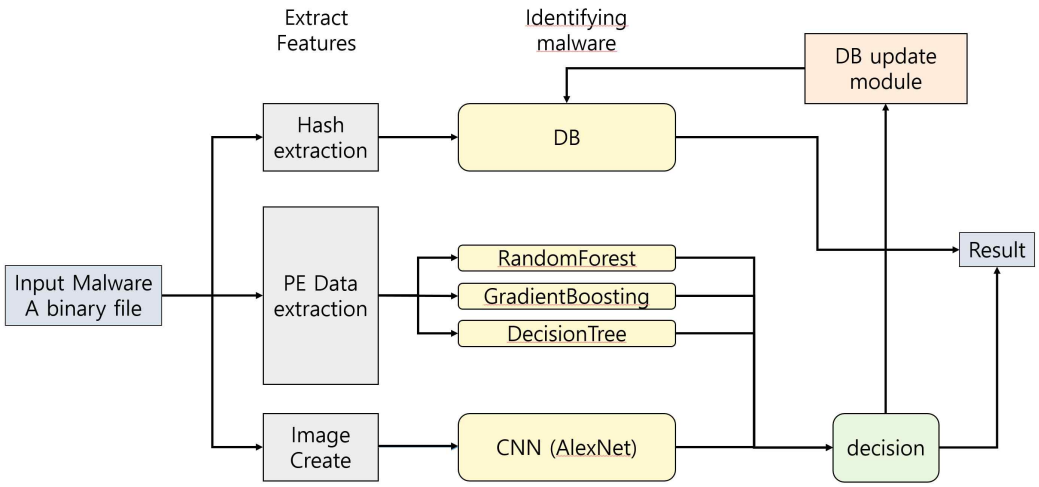
시스템의 진행 단계는 크게 전처리 단계, 악성코드 분류 단계, 결과를 DB에 반영시키는 단계로 나누어진다.

3.1 전처리 단계

시스템에는 여러 개의 기계학습으로 학습된 모델이 있다. 이 모델들에 데이터가 들어갈 수 있도록 입력된 파일을 정보 추출 및 가공을 진행하는 단계이다. 입력한 파일에서 해시값을 추출, PE 데이터 추출, 이미지 변환 작업을 진행한다. 추가로 입력한 파일이 PE구조를 가지고 있다면 파일에서 헤더 및 섹션 부분에서 엔트로피, 패커 등 총 55개의 특징을 추출한다. 이때 바이너리 파일에서 파일의 패킹 정보를 찾는 데에 YARA 룰 설정을 사용하였다. 기존 전통적인 악성코드 탐지 및 분류 시스템에서는 단순히 패턴을 비교하여 악성여부를 판단하였다면 설계된 시스템에서는 하나의 고려 대상으로 사용된다. 이미지 변환 작업은 입력된 파일을 CNN에 적합하게 이미지 데이터로 변환해주는 모듈이다. 입력 된 데이터는 1차원 벡터로 취급을 한다. 이 1차원 벡터를 수식 n=floor(√FileSize)를 통하여 n*n의 2차원 벡터로 변환한다. 변환된 벡터에 Deluxe Paint의 기본 팔레트를 참조하여 0~255의 숫자를 ID값에 대입시켜 3차원 벡터의 칼라 이미지를 생성한다. 아래 그림은 파일을 8bit 컬러 이미지로 변환한 결과이다. 그 다음 생성된 이미지를 CNN에 적용하기 위해 동일한 크기로 변경해주는 작업이 필요하다. OpenCV를 사용하여 256*256 크기의 이미지를 생성한다.



[그림 2] 악성코드를 시각화 결과



[그림 1] 악성코드 분류 시스템의 전체 구조도

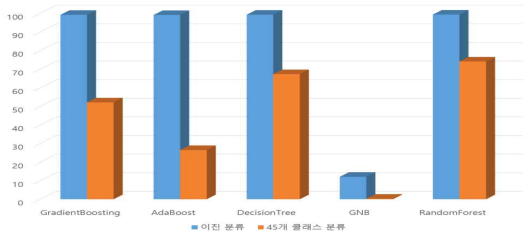
3.2 악성코드 분류 단계

악성코드 분류는 전처리 과정을 거친 데이터를 이용하여 진행한다. 분류에 적합한 모델을 찾기 위하여 다양한 실험을 진행 하였다. 실험에 대한 자세한 내용은 다음 장에서 설명한다. 실험 결과 PE구조 이용한 모듈에는 Decision Tree, Random Forest, Gradien Boosting 알고리즘을 사용하여 악성코드의 분류를 진행한다. 이때 54개의 속성들 중에 속성 별로 중요도 점수를 매겨 상위 9개에서 12개 사이의 속성을 사용한다. 이미지 모듈에서는 CNN의 알고리즘인 하나인 AlexNet을 이용하여 분류를 진행한다. 4가지의 분류 과정을 거쳐 투표방식을 사용하여 최종적으로 악성 여부를 판단한다. 투표 시 동률이 나올 경우 재분류를 진행하며 재분류 진행시에도 동률이 나온다면 정상을 악성으로 판단하는 것보다 악성을 정상으로 판단하는 것이 더 위험하므로 악성으로 판단을 진행한다. 판단된 결과는 DB에 가중치 값으로 수정하며 DB값이 특정 수치 이상이 되었을 경우 다음 분류부터는 분류를 진행하지 않고 DB만 보고 판단하여 분류를 진행하여 속도를 올린다는 장점이 있다.

IV. 실험

4.1 PE구조를 이용한 모듈

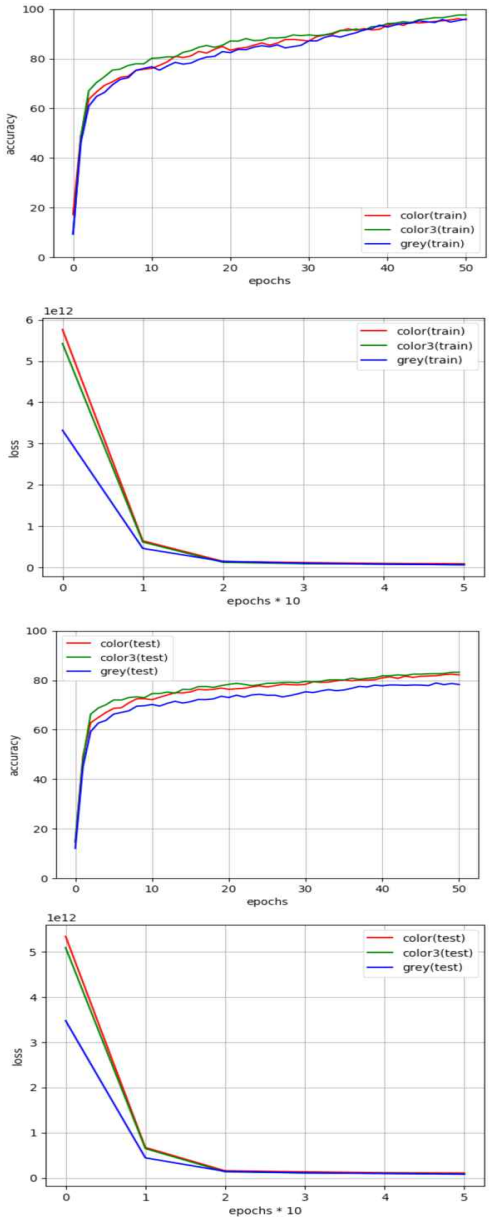
악성코드 분류에 적절한 알고리즘을 선택하기위하여 PE구조를 가진 악성코드들을 모듈별로 분류하는 실험을 진행하였다. 이진 분류는 VX heaven[8] 악성코드 랜덤으로 10000개와 윈도우 포터블 프로그램 정상파일 약 1000개를 분류하는 실험을 진행하였고 45개 종류의 악성코드 분류는 VX heaven에서 배포하여 제공해주는 약 248,347개의 악성코드를 45개의 그룹으로 상세히 분류한 뒤에 PE구조를 가진 214,458개의 추출하여 랜덤으로 80%를 잘라 데이터셋으로 사용하였다. 악성코드 분류에 적합한 알고리즘을 찾기 위하여 다양한 알고리즘으로 실험을 진행하였다. 알고리즘은 Decision Tree를 기반으로 50번 반복시킨 AdaBoost, 부트스트랩을 사용하여 10개의 깊이 제한이 없는 Random Forest, 사전 확률을 지정하지 않은 Gradien Boosting, 최대 깊이가 10으로 제한된 Decision Tree, 손실함수를 로지스틱 회귀를 적용하고 부스터를 50번 반복시킨 GNB를 사용하였다. 검증 방법은 나머지 20%의 데이터를 이용하여 교차검증 방법을 사용하였다.



[그림 3] 악성코드 기계학습 알고리즘 정확도 비교

4.2 CNN을 이용한 모듈

분류기의 성능 테스트를 위하여 Microsoft Malware Classification Challenge(Big 2015) 데이터셋을 사용하였다[9]. 9개의 클래스가 존재하며 10868개의 학습 데이터 중 PE 헤더가 제거된 바이트 코드 파일을 이용하였다. 랜덤으로 분할하여 80%를 학습하는데 20%를 검증하는데 사용하였다. 데이터를 흑백이미지, Deluxe Paint 매핑을 통한 컬러이미지, 3bit를 하나의 픽셀로 묶은 컬러이미지로 구성하여 실험을 진행하였다.



[그림 4] CNN으로 악성코드를 분류한 결과 (위의 두 그래프는 학습데이터, 아래의 두 그래프는 학습되지 않은 검증데이터)

3개의 실험 모두 학습데이터를 이용한 테스트에서는 비슷한 정확도를 보였으나 학습되지 않은 테스트 데이터를 이용하여 재 검증 결과 두 개의 컬러 이미지보다 흑백이미지가 약 3%정도 낮았다. 손실함수를 보면 흑백이미지가 컬러 이미지보다 비교적 learning rate가 적절히 선택되었고 전체적으로 learning rate가 높게 설정되어 있다는 결과를 보였다.

4.3 악성코드 분류 시스템 성능 실험

설계한 악성코드 분류 시스템의 성능 평가를 위하여 정상 파일과 악성 파일 분류 실험을 진행하였다. 악성 파일은 vxheaven 데이터에서 약 23,000여개 파일을 랜덤으로 선택하였고 정상 파일은 윈도우에서 dll 및 실행파일 등을 약 1100개 정도를 추출하였다. 결과는 아래 표1과 같다.

		실제 정답	
		True	False
실험 결과	Positive	1066	138
	Negative	159	22958

[표 1] 악성코드 분류 시스템 분류 결과

V.결론

변종 등으로 악성코드의 수 증가와 지능화 등으로 악성코드 분석 및 대응하기 위한 인력의 부족하다. 이에 대한 어려움을 극복하고자 본 연구에서는 악성코드를 자동으로 정적분석하여 악성여부를 분류 해주는 시스템을 설계하였다. Hash값, PE Metadata, Packer 정보 등의 여러 가지 특성인자를 추출하고 이를 다양한 기계학습 알고리즘으로 분류한다는 점에서 기존 자동 분석 도구인 단순 시그니처 기반의 분류와 차이점이 있다. 또한 파일 자체를 이미지화하여 CNN모델에 입력한다는 점에서 PE파일뿐만 아니라 쉘과 같은 파일들도 분류가 가능하였다.

향후 단순히 악성코드 여부만 판단하는 것이 아닌 다양한 악성코드 정보를 분류할 수 있도록 실험과 연구를 통해 개선할 계획이다. 또한 위의 설계한 시스템을 활용하여 단순히 분류뿐만 아니라 실시간으로 파일을 수신 및 전송에 탐지가 가능한 시스템으로 발전시킬 예정이다.

[Acknowledgment]

이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2018R1D1A1B07047395).

[참고문헌]

[1] D. Bansal, M. Godara, "Internet Security and Privacy." Vol 1, No 11 (2014): Vol 1 Issue 11 December 2014

[2] Louis Marinos(2015, Jan 27). ENISA Threat Landscape 2014(Overview of current and emerging cyber-threats) [Online]. Available: http://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/enisa-threat-landscape-2014(download 2015, Sep. 9)

[3] Malware Validation Techniques, http://blogs.cisco.com/security/comments/malware_validation_techniques/

[4] Z. Salehi, A. Sami, M. Ghiasi, "Using feature generation from API calls for malware detection", Computer Fraud & Security, Volume 2014, Issue 9, September 2014, Pages 9-18

[5] D. Devi, S. Nandi, "PE File Features in Detection of Packed Executables" International Journal of Computer Theory and Engineering, Vol. 4, No. 3, June 2012

[6] K. Espoir Kabanga, K. Chang-Hoon, "Malware Images Classification Using Convolutional Neural Network", Journal of Computer and Communications, 2018, 6, 153-158

[7] R. Pascanu, Jack W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, "Malware classification with recurrent networks", 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 19-24 April 2015

[8] VX Heaven, "Vx heaven virus collection 2010-05-18." http://vxheaven.org/. [Online; accessed 9-November-2018].

[9] Microsoft, "Microsoft malware classication challenge (big 2015)," 2015-02-03. [Online; accessed 9-November-2018]

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

포스터발표 트랙 F

사이버 기만을 위한 사이버 기만 트리 모델*

박창욱*†, 김영갑*‡
*세종대학교 정보보호학과 보안공학연구실

Cyber Deception Tree Model for Cyber Deception

Chang-wook Park*†, Young-gab Kim*‡
*Security Engineering Lab., Dept. of Computer and Information Security, Sejong Univ

요 약

미국 사이버 작전 교범은 사이버 공격을 차단과 조작으로 구분하고 조작의 세부 항목으로 기만을 포함하고 있다[1]. 이러한 상황은 공격과 방어 위주의 사이버 작전에 기만전술이 포함된 복잡한 작전환경으로 패러다임이 변화하고 있다. 따라서 앞으로 사이버 작전은 공격과 방어작전 수행하며 기만전술이 포함될 것이며, 사이버 작전에서 기만 요소를 빠르게 식별하는 것은 상대적 작전우위를 선점할 수 있는 필수 요소로 발전할 것이다.

사이버 기만과 관련된 연구는 초기 방어자 관점의 허니팟과 같은 도구 위주의 연구였다. 그러나 공격에 관한 관심이 높아지며 전술-전기-절차(TTPs)를 기준으로 작전 관점에서 사용하는 도구를 정리하였다[2]. 하지만 사이버 공간이 물리적, 논리적, 사회적 계층으로 구분됨에 따라 작전의 대상을 기계에서 사람으로 확대하여야 하며, 복잡한 기만 작전을 계획 할 수 있는 군사 기만 기반의 계획수립 모델이 필요하다.

이에 본 연구는 사이버 기만 작전의 대상-유형-인지-전술-기술로 계획을 수립할 수 있는 사이버 기만 트리 모델을 제안한다. 그리고 제안한 모델을 검증하기 위해 14년 한수원 사이버테러와 안티 랜섬웨어 기법을 대상으로 기만 작전을 재기획하며 기만 트리 모델의 적합성을 검증하였다.

I. 서론

'14년 12월 한국수력원자력을 대상으로 기만전술이 사이버 작전에 사용되었다. 그리고 3년 후, '18년 8월 미국 사이버 작전 교범에 '기만'이 정식으로 추가되었다[1].

'기만'은 상대방이 능력 및 의도, 작전에 대해 의도적으로 혼란 및 오도하여 아측의 작전에 기여하는 구체적인 행동이다[3]. 사이버 공격에 기만이 추가된 것은 기존 방어에 사용되는 기만 기술이 공격에도 사용되며 미래 사이버 작전환경이 복잡해짐을 보여준다. 복잡한 사이버 작전환경에서 사이버 작전을 수행하기 위해서는 기만과 관련된 요소를 식별하여야 한다.

초기의 사이버 기만 기술은 방어적 관점에서 허니팟과 같은 도구 중심으로 연구되었다. 그리고 Mitre 연구소의 ATT&CK[7]처럼 공격에 관한 관심이 증가하자 사이버 기만 기술도 전술-전기-절차(TTPs)를 기준으로 구분하기 시작하였다[2].

그러나 기존 사이버 기만 TTPs는 사이버 공간의 사회적 계층인 인지적 요소는 제외하였으며, 사용하는 정보의 종류 및 방식에 따른 단순한 구분으로 '14년에 발생한 한수원 해킹사건과 같은 복잡한 기만 환경을 군사작전 수준의 TTPs로 표현하지 못한다.

이에 본 논문에서는 사이버 공간에서 군사작전 수준의 기만 요소를 식별할 수 있는 사이버

* 이 연구는 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00756, IoT 이종 식별체계 상호연동 및 관리체계 기술 개발)
† 박창욱, parkchangwook@sju.ac.kr
‡ 김영갑, alwaysgabi@sejong.ac.kr(Corresponding author)

기만 트리 모델을 제안한다.

본 논문은 2장에서 사이버기만의 정의 및 전술, 기술을 설명하고, 3장에서 사이버 기만 작전을 계획하기 위한 사이버 기만 트리를 제안한다. 그리고 4장에서는 사례분석을 통해 기만 트리의 적합성을 확인하며, 마지막 5장은 향후 연구에 관하여 기술하고자 한다.

II. 관련연구

2.1 사이버 기만의 정의

군사적 기만의 정의는 미 합참의 JP 3-13.4 군사적 기만(MILDEC, Military Deception) [3]에 따라 '상대가 군사 능력 및 의도, 작전에 대해 의도적으로 오도(Mislead)하여 아군의 군사 작전에 이바지하도록 하는 구체적인 행동'이다.

사이버 기만의 정의는 Jim Yuill[4]에 의해 군사적 기만의 정의를 기반으로 '공격자가 방어에 이바지하도록 오도하는 일련의 계획된 행동'으로 정의하며, Mohammed H. Almeshekah[5]는 혼란(Confuse)을 추가하여 '공격자가 방어에 이바지하도록 혼란을 주거나 오도하는 일련의 계획된 행동'으로 정의를 확대하였다.

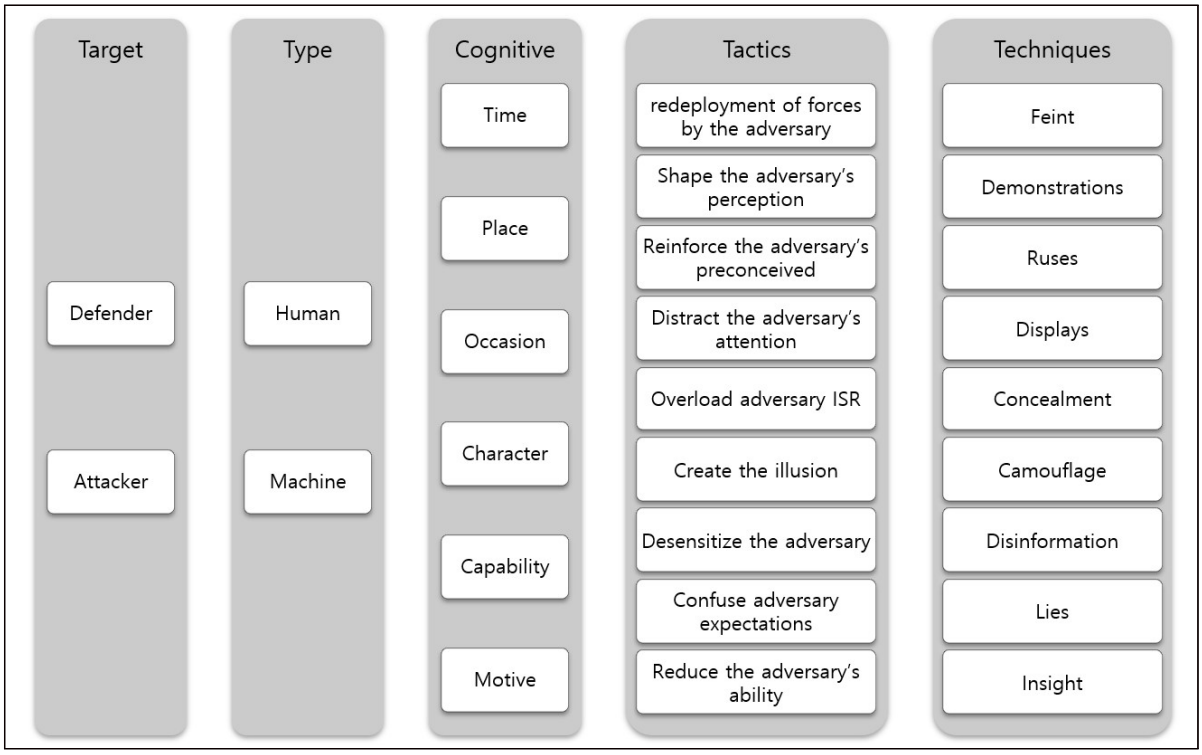
2.2 기만의 전술, 전기, 절차

기만의 전술(Tactics)은 9가지로 구성된다[3]. ① 상대방 무기, 힘의 재배치에 영향을 가한다. ② 전투에 도입되는 새로운 힘이나 무기에 대하여 상대방에게 인식시킨다. ③ 상대방의 선입견을 강화한다. ④ 상대방의 주의를 다른 활동으로부터 돌리게 한다. ⑤ 상대 ISR 수집 및 분석 기능을 과부하 시킨다. ⑥ 약한 곳에 힘의 환상을 만든다. ⑦ 우호적인 행동패턴에 대해 상대방을 무감각하게 만든다. ⑧ 적대적인 행위를 우호적인 행위로 혼란스럽게 만든다. ⑨ 상대방이 전투를 인식하고 관리하는 능력을 감소시킨다.

기만의 기술(Techniques)에 대하여 ROWE, Neil C[6]는 양공(Feints), 양동(Demonstrations), 계략(Ruses), 허식(Displays), 은폐(concealment), 위장(camouflage), 오정보(disinformation), 거짓(lies), 통찰(insights) 등 9가지로 구성하였다.

군사적 기만[3]에서 기만을 수행하는 절차(Procedure)는 보여주다(See), 생각하게 하다

Figure 1. Cyber Deception Tree Model



(Think), 행동하게 하다(Do)로 구성된다.

국사적 기만 절차와 유사하게 대부분 기만 절차는 기만을 수행하기 위한 절차를 제시하고 있다. 그러나 체계적으로 사이버 기만에 대하여 대상과 표적을 선정하고 TTPs에 따라 작전을 계획할 수 있는 모델은 부재하다.

III. 사이버 기만 트리

사이버 기만 작전을 수행하기 위해서는 작전의 대상을 선정하고, 목적을 설정하여야 한다. 그리고 목적에 도달하는 데 필요한 전술과 기술을 선택하여야 한다.

작전을 수행하는 관점에서 대상은 공격자와 방어자 모두 가능하며, 방어자가 사람이 경우와 기계인 경우로 유형을 구분하였다. 대상의 인지를 기만하기 위해 인지의 요소는 시간, 장소, 상황, 인물, 역량, 동기로 구분하였으며, 전술과 기술은 군사 기만에서 정의한 9가지 요소를 각각 선정하였다. 그리고 기만 작전을 계획순서에 따라 [그림1]과 같이 표적-유형-인지-전술-기술 순으로 사이버 기만 트리 모델을 제안한다.

IV. 사례분석

사이버 기만 트리 모델을 검증하기 위해 14년에 발생한 한수원 사이버 공격과 안티 랜섬웨어 기술을 대상으로 적용해 보았다.

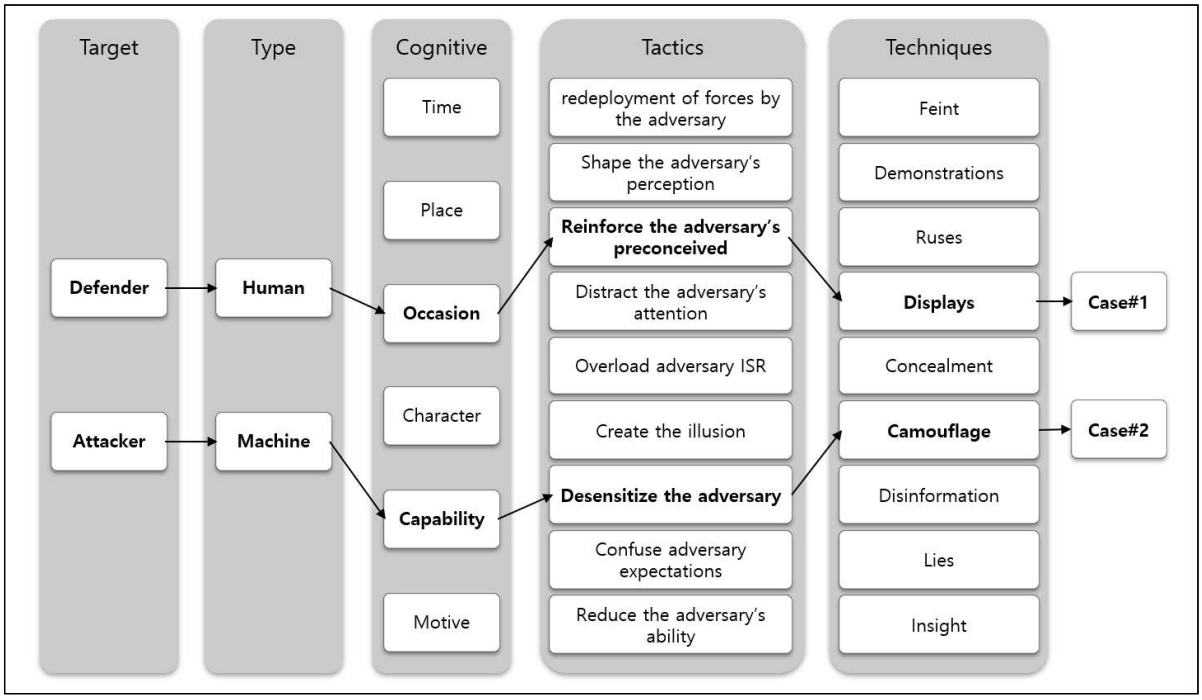
[사례 #1 – 한수원 사이버 공격]

'14년 한수원 사이버 공격의 공격자가 실시한 대표적인 기만 작전은 주요 언론사 기자들에게 유출정보를 제공한 것이다. 그리고 기만 작전의 결과로 피해당한 사실을 믿도록 만들었다. 사이버 기만의 표적과 유형은 [방어자 → 사람]이며 유형의 인지 영역은 [상황]이다. 그리고, 필요한 전술은 [상대방의 선입견 강화]이며, 사용된 기술은 [전시]이다. 사이버 기만 작전이 수행된 결과는 다수의 관계자가 해킹 사실을 믿게 되는 것이며 실제 사례분석 결과 효과적인 기만 작전으로 볼 수 있다.

[사례 #2 – 안티 랜섬웨어 기술]

랜섬웨어에 감염되면 모든 이미지, 문서에 대하여 암호화한다. 안티 랜섬웨어는 암호화하는 순간을 식별하기 위해 기만 작전의 결과로 가짜 이미지를 생성하고 감시한다. 사이버 기만의

Figure 2. Case Study of Cyber Deception Tree Model



표적과 유형은 [공격자 → 기계]이며 유형의 인지 영역은 [능력]이다. 필요한 전술은 [상대방의 무감각]이며, 사용된 기술은 [위장]이다. 사이버 기만 작전이 수행된 결과는 랜섬웨어가 안티 랜섬웨어의 위장한 가짜 이미지에 접근하는 순간 해킹 사실을 인지하게 되는 기만 작전으로 볼 수 있다.

[분석결과]

사례 #1과 사례 #2를 사이버 기만 트리 모델에 적용하면 [그림 2]와 같이 표현할 수 있다.

식별된 기만전술은 상황별로 대응책을 수립할 수 있다. 사례 #1은 정보를 전시하고 있는 SNS 계정을 신고하여 활동을 중단시킬 수 있으며, 사례 #2는 알려진 위장 파일을 식별하여 우회하는 방식으로 대응할 수 있다.

V. 결론

사이버 기만 작전은 단독으로 수행될 수 있으며, 공격과 방어작전과 동시에 수행되기도 한다. 기존 사이버 기만과 관련된 연구는 다양한 TTPs와 기만 도구를 정리하고 있으나 군사적 사이버 작전과 연계하지 못하였다.

이에 본 논문에서는 미국의 군사기만 교범과 사이버 기만과 관련된 TTPs를 참고하여 체계적인 사이버 기만 작전을 계획할 수 있는 기만 트리 모델을 제시하고, 한수원 사이버 공격과 안티 랜섬웨어를 통해 적합성을 확인하였다.

향후 연구과제로 사이버 기만 작전을 위한 인지적 효과와 무기를 정리하여 사이버 기만 트리를 확장하고자 하며, 사이버 기만 무기의 성능을 측정할 수 있는 기법을 연구하고자 한다.

[참고문헌]

[1] US Joint Chiefs of Staff, Cyberspace Operations, Joint Publication 3-12, Department of Defense, 2018

[2] STECH, Frank J.; HECKMAN, Kristin E.; STROM, Blake E. Integrating cyber-D&D into adversary modeling for active cyber defense. In: Cyber Deception. Springer, Cham, 2016. p. 1-22.

[3] US Joint Chiefs of Staff, Military DECEPTION, Joint Publication 3-13.4. Department of Defense, 2012

[4] YUILL, James Joseph, et al. Defensive computer-security deception operations: processes, principles and techniques. 2007.

[5] ALMESHEKAH, Mohammed H.; SPAFFORD, Eugene H. Planning and integrating deception into computer security defenses. In: Proceedings of the 2014 New Security Paradigms Workshop. ACM, 2014. p. 127-138.

[6] ROWE, Neil C.; ROTHSTEIN, Hy S. Two taxonomies of deception for attacks on information systems. 2004.

[7] STROM, Blake E., et al. Finding Cyber Threats with ATT&CKTM-Based Analytics. 2017.

전자적 문서의 법적 효력 향상을 위한 검증방안 연구

-보험회사를 중심으로-

허 원 석*, 임 상 헌**, 허 재 영***

*고려대학교 정보보호대학원, **동국대학교 국제정보보호대학원, ***한국교육학술정보원

Study verification method for improving legal effectiveness of electronic document. - Focused on insurance companies -

Wonseok Heo*, Sangheon Lim**, Jaeyeong Heo***

*Graduate School of Information Security Korea University, **Graduate School of International Affairs and Information Security Dongguk University, **Korea Education and Research Information Service

요 약

전자문서, 전자화 문서와 같은 전자적 문서에 대하여 법률상에서의 지위는 일반 문서와 동일하다고 명시하고 있으나, 법률에 특별히 명시되지 않은 경우에는 인정하지 않았던 과거의 판례들에 의해 기존 종이 문서에서 벗어나지 못하고 있는 실정이다. 하지만 점차 전자적 문서 효력에 대한 판례가 변화함에 따라 전자적 문서에 대한 법적 효력이 점차 인정되고 있으며, 이에 본 논문은 전자적 문서의 특징인 변조가 가능한 부분을 보완하여 전자적 문서의 법적 효력을 보다 향상시킬 수 있는 검증방안을 제시하고자 한다.

I. 서론

전자화 문서는 전자문서 및 전자거래 기본법(이하 전자문서법)에 명시된 바와 같이 종이 문서나 그 밖에 전자적 형태로 작성되지 아니한 문서를 정보처리시스템이 처리할 수 있는 형태로 변환한 문서를 일컫는다. 그리고 전자문서법 제6조 제1항에 의해 전자문서는 전자화 문서의 포함을 명시하고 있다.

전자문서와 전자화 문서를 포함한 전자적 문서에 대해서 전자문서법 제4조 제1항에서는 "다른 법률에 특별한 규정이 있는 경우를 제외하고는 전자적 형태로 되어 있다는 이유로 문서서의 효력이 부인되지 아니한다"고 명시적으로 일반적인 종이 문서와 그 효력이 동일하다고 인정하고 있다.

하지만 문서보관 규제를 받는 사업자들은 전자문서 이용에 관해 명시된 법률상의 특별한 내용이 없을 경우 전자문서 사용에 미온적이며, 그 결과 민간업체의 전자문서 활용률은 57.3%에 불과한 것으로 확인되었다.[1]

이로 인해 종이 문서 중심의 업무처리 방식이 현재에도 유지됨에 따라 연간 약 1.3조원 이상의 경제적 비용이 발생하고 있으며 이외에도 종이 문서의 생산 및 보관비용이 지속해서 증가하고 있는 실정이다.

특히 금융권의 경우 2010년 기준으로 정보인쇄용지의 소비량을 보면 전체 약 25% 수준인 75억장을 사용하였으며, 그중 금융권 업종별 사용된 문서는 은행 8억2천만 장, 보험 3억1천만 장, 증권 1억3천만 장으로 상당량의 종이 문서를 사용하고 있다.[2]

이에 본 논문은 전자적 문서에 대한 현 상황을 재고하여, 불필요한 비용을 줄이면서 나아가 전자적 문서의 법적 효력을 인정받기 위한 추가방안을 금융회사를 중심으로 제시하고자 한다.

II. 전자적 문서의 효력

본 장에서는 전자적 문서의 법률적 정의와 법적 효력을 인정받기 위해 전자적 문서가 갖추어야 하는 기술적 요건들에 대하여 설명한다.

2.1 법률상 전자적 문서의 지위

2.1.1 전자문서 및 전자거래 기본법

전자문서법은 1999년 2월에 제정이 되었으며, "정보처리시스템에 의하여 전자적 형태로 작성, 송신, 수신 또는 저장을 위하여 이용되는 정보처리능력을 가진 전자적 장치 또는 체계"로 전자문서를 정의하고 있다. 이 경우 전자문서에 대한 해석은 전자적 데이터베이스를 의미하며, 입출력 매체의 형태, 저장, 표현 방식이 모두 전자적일 경우에만 해당한다.

전자화 문서는 동 법 제5조 제2항에서 "종이 문서나 그 밖에 전자적 형태로 작성되지 아니한 문서를 정보처리시스템이 처리할 수 있는 형태로 변환한 문서"로 정의하고 있으며, 이는 종이 문서와 같은 일반 문서를 전자적 매체를 통해 전자적 형태로 변환하였을 때 산출되는 형태를 말한다.

이러한 전자문서, 전자화 문서가 공인전자문서보관센터에 보관이 되어야만 동 법 제5조 제1항 또는 제2항에 따른 전자문서의 보관이 행하여진 것으로 보게 되며, 이는 문서의 효력을 인정받을 수 있음을 의미한다.

2.1.2 전자정부법

전자정부법은 2001년 전자정부를 구축하기 위해 제정된 법률이다. 전자정부법의 적용 범위는 위 전자문서법과 그 대상부터 다르다. 전자문서법의 경우 적용 범위의 대상에 대한 제한을 두고 있지 않으나 전자정부법에서는 행정기관 및 공공기관에 대해서만 그 적용 범위를 한정하여 명시하고 있다.

전자정부법 제26조 제3항에서는 전자적 문서의 효력에 대한 범위를 특별한 규정이 없는 한 종이 문서와 동일하다고 명확하게 명시하고 있다. 이는 법적 효력이 부여됨을 정확히 표현한 것으로 주민등록등본, 가족관계증명서 등 행정기관에서 발급하는 전자적 문서를 말한다. 주민등록등본의 경우 출력 시에는 종이 문서로 출력이 되지만 전자문서가 먼저 생성이 되고 이를 종이로 변환하여 출력되는 형태이므로 종이 문서가 아닌 전자문서의 형태로 봐야한다.

2.2 공인전자문서보관센터

공인전자문서보관센터(또는 공인전자문서보관소)는 타인을 위하여 전자문서를 안전하게 보관하고 이를 증명하거나 관련된 업무를 수행할 수 있도록 정부의 지정을 받은 제3의 기관을 말하며 신뢰성 있는 제 3자인 보관소[3]가 전자문서의 법적인 효력을 보장함으로써 종이 문서를 대체·폐기하고 전자문서의 이용을 촉진하고자 하는 취지에서 도입되었다.

공인전자문서보관센터는 전자문서 보관의 대행, 보관내역에 대한 증명서 발급, 전자문서의 내용 및 송·수신 여부 증명 등 전자문서와 관련된 다양한 업무를 수행하며[4] 보관 중인 전자문서에 대해서는 법적인 효력과 법률상 불변경 추정의 효력이 인정되고 보관소 발급 증명서에 대해서는 진정성 추정력이 부여된다.

이는 전자문서 및 전자거래 기본법에서 명시된 내용과 같이 전자문서가 공인전자문서보관센터에 보관될 경우 원본과 동일한 효력을 갖출 수 있게 된다. 다만, 공인전자문서보관센터에 보관된 전자문서가 법적 효력을 지니기 위해서는 적절한 절차에 따른 저장방식이 요구되는데 이는 다음과 같은 공인전자문서보관센터의 보안 요구사항을 따른다.

첫 번째, 전자적 문서의 무결성이 입증되어야 한다. 무결성 입증은 일반적으로 일방향 해시 알고리즘(Hash), 메시지 인증코드(MAC), 순환

중복코드(CRC)를 사용한다. 이 무결성 방안으로 전자적 문서의 변조 여부를 판단할 수 있으므로 생성 시의 전자적 문서가 변하지 않고 유지되고 있음을 증명할 수 있다.

두 번째, 전자적 문서의 검증이 가능해야 한다. 공인전자문서보관센터에 저장되는 전자적 문서는 WORM(Write Only Read Many) 방식으로 저장이 되며, 그 이유는 파일 덮어쓰기가 차단되어야만 변조가 방지되기 때문이다. 따라서 WORM 디스크에 저장된 전자적 문서는 저장된 시점 이후 변경되지 않음을 입증할 수 있다.

세 번째, 전자적 문서의 부인방지가 가능해야 한다. 전자문서가 전송될 때 전송자의 공인인증서로 전자서명(Digital Signature), 타임스탬프(Time-stamp) 등을 적용하여 전자문서의 출처를 입증하여야 한다.

하지만 전자문서와 전자화 문서가 문서로써 효력을 가질 수 있는 조건은 상이하다. 전자문서는 그 자체로 문서의 효력을 인정받을 수 있으나, 전자화 문서는 원본 문서에서 전자적 형태로 변환된 것으로 다음의 사항이 준수되었을 때 문서로써의 효력이 인정된다.

첫 번째, 원본 문서와 전자화 문서가 문서적 형태만 다를 뿐 양식이나 내용 등이 일치해야 한다.

두 번째, 전자화 문서로 변환되는 방법 및 절차가 항상 동일해야 하고 그 내용 및 형태가 일치해야 한다는 점이다. 그래야 동일성이 인정되므로 효력도 동일하게 볼 수 있다는 것이다.

세 번째, 전자화 문서로 변환되는 방법 및 절차에 문제가 없어야 한다. 전자문서는 입력 시점부터 전자적인 매체로 입력이 되는 반면, 전자화 문서는 다른 형태의 문서가 전자적 문서로 변환이 되기 때문에 그 변환 과정에서 내용이 변할 가능성이 있으므로 무결성이 유지된다는 것이 입증되는 방법과 절차여야 효력도 인정된다고 보는 것이다.

III. 현행 전자적 문서의 한계

본 장에서는 현재의 보험회사 전자청약 프로세스(보험계약을 전자화하여 계약 내용 이행 및 내용을 보존하기 위한 청약 프로세스) 상의 전자적 문서에 대한 법적 효력의 한계성을 갖는 이유에 대하여 설명한다.

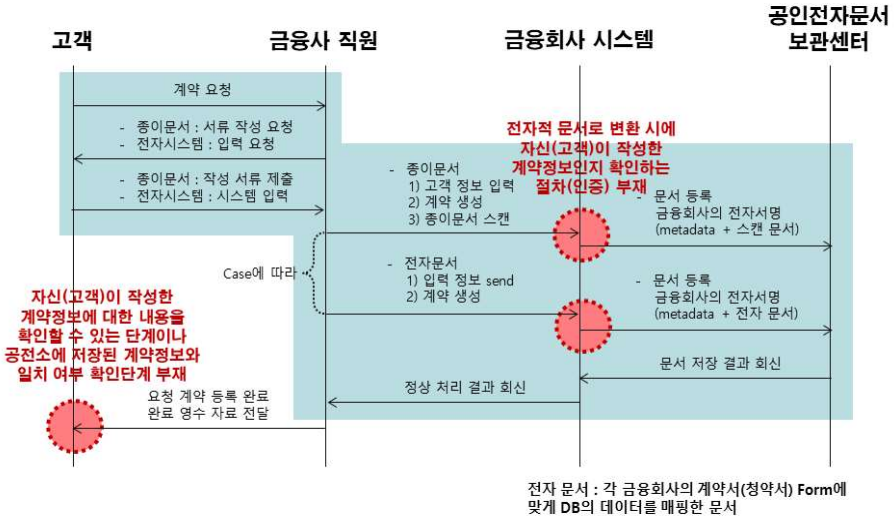
3.1 법규의 불명확성

전자문서 및 전자거래 기본법은 위에서 언급한 바와 같이 1999년에 제정되었다. 우리나라에서 전자정부 지원사업이 추진된 것은 2001년 전자정부의 구축을 위해 2001~2003년에 설립, 운영되었던 전자정부특별위원회가 그 시초이다.[5] 그리고 2001년 3월에 전자정부 구현을 위한 행정업무 등의 전자화 촉진에 관한 법률(현 전자정부법)이 제정되었다. 이처럼 전자문서, 전자화 문서에 대한 시대적 도입이 빨랐던 만큼 발전 속도도 빨랐으며, 현재 대한민국은 UN에서 발표한 전자정부 순위에서 2010년부터 현재까지 1위를 기록하고 있다.[6] 그리고 u-Paperless와 같은 국제 컨퍼런스를 주관하는 등 전자적 문서 활성화에 대해 많은 노력을 하고 있다.

이처럼 전자정부법을 근간으로 우리나라는 전자적 문서에 대한 이해도가 높은 것은 사실이나 제2장에서 언급한 바와 같이 문서로써의 법적 효력이 공공과 민간에 따라 다르게 적용되므로(즉, 법원에 제출이 가능한 문서인 법적 효력 부여가 다르기 때문에) 실제 민간기업의 경우 전자문서 또는 전자화 문서의 보관만으로는 원본파기에 대한 부정적 견해를 가질 수밖에 없다.[7]

3.2 정보제공자가 제공한 정보의 무결성 확인 절차 미흡

정보제공자인 고객이 종이 문서를 작성하거나 금융회사의 정보처리시스템에 정보를 입력하는 경우 고객의 인증 절차가 존재하나 청약서를 작성하는 시점의 계약 정보들이 금융회사



[그림 1] 보험회사의 일반적인 전자청약 프로세스

의 기간계 시스템 및 공인전자문서보관센터에 저장된 정보와 일치함을 입증할 수 있는 절차가 미흡하다. 즉 전자적 문서 형태로 보관하기 위한 전자금융 프로세스(전자청약 프로세스)에서 최초 계약정보에 대한 전송, 처리, 저장 과정의 무결성을 보장하는 절차가 부족하다는 의미이다.

이에 금융회사는 대체 통제로 계약 후에 고객에게 내용을 알리고 보험증권을 전달하고 있으나 이는 계약정보에 대한 무결성을 입증할 수 있는 절차가 아닌 계약의 이상 여부만을 확인할 수 있는 사후 통제이므로 무결성을 입증할 수 있는 절차가 부족하다고 할 수 있다. (그림 1 참고)

IV. 방안 제시

제안하는 전자적 문서의 법적 효력을 부여하는 기술적 방안은 전자적 문서의 생성 과정에서 요구되는 서명 영역을 캡쳐하여 이미지 파일화(File A) 한 후 이미지 파일의 해시값(Hash A)을 생성하고 생성된 해시값을 이용하는 방법이다.

제안하는 방식은 공인전자문서보관센터에 전자적 문서를 보관하기 위해서는 전자적 문서 위탁자의 공인인증서로 전자서명을 하는 단계에서 전자적 문서에 대한 해시값(Hash B)와 앞서 생성한 이미지 파일의 해시값(Hash A)를 XOR 연산을 하여 해시값(Hash C)를 생성하고 생성된 해시값(Hash C)과 이미지 파일(File A)를 기존 공인전자문서보관센터에 저장하려는 값들과 함께 저장하는 것이다.

해시값을 이용하는 방법은 디지털포렌식에서 디지털증거의 무결성을 입증하기 위해 증거물에 대한 고유한 값을 부여하는 것과 유사한 원리이며, 해시값을 이용할 경우 초기에 고객이 작성한 계약문서(종이 문서 또는 전자문서)가 금융회사 시스템 및 공인전자문서 보관센터에 저장된 전자적 문서와 동일함을 입증할 수 있게 된다.

그 이유는 생성된 전자적 문서를 대상으로 생성한 고객 서명 이미지 파일(File A)와 그 이미지의 해시값(Hash A)을 계약의 마지막 단계에서 고객에게 제공하여 최종적으로 고객이 직접 작성한 계약서임을 확인하는 단계가 포함되어 있으며, 이 단계에서 고객이 자신이 작성한 계약 내용이 맞음을 고객 소유의 또는 고객 계정의 전화/이메일/SNS 메신저 등의 방식을 통해 확인받게 되고 확인받은 최종시간을 계약 수락 시간으로 업데이트함으로써(Time-stamp 이용) 디지털 데이터에 대한 진정성과 신뢰성이

성립된다. 또한 이 단계에서 작성한 청약서의 내용을 고객이 검토한 후 계약 내용이 맞음을 확인하므로 원본성도 성립된다고 할 수 있다.

결론적으로 해당 방식은 무결성, 진정성, 신뢰성, 원본성이 보장되므로 디지털 증거의 법적 효력(증거능력)을 인정받기 위한 요건을 충족한다고 할 수 있어 법적 효력을 부여받을 수 있는 방식이라 할 수 있다.(그림 2 참고)

V. 결론 및 향후 연구과제

본 논문에서 제안한 기술적 방법은 새로운 기술이 아니라 기존 정보보호 및 디지털포렌식 분야에서 많이 사용되고 발전된 기술이다. 따라서 언급한 기술 및 방식에 대한 완성도는 높다고 할 수 있으나 제안 방안이 실제 금융회사에 적용될 때 얼마나 효율적인지와 적용하였을 시 관련 부서의 업무 효율 향상도 등에 대한 구체적인 측정값이 부족하다.

하지만, 최근 금융권에서 전자문서만을 이용하고자 하는 업계의 흐름과 점진적으로 전자문서의 원본성을 입증하기 위한 노력이 계속되고 있어 이와 관련된 지속적인 연구와 긍정적인 관련 판례가 더해진다면, 전자문서에 대한 활용도가 더욱 높아질 것으로 생각된다.

[참고문헌]

[1] 미래창조과학부, 전자문서법 해설서, 2017

[2] 생명보험협회, 생명보험협회보 통권43호 (상법 개정안에 대한 생보업계 의견), 2018

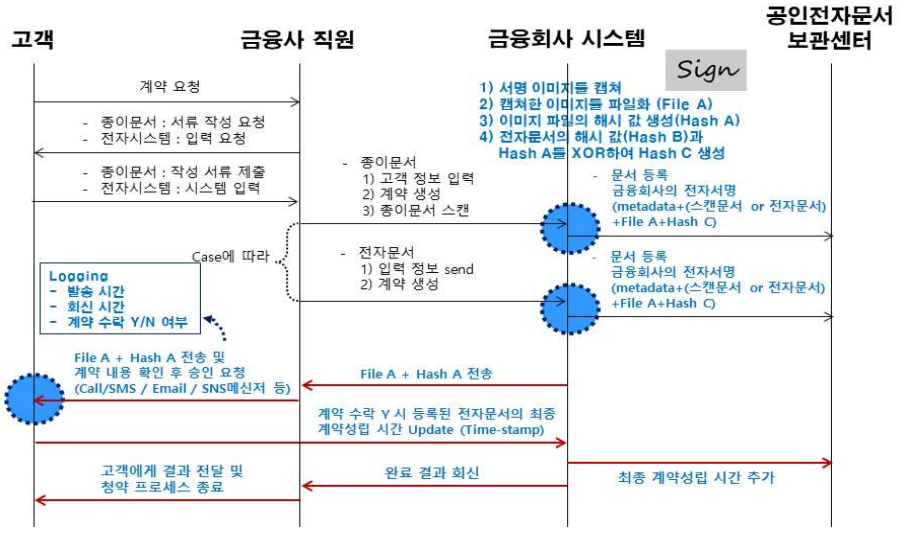
[3] 정보통신산업진흥원, 공이전자문서보관서 해설가이드, 2010

[4] 한국은행 금융결제국, 금융권의 공인전자문서보관소 이용 현황 및 제도적 문제점, 2010

[5] 한국민족문화대백과사전, http://encykorea.aks.ac.kr

[6] UN경제사회처 공공행정국, http://www.index.go.kr/potal/main/EachDtlPageDetail.do?idx_cd=1027

[7] 서울시립대학교 법학연구소, 금융기관 문서의 안전한 유통과 보관 방안에 관한 연구, 제21권 제3호



[그림 2] 제안하는 전자청약 프로세스

중복된 난수 사용을 통한 EC-KCDSA 알고리즘 개인키 추출 방안

고주성*, 곽진**

*아주대학교 컴퓨터공학과 정보보호응용및보증연구실

**아주대학교 사이버보안학과

Extraction a Private Key on EC-KCDSA Algorithm with Duplicated Random Nonce

Ju-Seong Ko*, Jin Kwak**

*ISAA Lab., Department of Computer Engineering, Ajou University.

**Department of Cyber Security, Ajou University.

요 약

전자 서명은 온라인 상에서 신원 인증 및 무결성 확인을 위해 사용될 수 있다. 대표적으로는 공개키 기반 구조의 공인인증서로 사용되어 왔으며, 최근 블록체인의 발달로 거래 정보의 무결성을 보장하기 위해 사용되고 있다. 암호 알고리즘들은 이론상으로는 안전하나 프로그램 구현 시 취약해질 수 있다고 평가되며, 이는 전자 서명 알고리즘에도 적용된다. 전자 서명 알고리즘의 경우 구현을 위해 여러 암호학적 요소가 수반되어야 한다. 난수 생성, 해시 알고리즘은 전자 서명의 안전성에 큰 영향을 미치며, 안전하지 않다고 평가되는 알고리즘 사용 시 전자 서명 알고리즘의 안전성을 저하시킬 수 있다. 따라서 본 논문에서는 국산 타원 곡선 기반 전자 서명 알고리즘인 EC-KCDSA에 대해 중복된 난수를 사용한 경우 개인키를 추출하는 과정을 분석하며 안전하다고 평가되는 난수 생성, 해시 알고리즘 사용의 중요성을 강조한다.

I. 서론

전자 서명은 데이터의 출처와 무결성을 확인할 수 있도록 데이터에 첨부되는 값으로 수신자가 전자 서명 검증을 통해 데이터 위조 확인 및 부인 방지 등의 서비스를 제공받을 수 있다[1]. 대표적인 국산 전자 서명 알고리즘으로는 KCDSA, EC-KCDSA가 있으며 EC-KCDSA는 타원 곡선 상에서 KCDSA를 구현한 것이다[2, 3].

KCDSA와 EC-KCDSA는 국산 암호 활성화를 위해 국내의 신뢰받는 기관을 통해 개발 및 검토되었으며, 현재까지 알려진 최선의 공격 방법에 기반하여 공격자로 데이터를 보호할 수 있는 수준으로 평가된다[2, 3].

하지만 암호 알고리즘은 이론상 안전하더라도 실제 적용 시 취약해질 수 있다. 보안에 취약한 난수 사용은 암호 알고리즘의 안전성 저하를 초래할 수 있다는 것이 밝혀졌으며[4], 이는 전자 서명 알고리즘에도 적용될 수 있다.

EC-KCDSA의 서명 생성 준비 과정을 살펴보면 키와 난수 값을 생성하기 위해 난수 생성 및 해시 알고리즘을 사용하고 있다[2, 3]. 이는 안전하지 않은 난수 생성 및 해시 알고리즘 사용으로 EC-KCDSA의 안전성이 저하될 수 있음을 보여준다.

따라서 본 논문의 2장에서는 국산 전자 서명, 난수 생성 및 해시 알고리즘에 대해 설명한다. 3장에서는 중복된 난수 사용 시 발생할 수 있는 문제에 대해 분석하며, 4장에서는 분석한 문제를 기반으로 서명 값으로부터 개인키를 추출하는 과정을 보여준다. 5장에서는 개인키 추출 방지를 위해 권고되는 암호 알고리즘을 사용해야 함을 강조하며 결론을 맺는다.

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00304)

II. 관련 연구

2.1 KCDSA

KCDSA(Korean Certificate-based Digital Signature Algorithm)는 임의의 길이를 갖는 메시지에 대해 전자 서명을 생성 및 검증할 수 있게 해주는 한국형 인증서 기반 전자 서명 알고리즘으로 인증, 무결성 및 부인 방지 등의 정보보호 서비스를 제공한다[2].

2.2 EC-KCDSA

EC-KCDSA(Korean Certificate-based Digital Signature Algorithm using Elliptic Curves)는 한국형 인증서 기반 타원 곡선 전자 서명 알고리즘으로, KCDSA와 동일한 정보보호 서비스를 제공한다[3]. EC-KCDSA의 서명 생성 과정은 (표 1)과 같으며 개인키는 d로 표기한다.

(표 1) EC-KCDSA 서명 생성 과정

순서	과정
1	난수 k를 임의로 선택 $(k < n-1)$
2	$kG = (x_1, y_1)$
3	$r = Hash(x_1)$
4	$c_Q = MSB(x_Q \| y_Q, L)$
5	$v = Hash(C_Q \| M)$
6	$e = (r \oplus v) \bmod n$
7	$s = d(k-e) \bmod n, (s \neq 0)$
8	(r, s) 반환

2.3 난수 생성 및 해시 알고리즘

불안전한 난수 생성 알고리즘 사용 시 개인정보 유출 등의 해킹사고가 발생할 수 있다. 대표적으로는 RC4가 있으며, 해당 알고리즘은 사용하지 않는 것이 권고되고 있다[5].

안전한 난수 생성 알고리즘으로 난수를 생성하더라도 이를 사용 전 재가공하는 해시 알고리즘이 취약할 경우 문제가 발생할 수 있다. SHA-1의 경우 다른 입력에 대해 동일한 출력이 나오는 문제가 발생할 수 있으며[6], 이는 중복된 난수 생성을 사용하는 것과 동일한 결과를 초래할 수 있다.

III. 문제점 분석

EC-KCDSA는 개인키를 이용해 공개키를 생성하며, 이 공개키는 누구나 검증할 수 있도록 공개되어 있다. 따라서 여러 전자 서명들의 공개키가 같은 경우, 서명을 생성하기 위해 이용된 개인키가 동일함을 알 수 있다. 3.1절에서는 EC-KCDSA 알고리즘의 서명 생성의 마지막 단계를 나타내며, 3.2절은 중복된 난수 사용으로 발생할 수 있는 문제를 분석한다.

3.1 EC-KCDSA의 서명 생성 과정

EC-KCDSA를 이용한 전자 서명은 서명 값 r, s를 생성하며, 생성 과정은 다음과 같다.

- $r = Hash(x_1), kG = (x_1, y_1)$
- $s = d(k-e) \bmod n$

3.2 발생할 수 있는 문제 분석

① 서명 값 r이 같으면 동일한 난수를 사용함

전자 서명의 서명 생성 과정으로부터 난수 k가 서명 값 r 생성에 관여하고 있음을 알 수 있다. 난수 k 외의 값들은 정의한 파라미터에 따라 고정된 값이기 때문에 난수 k의 변화가 서명 값 r의 변화에 영향을 미치고 있다. 즉, 난수 k가 변경되지 않으면 서명 값 r도 동일하다.

② 난수 중복 사용 시 개인키 추출 가능

서명 값 s 생성 과정은 3.1절에서 볼 수 있다. 이 과정을 동일한 개인키에 대해 중복된 난수 k를 사용하여 두 개의 EC-KCDSA 전자 서명을 생성한 경우 다음과 같이 나타낼 수 있다.

- $s_1 = d(k-e_1) \bmod n,\ s_2 = d(k-e_2) \bmod n$

두 개의 식을 뺀 후 정리하면 다음과 같이 나타낼 수 있다.

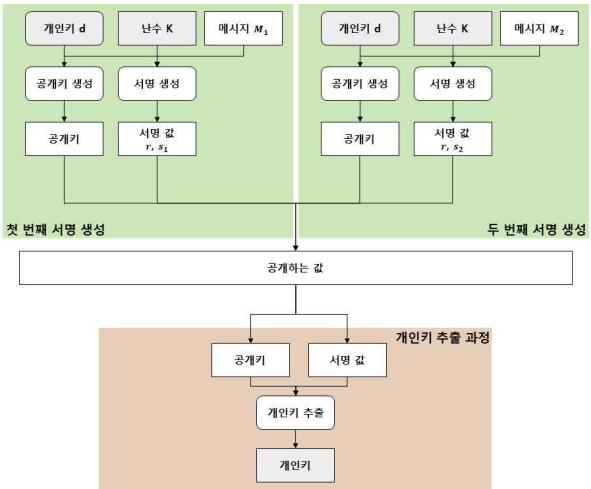
- $s_1 - s_2 = d(e_2 - e_1) \bmod n$
- $d = (s_1 - s_2)(e_2 - e_1)^{-1} \bmod n$

위의 정보들로부터 EC-KCDSA 전자 서명 알고리즘에 대해 중복된 난수 k가 사용되는 경우, 개인키 추출이 가능함을 확인할 수 있다.

IV. 전자 서명 개인키 추출

4.1절에서는 [3]에서 제시한 EC-KCDSA(secp 224r)/SHA-256 환경의 전자 서명 알고리즘을 기반으로, 서명 값으로부터 개인키를 추출하는 과정을 보여준다. 4.2절에서는 실제 값을 통해 개인키가 추출될 수 있음을 보인다.

4.1 개인키 추출 과정



(그림 1) 전자 서명 생성 및 키 추출 과정

(그림 1)은 EC-KCDSA 서명 생성과 키 추출 과정을 요약한 것이다.

① 서명 생성 과정

(그림 1)의 두 서명 생성 과정에서 메시지 M, 서명 값 s만 다르다. 이는 동일한 개인키를 이용하기 때문에 동일한 공개키를 가지며, 중복된 난수를 사용하여 동일한 서명 값 r을 가지기 때문이다.

② 개인키 추출 과정

개인키 추출 과정에서 얻을 수 있는 정보는 메시지, 공개키 쌍, 두 개의 서명 값 쌍이며, 필요한 정보는 중간 값 e이다. 따라서 얻은 정보를 통해 e를 구하기 위해 (표 1)의 4, 5, 6 과정을 거친다.

- $d' = (s_1 - s_2)(e_2 - e_1)^{-1} \bmod n$

필요한 정보가 계산되면 위의 식을 통해 개인키 d를 계산한다.

4.2 EC-KCDSA 개인키 추출

(표 2)는 EC-KCDSA 서명 생성에 사용된 값들을, (표 3)은 개인키 추출에 사용된 값들을 정리한 것이다.

(표 2) EC-KCDSA 서명 생성 과정

과정	정보	실제 값(Hex)
첫 번째 서명 생성	메시지	4E 65 73 73 61 67 65 5F 66 6F 72 5F 57 68 6F 49 73 5F 32 30 31 38 5F 6D 69 6E 69 5F 43 54 46
	난수 k	61 6E 64 20 43 72 79 70 74 6F 67 72 61 70 68 79 2D 57 69 6E 74 65 72
	개인키 d	43 30 6D 70 49 65 74 65 20 33 43 4B 43 44 53 41 20 50 72 6F 62 6C 65 6D 6D 6D 6D 21
	공개키 X_q	AC 3D D2 3F DC 20 1D 54 2B 7D 22 B8 C7 28 D9 14 F1 1E 84 27 50 48 C1 69 E7 A9 2C 8B
	공개키 Y_q	C5 9F AE BB 19 35 62 93 94 C2 10 1B FF 0F FA A7 0E 4E 1F 5A 5A 8C 25 31 0A B6 68 DD
	서명 값 r	D1 CD E0 4F 8B CA 2A 9E 11 59 EE 51 BE 22 20 B9 E0 11 AF 09 51 2A 33 4E B4 A3 44 A2
	서명 값 s_1	51 CA 2E 9F 17 72 C8 BE 65 80 BE 95 7B 9A 3B 5F 92 6B 24 68 E7 9A 53 F3
두 번째 서명 생성	메시지	65 63 2D 6B 63 64 73 61 20 64 69 67 69 74 61 6C 20 73 69 67 6E 61 74 75 72 65
	난수 k	첫 번째와 동일
	개인키 d	
	공개키 X_q	
	공개키 Y_q	
	서명 값 r	
	서명 값 s_2	9C 8B E5 28 CB A4 33 98 E8 CC A0 A3 7F 78 46 B3 A8 12 C7 59 D3 9E C3 59 CC 86 FE C7

(표 3) EC-KCDSA 개인키 추출 과정

과정	정보	실제 값(Hex)
개인키 추출	모듈러 n	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 16 A2 E0 B8 F0 3E 13 DD 29 45 5C 5C 2A 3D
	$s_1'-s_2'$	70 BF AE D2 86 25 FB 06 2E A6 28 1A B6 07 8E 84 B4 40 64 43 D2 A9 8A 54 77 6F 7F 69
	$e_2'-e_1'$	A5 E6 B0 42 58 96 4D 2A 1C A0 DE 3A 19 6A A2 9B 5C E9 FB 5F 2B BE E7 BD AE 33 57 19
	$\mathrm{inv}(e_1'-e_2')$	3D 08 07 74 8D 8A A9 CF 5A EE 91 1D 66 37 5B 3E FC BA 6B 75 61 9E B7 43 B3 FB F0 09
	d'	**43 30 6D 70 49 65 74 65 20 33 43 4B 43 44 53 41 20 50 72 6F 62 6C 65 6D 6D 6D 6D 21**

(표 2)의 개인키 d와 (표 3)의 d'을 비교한 결과 동일한 것을 확인할 수 있으며, 이는 개인키를 성공적으로 추출한 것이다.

V. 결론

본 논문에서는 전자 서명을 사용하기 위해 부가적으로 필요한 암호학적 요소로부터 취약점이 발생할 수 있음을 언급하였다. 이는 전자 서명 알고리즘이 안전하더라도 프로그램 구현 시에 취약점이 발생할 수 있음을 의미한다.

동일한 개인키를 통해 중복된 난수 값을 사용하는 경우 개인키를 추출할 수 있는 방안에 대해 분석하였다. 실생활에서 공인인증서의 재발급 기간이 짧지 않기 때문에, 난수 재사용으로 인한 위협이 발생할 수 있다.

따라서 난수 생성 및 해시 알고리즘 등 난수 사용에 영향을 미치는 알고리즘들은 취약하지 않은 버전을 사용 중인지 확인해야 한다. 사용 중인 알고리즘의 취약점이 알려진 경우, 빠른 시일 내에 취약하지 않은 알고리즘으로 대체해야 한다.

[참고문헌]

[1] 한국정보통신기술협회, "전자 서명, digital signature", 한국정보통신기술협회 정보통신 용어사전

[2] 한국정보통신기술협회, "부가형 전자 서명 방식 표준 - 제2부: 한국형 인증서 기반 전자 서명 알고리즘(KCDSA)", 정보통신단체 표준, 2016.12.

[3] 한국정보통신기술협회, "부가형 전자 서명 방식 표준 - 제3부: 한국형 인증서 기반 타원 곡선 전자 서명 알고리즘(EC-KCDSA)", 정보통신단체표준, 2016.12.

[4] KISA 침해사고대응단 종합상황대응팀, "구글, 안드로이드 취약점으로 인한 비트코인 탈취 인정", KISA 인터넷 보호나라&Krcert 자료실 게시판, 2013.08.

[5] 서울대학교 정보화본부, "유무선 공유기 보안가이드", 서울대학교, 2018.05.

[6] CWI Amsterdam, Google Research, "The first collision for full SHA-1", https://shattered.io/, 2017.01.

Performance Analysis of IPsec in Kernel-land and User-land with multi-core implementation

Sami Ullah, Joontae Choi and Heekuck Oh

Department of Computer Science and Engineering, Hanyang University.

Abstract

In Linux kernel networking, packet transmit/receive path is a layered architecture. Layered packet processing involve copy operations, which increase latency and consequently decrease the throughput of network by not fully utilizing the network capacity. In contrary to kernel networking, data plane solutions like DPDK provides direct access to packets from NIC, in user-land bypassing kernel stack, with zero intermediate copy operations. For normal packets, DPDK claim 10x improvement in throughput over kernel networking. Being inspired by remarkable efficiency, we have analyzed bottlenecks in strongswan, an open-source IPsec IKE implementation, and developed data plane based solution for strongswan using DPDK. In our architecture, we have also incorporated the multi-core design in encryption/decryption modules and analyzed the performance in terms of throughput and latency. We have achieved upto 2.56x improvement in throughput while using 4 cores.

I. Introduction

IPsec is used as an end-to-end packet encryption protocol that ensures the confidentially of transmitted data. The most widely used open-source implementation in Linux is strongswan [1]. Strongswan is built over kernel socket and all the IPsec packet processing is done in single core model[3].

Recently, a new architecture DPDK is designed by Intel, which bypasses kernel in network packet processing. DPDK is a set of data plane libraries which are designed for fast packet processing in user-land[2]. It provides a performance upgrade of 10x over conventional kernel stacked approach.

In this paper, we have analysed the bottlenecks in strongswan for IPsec implementation in kernel-land and proposed an equivalent data plane solution in user-land. Originally, strongswan IPsec processing module a is single core implementation but we have designed parallel solution by incorporating multi-core processing and also analysed the effect of the number of cores on throughput. Our contribution in this area is as follows.

1. We have implement strongswan with DPDK API, that ensure all packet processing in user-land and zero copy operation in all the network packet processing.
2. Performance analysis of strongswan and DPDK-strongswan in terms of throughput and latency.
3. Effect of number of cores on throughput and latency.

The rest of the paper is organised into following sections. 2) Background 3) Proposed design 4) Test cases 5) Results and analysis 6) Conclusion.

* This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2014-1-00636) supervised by the IITP(Institute for Information & communications Technology Promotion)

* This work was supported under the framework of international cooperation program managed by the National Research Foundation of Korea(NRF-2017K2A9A2A06016147, FY2018)

Author: sami.ullah2316@gmail.com

Coauthor: joontaechoi@hanyang.ac.kr

II. Background

Strongswan is an open-source implementation of IPsec, with packet processing based on kernel socket. Kernel networking reads raw packets from NIC, which are converted to socket packets through a layered architecture involving many copy operations[1]. These copy operations enhance latency and degrade overall throughput. The root cause of this low performance is kernel networking, which is a bottleneck for all socket-based applications. The detailed modular design of strongswan with the packet flow is shown in figure 1. To understand and identify bottlenecks in strongswan design, we divide it into two major parts (1) network packet processor (2) IPsec packet processor.

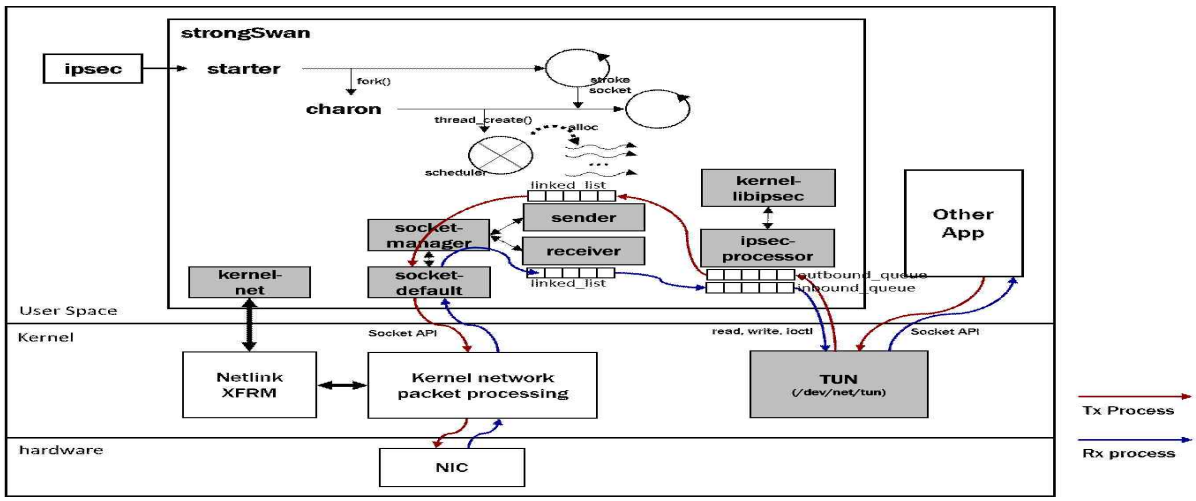


Figure 1: strongswan architecture

Network packet processor job is to prepare the raw packets from NIC and deliver it to the IPsec packet processor module and vice versa. In kernel layer, "kernel network packet processing" module prepares the raw packet and involve lots of copy operations due to kernel's layered architecture. In the user space, strongswan socket receive/transmit packets from kernel and routes it to IPsec processor module. Specifically network packet processing module contain lots of copy operation per packet and is a bottleneck in fast packet processing.

The second part of design is IPsec processor module and its basic task it to manage IKE, security policies, more importantly packet encryption/decryption and finally packets are routed to TUN device, so that the processed packets are available to other applications. One bottleneck in IPsec processor module is TUN device, which involve copy operation and generally TUN device is comparatively much slow than NIC. Second bottleneck contributing to low performance is single core implementation of kernel-libipsec module under the IPsec processor module.

So in overall design, we identified bottlenecks which involve excessive copy operations; one is kernel packet processing and other is TUN device, and single core implantation of kernel-libipsec module. In our proposed design we have addressed all above mentioned bottlenecks and analyzed their individual effect on performance.

III. Proposed design

In our proposed design, we have addressed the multiple copy operations and single core implementation issues. Our proposed model is shown in figure 2, yellow boxes are our

contribution. In the network packet handler module, we have used DPDK (Data Plane Development Kit) by Intel, which bypasses kernel layer and gives direct access to raw packets in user space with zero copy operation, and implemented DPDK-socket. In our proposed design, DPDK-socket can read/write packets directly in NIC and forwards it to IPsec processor module and vice versa.

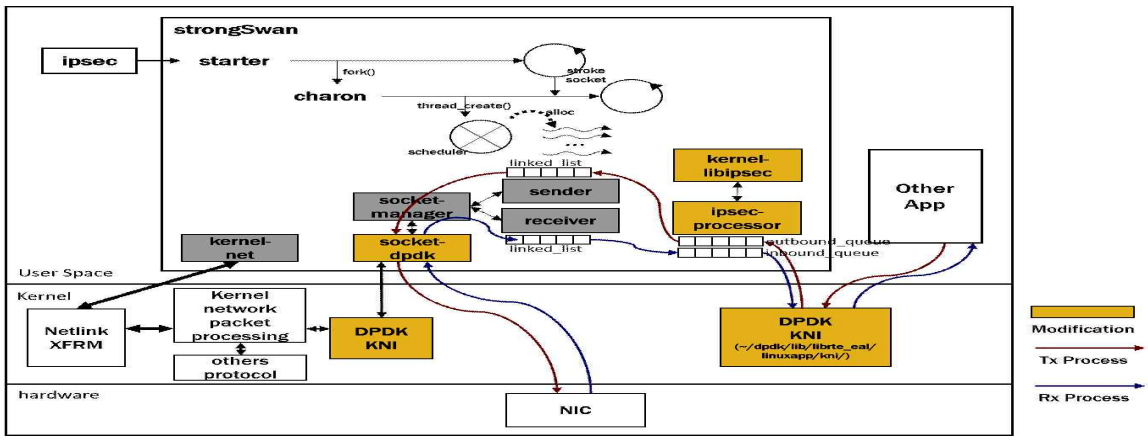


Figure 2: Data plane strongswan architecture

In the processor module, for the copy operations the bottleneck is TUN device. Fortunately DPDK has "libkni" library, which implements KNI device with zero-copy and it performs much better than TUN device. In our design we have implemented routing and packets read/write to kernel via KNI device. The other bottleneck in processor module is single core implementation for packets encryption/decryption tasks. In our design, we have implemented multi-core design for kernel-libipsec module using DPDK and achieved better results.

IV. Test cases

In this section, we develop the test case scenarios and lay the foundation for performance analysis of each module. In the problem identification section, we have highlighted three modules which are the bottleneck for strongswan performance. 1) kernel network packet processing 2) TUN device 3) kernel-libipsec (single core). To prove our design performance over strongswan, we will test the following test-cases:

1. Original strongswan
2. DPDK-socket module and TUN
3. DPDK-socket and KNI
4. DPDK-socket, KNI, and multi-cores

V. Results and Analysis

In this section, we will do the performance analysis of our test case scenarios. The hardware used for testing are two machines with intel i7-6700 CPU and X520-DA2 NIC. The software versions we used are DPDK 18.02, strongswan 5.6.2 on Ubuntu 16.04 LTS. The performance measuring parameters are throughput and latency. We have used Linux iperf and Linux sockperf to measure throughput and latency respectively.

Figure 3, represents the bar graph of throughput for all test cases. There is an increase of 1.18x, when we applied only DPDK-socket i.e. only changed the network packet processing part and kept the TUN device. But when we adopt DPDK-socket alongwith KNI device, the throughput increases to 1.4x.

Now for the multi-core scenario, there is a

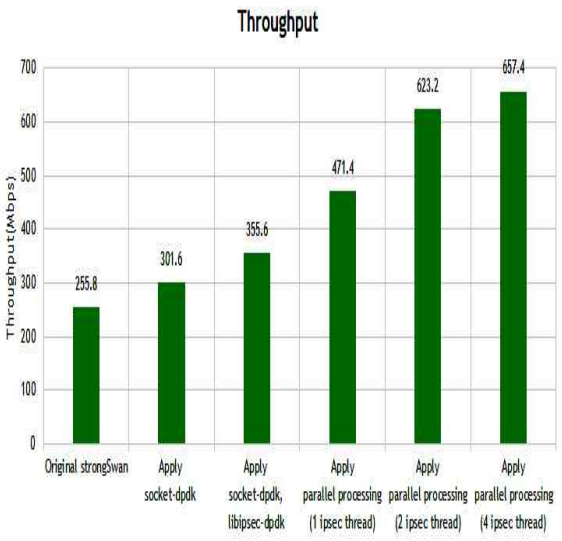


Figure 3: Throughput Graph

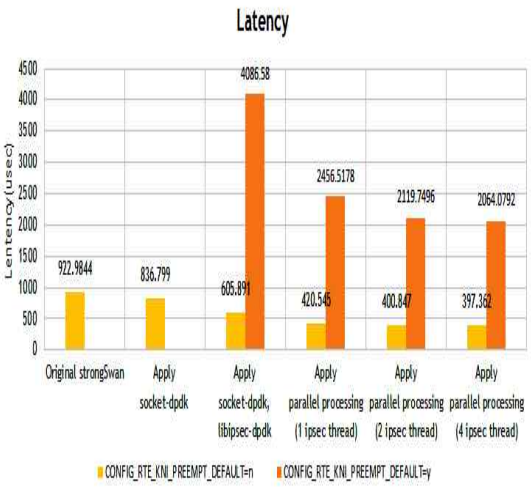


Figure 4: Latency Graph

significant improvement, when we increase the no of core in the encryption/decryption processing part. Using 2 cores, throughput increase is 2.44x of original strongswan' throughput. But throughput improvement is not signigicant between 2 and 4 cores. Using 4 cores, we have achieved upto 2.56x improvement in throughput over origional strongswan.

Figure 4, represents the latency analysis for the test cases. Latency is inversely proportional to through, if throughput is to increase then latency should decrease. it can be seen that latency has reduced upto 1.5x when we used DPDK socket with DPDK KNI. For the multi-core scenario, it further decreases to 2.5x.

From figure 3 and figure 4, it is clear that using zero copy operations in network packet processing operations, we get significant improvement over conventional kernel network packet processing. And multi-core feature, further improves throughput and latency.

VI. Conclusion

From the results, we can conclude that throughput increases with zero-copy operations using DPDK. But there is a high performance boost when adopting multi-core solution in IPsec packet processing. Data plane solution can help up-to a certain point for IPsec processing. As in IPsec the major bottleneck is encryption and decryption. When we changed the model to multi-core, performance improvement is significant relative to non-parallel solution with DPDK. We can achieve up to 2.56x improvement in performance by using parallel processing with DPDK.

References

[1] Steffen, A. "strongSwan-the OpenSource IPsec-based VPN Solution." 2012-03-31]. http://www. strongswan. org (2014).

[2] Scholz, Dominik. "A look at Intel's dataplane development kit." Network 115 (2014).

[3] Steffen, A. "the OpenSource IPsec-based VPN Solution: StrongSwan."

[4] Lacković, Dario, and Mladen Tomić. "Performance analysis of virtualized VPN endpoints." Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on. IEEE, 2017.

비콘 기반 서비스의 접근 취약점에 대한 보안 방안

박준영*

*대륜고등학교 2학년

Security Measures for Access Vulnerability of Beacon-based Services

Junyoung Park*

*Daeryun High School 2nd Grade

요 약

비콘은 Bluetooth 4.0 기반의 저전력 무선 신호 송신 모듈로서, 반경 70m 이내의 모든 스마트폰에 짧은 주기로 신호를 보내 특정 서비스를 요청할 수 있다. 하지만 서비스가 본인인증처럼 개인정보와 연관된 경우, 원격에서의 대리인증 같은 오남용에 대한 구체적인 보안 체계가 요구된다. 따라서 본 논문에서는 비콘이 확실한 위치기반 서비스를 제공하기 위해 접근 취약점에 대응할 수 있는 보안 방안들에 대해 제안한다.

I. 서론

비콘은 Bluetooth 4.0 기반의 스펙으로 저전력 블루투스 기술(BLE, Bluetooth Low Energy)이 핵심이다. 최대거리 70내외로 중장거리 통신이 가능하며, 단순히 특정 신호를 짧은 주기로 계속 브로드캐스팅하는 특징을 지녔다 [1]. 이런 비콘 신호를 스마트폰이 인지하고 어플리케이션을 통해 서비스 서버에 비콘 신호를 전달해주면, 서버는 그에 해당하는 서비스를 제공한다. 그림1은 비콘 서비스의 흐름도이다. 비콘의 신호는 일종의 UUID(Universal Unique Identifier)로, 비콘마다 특정 UUID 값을 가지고 있기 때문에 서비스를 다양하게 제공할 수 있다.

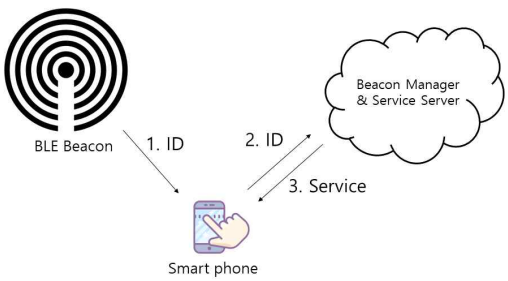


그림 1 . 비콘 서비스의 흐름도

하지만 이런 서비스가 단순한 정보를 제공하는 것이 아닌, 개인정보를 활용하여 제공되는 서비스인 경우에는 보안에 관한 심각한 문제점들이 발생할 수 있다. 가령, 웹 서버를 사용하여 서비스를 제공하는 경우, 서버의 주소와 비콘의 UUID 값을 알 수 있다면, 비콘 반경 내의 스마트폰을 이용하지 않아도 원격으로 웹 서버에 접근이 가능하다. 그렇게 접근한 서버에서 개인정보와 UUID를 이용하여 비콘 반경 내에서 받아야할 서비스를 제공 받을 수 있다.

또 다른 문제점으로, 비콘을 특정 장소에 참여한 사람에게만 서비스를 제공하는 경우를 들 수 있다. 특정 장소에서 이벤트 형식으로 참여하여야 받을 수 있는 서비스가 비콘의 넓은 커버리지로 인하여 장소 외에 있는 사용자에게까지 서비스가 제공될 수 있는 문제가 있다.

따라서 본 논문은 위의 문제점들을 해결할 수 있는 방안으로, 2장에서 비콘의 UUID와 토큰을 이용하는 방식, 3장에서 거리에 따른 비콘의 세기를 측정하여 특정 세기 이상의 신호만 서비스를 제공 받을 수 있는 방식을 제안한다. 4장은 본 논문의 결론이다.

II. UUID 와 토큰의 암/복호화

비콘으로 개인정보를 이용한 위치기반 서비스를 제공 받기 위해서는 먼저 사용자 인증이 필요하다. 서버에서 사용자 인증을 거친 후 비콘의 UUID가 인식되면 스마트폰 어플리케이션이 서버로 UUID 값을 보내 특정 사용자가 서비스를 제공받는 것이 일반적인 흐름이다. 하지만 앞에서 지적했듯 서비스를 제공하는 서버와 어플리케이션에 별 다른 보안 조치가 없다면, 개인정보와 UUID 값만 알고 있어도 비콘 반경 내에 없는 상태에서 서비스를 제공 받을 수 있는 가능성이 존재한다.

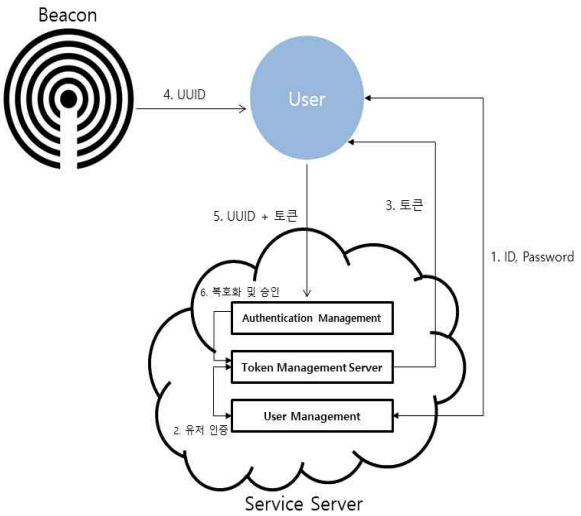


그림 2. UUID 와 토큰을 이용한 서비스 제공

이 문제를 해결하기 위해 서버에서 사용자에게 토큰을 제공하여 UUID와 함께 인증을 거치는 것이다. 그림 2는 사용자 인증으로 토큰이 생성되고, 그 토큰과 UUID이 결합된 값을 이용해 사용자가 비콘 반경 내에 있음을 확인하는 흐름도이다. 먼저, 사용자가 비콘 신호 반경 내에 들어가 있는 상태에서 어플리케이션으로 본인 인증을 거친다. 본인 인증이 확인되면, 서버는 토큰을 생성하여 사용자에게 보낸다. 사용자는 어플리케이션에서 토큰과 비콘에서 받은 UUID를 함께 암호화한다. 암호화된 값을 서버로 전송하면, 서버는 값을 복호화하여 서비스 제공을 위한 마지막 인증 절차를 거치게 된다. 즉, 사용자의 어플리케이션에서 비콘의 UUID와 토큰을 결합하여 암호화가 일어나고, 이 말은 곧 비콘의 반경 안에 없으면 인증을 할 수 없음을 의미한다.

III. 거리에 따른 비콘의 RSSI

비콘은 무선 RF 신호의 일종으로, RSSI(Received signal strength indication)를 신호의 세기로 사용한다. 표 1은 거리에 따른 RSSI 값을 각 거리당 10번씩 측정한 표이다.

Distance(m)	RSSI(dBm)
1m	-64.6
2m	-71
3m	-73.6
4m	-75.6
5m	-76.9
6m	-77
7m	-78
8m	-75.3
9m	-73.6
10m	-82

표 1. 거리에 따른 RSSI 값을 10번 측정한 평균값

표 1과 관련 그림 3은 거리에 따른 평균RSSI 값을 그래프로 나타낸 결과이다. 각 거리당 RSSI를 10 번 측정하여 평균 값을 내었는데 비콘은 RSSI가 안정적이지 않아 평균을 상회하는 값이 있었기 때문에 8m, 9m에서는 예상 값보다 큰 값이 도출되었다.

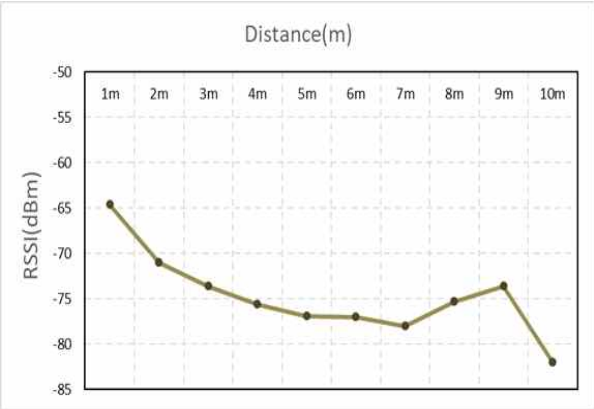


그림 3. 거리에 따른 비콘의 RSSI 그래프

이 그래프의 결과에 따라서, 특정 장소에서 비콘을 설치하고 서비스를 제공할 때, 장소 이외의 이용자가 서비스 제공을 받지 않도록 특정 RSSI 값 이상의 신호만 어플리케이션에서 인식하는 설정이 필요하다. 하지만 멀리 있어도 갑자기 큰 RSSI 값을 받을 때가 있기 때문에, 연속적으로 RSSI를 측정하여 모든 값이 기준 값 이상일 때 서비스를 제공하도록 설정하는 것도 추가적으로 필요하다.

IV. 결론

본 논문에서는 비콘 서비스가 본인인증처럼 개인정보와 연관된 경우, 원격에서의 대리인증 같은 오남용에 대한 구체적인 보안 체계가 요구됨을 파악하고, 사용자가 비콘 서비스에 접근할 때의 취약점을 분석하였다. 이에 접근 취약점에서 대응하는 보안 방안으로 비콘 UUID와 토큰의 암/복호화를 통한 인증방법과 거리에 따른 비콘 RSSI를 이용한 서비스 접근 제어 방법으로 두 가지를 보안 대응방안을 제안한다.

[참고문헌]

[1] 김정환, Design and Implementation of Bluetooth4.0-Based Indoor Location-Based Service System, 석사학위논문, 강원대학교, Feb, 2015.

가상화폐 거래소 탈중앙화 동향에 관한 연구

조형택*, 염흥열**

*순천향대학교 융합서비스보안학과, **순천향대학교 정보보호학과

Research on the trends of decentralized cryptocurrencies exchange

Hyoung-Taek Cho*, Heung Youl Youm**

*Department of Convergence Service Security, Department of Information Security Engineering,, Soonchunhyang University

요 약

최근 많은 사람들이 가상화폐를 온라인으로 거래하기를 원하면서 이를 중개해주는 거래소가 급증 하였다. 이에 따라 거래소를 대상으로 한 공격이 늘어나고 있으며 기존 거래소의 문제점들을 보완할 탈중앙 거래소에 대한 관심이 늘어나고 있다. 따라서 본 논문에서는 기존 중앙화 거래소의 동작방식과 문제점에 대해 알아보고 탈중앙 거래소 모델의 동작방식 및 한계점에 대해 알아본 뒤 향후 발전 방향을 제시 한다.

I. 서론

2008년 비트코인이 창시되고 그 외 다른 가상화폐가 생겨남에 따라 많은 사람들이 가상화폐를 거래하기 시작했다. 많은 사람들이 가상화폐를 온라인으로 거래하기를 원했고 이를 중개해주는 거래소가 점점 증가하고 있다. 기존 거래소는 하나의 서버에 모든 거래내역을 저장하는 중앙화 방식을 채택하고 있었으나 최근 Coincheck, bitstamp 등 여러 해외 거래소 및 빗썸, 유빗 등의 국내 거래소가 해커들에 의해 공격당해 많은 피해를 입음으로서 거래소의 탈중앙화가 주목 받고 있다.[1]

본 논문의 구성은 다음과 같다. 2장에서 기존 중앙화 거래소의 동작방식과 문제점에 대해 알아보고 3장에서는 탈중앙 거래소 모델과 동작방식을 서술한 뒤 탈중앙 거래소의 한계점을 제시한뒤 4장 결론에서 향후 거래소 모델의 보완을 위한 발전 방향을 제시하고 끝을 맺는다.

II. 중앙화 거래소

2.1 중앙화 거래소 개요

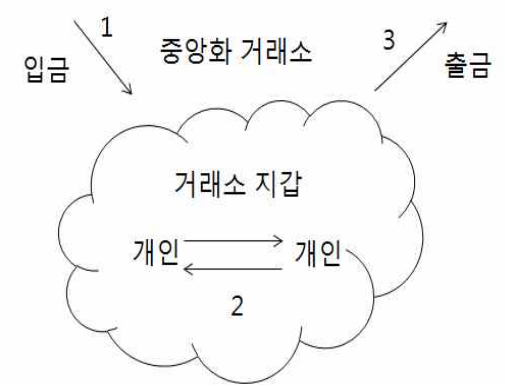
가상화폐 거래소는 거래자가 일반 통화 또는 기타 가상화폐를 통해 가상화폐를 사고, 팔고 교환할 수 있게 해주는 플랫폼 및 응용 프로그램이다. 토큰을 거래할 수 있는 곳이자 한국 원화나 미국 달러와 같은 일반 통화로 가상화폐를 구입할 수 있게 함으로서 가상화폐 생태계에 중요한 역할을 하고 있다.[2]

사용자들에게 익히 알려진 거래소로는 Coinbase, Upbit, Bithumb등이 존재한다.

2.2 중앙화 거래소 동작 방식

기존의 가상화폐거래소는 모든 자산을 거래소가 소유한 지갑에 보관하고 거래소의 사용자들은 실제 자신의 자산이 오가는 것이 아닌 가상화폐를 소유하고 있다는 데이터만 오고가는

시스템이다. [그림 1]은 거래소 사용자가 중앙거래소를 통해 거래과정을 그림으로 나타내고 있다.[3]



[그림 1] 중앙화 거래소 동작방식[3]

1. 거래를 원하는 이용자가 거래소 지갑안의 개인 개좌에 현금 혹은 토큰을 입금한다.[3]

2. 거래소 지갑안의 개인계좌끼리 가상화폐를 거래한 데이터를 송수신 한다.[3]

3. 거래를 마치고 개인 계좌의 가상화폐를 현금으로 환전하여 출금한다.[3]

2.3 중앙화 거래소의 문제점

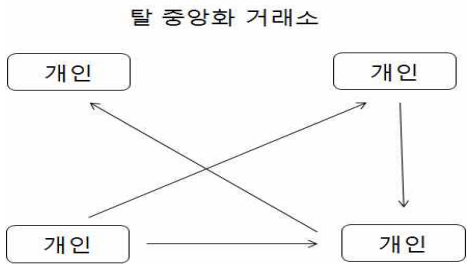
이러한 중앙화 거래소는 모든 데이터가 거래소 서버 한곳에 집중되어 있는 점으로 P2P 네트워크상에서 거래, 운용되어 분산, 공개, 투명이라는 블록체인의 본질적 요소와 전혀 부합되지 않고 있다. 가상화폐 거래 시에 거래 당사자들이 직접 보관·관리해야 할 비밀번호(디지털 서명화된 암호화키), 즉 PK(공개키·개인키)를 이들 중앙화된 거래소가 대리 보관하고 수신·매도·매수를 대리하는 행위는 매우 위험하다. 이러한 중앙화 방식의 거래는 정상적으로 당국에 신고돼 피해 발생 시 충분히 책임질 수 있는 대형 금융 수신 기관만이 운용할 수 있는 것이기 때문이다. 현재 중앙화 방식 거래소의 거래 대리 수신업무 행위는 사실상 유사수신 행위로 볼 수 있다.[3]

III. 탈중앙 거래소

3.1 탈중앙 거래소 개요

탈중앙 거래소는 기존 중앙화 거래소의 취약점을 보완하기 위해 제시된 거래소 모델로 모든 데이터가 거래소 서버 한 곳에 집중되어 있는 중앙화 거래소와 달리 거래 데이터를 거래 대상자들에게 분산하여 저장하는 모델이다. [그림 2]는 탈중앙 거래소를 통한 거래과정을 그림으로 나타내고 있다.[3]

본 논문에서는 대표적인 탈중앙 모델 IDEX, 0x Project, Kyber Network에 대하여 서술한다.



[그림 2] 탈중앙 거래소 동작방식

3.1 IDEX

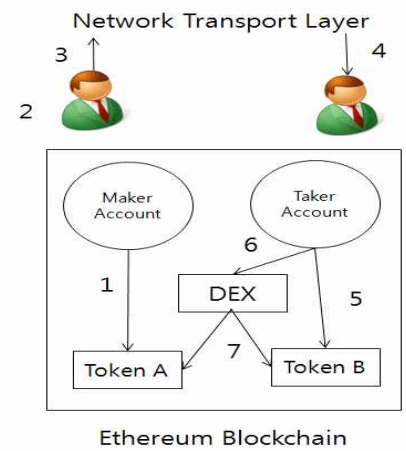
IDEX 모델은 기존의 중앙화 거래소와 비슷한 컨셉인 입금 출금 컨셉을 사용한 거래소 이다. 기존의 중앙화 거래소와의 차이점은 거래소 지갑에 이용자가 입금을 하는 것이 아닌 스마트 컨트랙트 상에 입금 하여 거래를 한다는 점이 큰 차이점이다. 스마트 컨트랙트는 사용자의 투자금을 귀속시키고, 그들의 플랫폼에서 거래를 할 수 있도록 허용 한다. 거래가 종료되면 사용자는 스마트 컨트랙트에서 자신의 투자금을 출금 할 수 있다. 입금 / 출금 모델은 거래 방식이 매우 직관적이며 복잡하지 않으며 스마트 컨트랙트에 투자금이 입금 승인이 될 때까지 대기해야 한다는 단점이 존재한다.[4]

3.2 0x Project

0x Project는 하나의 거래소가 아니며, 여러 거래소들이 사용할 수 있는 공통된 프로토콜을 제공하는 것이 목적이다. 0x 프로토콜은 ERC20

토큰들을 거래하는 능력을 가지고 있으며, 스마트 컨트랙트로 작성되어 이더리움 블록체인 위에서 동작한다. 이는 코스모스나 비트쉐어와 차별되는 큰 장점으로, 사용자들은 중간자를 필요로 하는 별도의 전송 없이 바로 이더리움 토큰을 입금하고 거래할 수 있다.[5]

0x Project 프로토콜의 동작방식은 아래 [그림 3]과 같다.



[그림 3] 0x Project 동작방식

1. 메이커가 이더리움 네트워크에서 DEX(탈중앙 거래소)에 A 토큰을 입금한다.[5]
2. 메이커가 토큰 A와 토큰 B를 거래한다는 메세지를 생성하고, 별도의 중계자에 보낸다.[5]
3. 이더리움이 아닌 이 중계자는 그 메세지를 전파한다.[5]
4. 테이커는 메세지를 받고 해당 주문을 사용하기로 한다.[5]
5. 테이커는 이더리움 네트워크에서 DEX에 B 토큰을 입금한다.[5]
6. 메이커는 DEX에 메이커의 메세지를 보내 주문 처리를 요청한다.[5]
7. DEX는 서명, 만료일 등 여러 조건을 확인하고 두 주체간에 토큰 교환을 실행한다.[5]

3.3 Kyber Network

Kyber Network 모델은 블록체인 상의 탈중앙화된 환전을 제공하며, 실제 호나전 기능을 구축하고 판매자와 사용자를 위한 유용한 결제 API들을 제공하여 토큰 변환을 쉽고 신용 검증이 필요 없게 하는 것이 특징이다. [6]

카이버네트워크에는 5 종류의 역할 구성원들이 있다.

1. 사용자 : 네트워크에서 토큰을 주고 받는다. 개별 사용자와 스마트 컨트랙트 계좌 및 판매자를 포함한다.

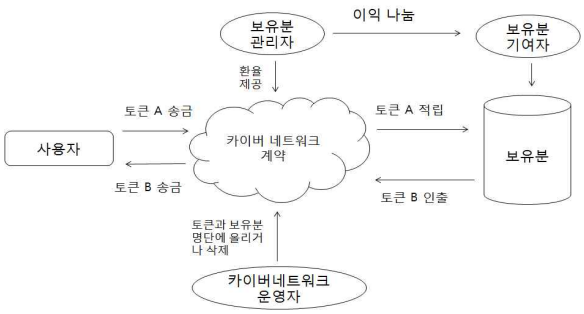
2. 보유분 : 플랫폼에 유동성을 제공한다.

3. 보유분 기여자 : 보유분 개체에 자본을 제공하고 플랫폼 이익을 나눠 갖는다.

4. 보유분 관리자 : 보유분을 유지 관리하고 환율을 결정하여 카이버네트워크에 결정한 환율을 제공한다.

5. 카이버네트워크 운영자 : 네트워크에 보유분 개체들을 추가, 삭제하는 업무를 담당한다

각 구성원들은 스마트 컨트랙트와 독립적인 다른 방식으로 상호 작용하며 사용자는 한 거래에서 보유분이나 카이버네트워크 운영자로부터 어떤 응답도 기다리지 않고 토큰들을 주고받을 수 있다. 보유분 관리자가 환율을 결정하고 계약에 환율을 제공하는 고정된 시간 동안 카이버네트워크 운영자는 보유분을 추가하거나 없애는 책무를 수행한다. 주요 계약은 높은 유동성을 보장하기 위해 보유분 개체에 의존한다. 아래 [그림 4]는 각 구성원들 간의 상호 작용을 나타낸다.[6]



[그림 4] Kyber Network 구성원 간 상호작용

3.4 탈중앙화 거래소 모델별 특징

<표 1>은 본 논문에서 알아본 탈중앙 거래소의 특징을 비교하여 정리 한 것이다.

<표 1> 탈중앙 거래소 비교 분석

거래소	거래 비용	즉각적 거래	블록체인 상	유동성 보장	보안성
Etherdelta[4]	높음	X	O	X	O
0x Project[5]	저	X	혼용	X	불확실
Kyber Network [6]	저	O	O	O	.O

3.5 탈중앙 거래소 한계점

탈중앙 거래소는 기존 중앙화 거래소의 문제점을 해결해 주지만 한계점이 존재한다. 공통적이고 가장 치명적인 단점은 실시간 거래 반영이 늦어진다는 점에 있다. 이는 모든 거래 행위가 블록체인 상에 기록이 되기 때문에 기존 중앙화 거래소 보다 비교적 많은 시간 소요가 나타난다.[1]

Etherdelta 모델은 기존 중앙화 거래소와 비슷한 방식을 채택 하고 있어 스마트 컨트랙트 상에 어떠한 버그가 발생한다면 컨트랙트 상의 자산의 안전을 보장할 수 없다는 단점을 지니고 있으며 0x Project 모델의 경우 거래 속도의 향상을 위해 주문 메시지를 외부 중계자에게 위임하였으나 이 경우 블록체인 상에서 거래 내역이 잠시 벗어나 외부로 전송되기 때문에 상대적으로 보안에 취약할 수 있다는 결점을 가지고 있다. Kyber Network 모델은 타 플랫폼간의 환전이 편리하게끔 구성되었으나 모든 거래가 스마트 컨트랙트 상에서 일어나면서 모든 거래내역, 환전 내역 등의 데이터가 저장되는데 오랜 시간이 걸린다는 단점이 존재한다.[4][5][6]

IV.결론

본 논문에서는 기존 가상화폐 거래소 모델인 중앙화 거래소의 동작방식 및 문제점을 보완하는 탈중앙 거래소 모델의 종류와 동작방식에 대해 알아보고 한계점을 알아보았다.

현재 탈중앙 거래소들은 기존의 중앙화 거래소의 모든 데이터가 한곳에 집중되는 문제점을 해결하고자 하였다. 하지만 "탈중앙 거래소"라는 개념 그 자체로 모순이라는 지적이 있는 것처럼 본 논문에서 제시한 한계점에 대한 연구가 필요할 것이다.

[참고문헌]

[1] "2018 탈중앙화 거래소 생태계 및 현황", 2017

[2] "가상화폐 거래소 구조의 이해와 리스크 관리" coinone, 2017

[3] 홍준영 "문제는 가상화폐가 아니라 '거래소'다", 2017

[4] Aurora Labs,

"IDEX : A Real-Time and High-Throughput

Ethereum Smart Contract Exchange", IDEX, 2017

[5] Will Warren, Amir Bandeali,

"0x : an open protocol for decentralized exchange on the ethereum blockchain, 2017

[6] Loi Luu, Yaron Velner,

"KyberNetwork A trustless decentralized exchange and payment service", KyberNetwork,, 2018

공개툴을 이용한 패턴 기반 보이스피싱 탐지 시스템

탁현주*, 임수민, 김민서, 염흥열*
*순천향대학교 정보보호학과

A Pattern-based Voice Phising detection system using openly available tools

Hyun-ju Tak*, Su-min Lim, Min-seo Kim, Heung Youl Youm*
*Division of information security, Sooncheonhyang University.

요 약

핸드폰의 보급률이 늘어감에 따라 전화를 통한 금융사기 피해가 날로 증가하고 있다. 과거 전화금융사기 범죄수법은 공공기관을 사칭하여 현금인출기로 유도, 지정된 계좌로 돈을 이체하는 수법이 대부분이었으나, 점차 수법이 교묘해지고 지능화 되고 있는 추세이다.
본 논문에서는 늘어가는 전화금융사기 피해를 줄이고자 범죄자의 번호를 차단하는 수동적인 방법이 아닌 전화금융사기에 꼭 필요한 단어를 색출하여 능동적으로 대처하는 시스템을 제안하고자 한다. 이용자는 전화금융사기로 의심되는 전화통화를 녹음하여 논문에서 제공하는 보안전화 시스템에 등록하게 되면 DB에 저장된 피해사례와 비교하여 기존에 존재하는 전화금융사기와의 유사도, 전화금융사기의 유형을 사용자에게 제공한다.

I. 서론

전화금융사기 또는 보이스 피싱(Voice Phishing)은 음성(Voice)과 개인 정보(Private Data), 낚시(Fishing)의 합성어로 음성전화로 범행 대상자에게 전화를 걸어 대상에게 허위사실을 이야기해 피해자의 돈을 가로채거나 특정 개인 정보를 수집하는 사기 수법을 말한다.

표 1. 전화금융사기 피해 현황[1]

구분	발생 건수	검거 건수
2014년	21,634	8,143
2015년	22,205	8,170
2016년	17,040	11,386
2017년 上	**10,626**	**11,005**
2018년 上	**16,338**	**15,135**
증감률	53.8%↑	37.5%↑

경찰청이 발표한 2018년도 상반기 전화금융사기 피해 규모는 2017년도 상반기와 대비하여 53.8.4%, 피해액은 68.3% 증가된 것으로 조사됐다.[1]

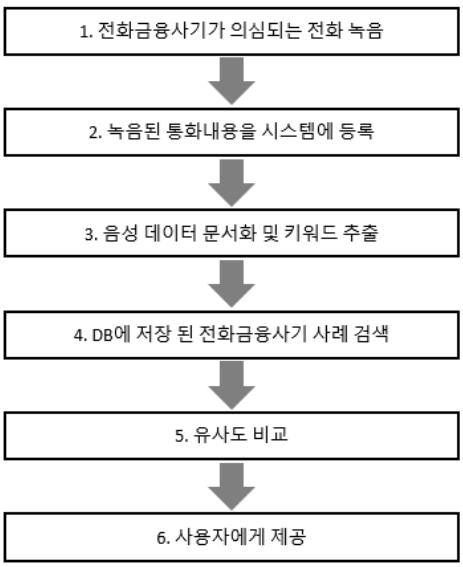
기존 전화금융사기에 대한 대표적인 예방 방안으로는 전화금융사기에 사용된 전화번호를 수집하고 데이터베이스를 구축하여 범죄를 사전에 차단하는 방법이 사용되었으나 시간이 지남에 따라 범죄조직이 발신번호를 변경하여 범죄에 이용하면 효과적으로 차단할 수가 없는 수동적 기능을 제공해 오고 있다.[2]

본 논문에서는 2장에서는 전화 금융 사기에서 자주 언급되는 단어들을 색출하여 범죄를 가능성을 기존에 발생한 전화금융사기를 기준으로 색출된 단어를 통해 유사도 분석하고 이를 통해 범죄 가능성을 판단하는 시스템에 관해 서술한다. 판단된 결과는 전화금융사기 피해를 사전에 방비할 수 있도록 하는 능동적인 시스템을 제공하여 추후 여러 방면에서 활용할 수 있도록 하고자 한다.

II. 본론

2.1 시스템 흐름도

사용자는 전화금융사기가 의심되는 전화를 녹음하고, 녹음된 음성 파일을 보이스피싱 탐지 시스템이 탑재된 사이트에 등록하게 된다. 등록된 음성 파일은 사이트 내부의 Google Cloud Speech-to-Text API를 이용해 문서화 되어 저장되고, 저장된 문서는 사이트 내에서 TF-IDF 알고리즘을 이용해 유사도를 분석하게 된다. 유사도는 사용자가 등록한 새로운 사기사례 파일과 DB에 미리 저장된 사기사례 문서 군을 서로 비교하고 그에 따라 전화금융사기 유형과 유사도 퍼센트를 사용자에게 제공한다.



[그림 1] 시스템 실행 순서

2.2 구글 웹저장소 음성 지원 프로그램 적용기(Google Cloud Speech API)

녹음된 통화 내용을 텍스트로 저장하는 데 활용하는 공개 도구로 짧거나 긴 오디오를 사용할 수 있는 머신러닝 기반 음성 텍스트 변환 기술이다. 사용하기 간편한 API로 신경망 모델을 적용한 Google Cloud Speech-to-Text를 사용하면 개발자가 오디오를 텍스트로 변환할 수 있다. 음성 명령 및 제어기능을 구현하고 전화 상담실의 오디오를 텍스트로 변환하거나 실시간 스트리밍 또는 사전 녹음 오디오를 처리할 수 있다. 총 4개의 모델을 제공하고 있으며 본 논문에서는 전화통화 오디오 모델을 사용한다.

시스템의 3번째 순서에 활용되는 Google Cloud Speech-to-Text API의 전체적인 흐름은 다음과 같다. 먼저 Google Cloud의 platform에 연결한 후, 해당 Google Cloud 내의 Storage에 저장된 파일을 읽어와 각 음성파일들을 API를 호출하여 JSON형식으로 나타내고 이를 .txt 형태로 문서화 한다.

```
Input your project id: polished-engine-220913
Input yout bucket: yumstone

===== FILE LIST =====
[0] filename: 1.경찰 사칭.flac
[1] filename: 2.은행 사기.flac
[2] filename: 3.경찰 사칭.flac
=====================
```

[그림 2] 프로그램 초기 실행 화면

[그림 2]와 같이 사용자의 프로젝트 ID와 버킷 이름을 입력받아 해당 Google Cloud Platform에 접근한다. 사전에 발급받은 JSON 형식의 사용자 인증키를 통해 사용하고자 하는 API를 호출한다.

Google Cloud Platform에 연결되어 Storage의 버킷 안의 음성 파일들을 읽어온 후에 호출되는 Speech-to-Text API는 API 자체적으로 음성을 텍스트화하여 JSON 형식으로 나타내고 이를 .txt형식으로 저장한다. 이때 모든 음성 파일들을 .flac 확장자를 사용한다.

```
// flac 파일
this.recognitionConfig = new RecognitionConfig()
{
    Encoding = RecognitionConfig.Types.AudioEncoding.Flac,
    LanguageCode = "ko-KR",
};
```

[그림 3] FLAC 확장자 사용

FLAC의 경우 MP3, AAC와는 다르게 무손실 오디오 확장자이며 한정적인 용량에서도 비교적 활용하기 쉽기 때문에 휴대전화기에 통화내용을 녹음하는 본 논문의 시스템에 적합하므로 FLAC의 음성 파일 확장자를 사용한다.

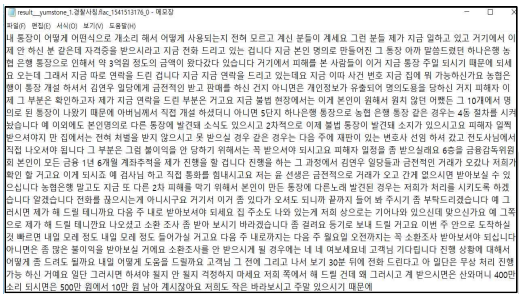
```
foreach (var result in responseResult.Result.Results)
{
    var timestamp = SpeechConnector.UnixTimeNow();
    using (StreamWriter rdr = new StreamWriter($"result_{soundFilename}_{timestamp}_{i}.txt"))
    {
        foreach (var alternative in result.Alternatives)
            rdr.WriteLine(alternative.Transcript);
        rdr.Close();
        Console.WriteLine($"The result saved the file -> result_{soundFilename}_{timestamp}_{i}_.txt");
    }
    i++;
```

[그림 4] .txt 형식의 저장방법 정의

.txt식으로 저장된 파일들은 형태소분석 및

유사도 분석을 위해 활용된다.

결과는 아래 [그림 5]와 같이 프로그램이 존재하는 폴더 내에 분기별로 나뉘어 저장되며 아래 그림은 분기별로 나뉜 문서를 병합한 내용이다.



[그림 5] API 사용 결과 문서

2.3 문서 유사도 분석

2.3.1 한국어 정보처리 모음(KoNLPy)

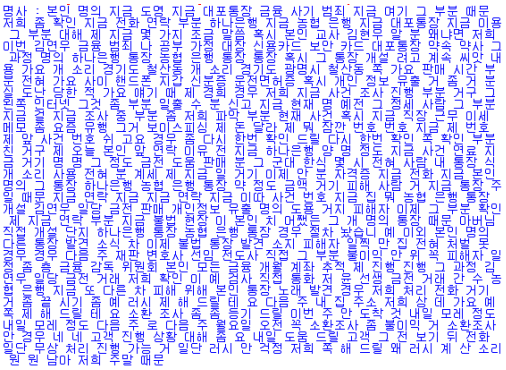
KoNLPy는 파이썬으로 한국어 정보처리를 할 수 있게 하는 패키지이다. KoNLPy를 활용하여 추출해온 텍스트 파일들의 명사만 분류한 후 TF-IDF 알고리즘을 통해 여러 텍스트 파일들의 유사도를 측정할 수 있다.

먼저 from konlpy.tag import Twitter, from konlpy.utils import pprint 명령어를 사용하여 KoNLPy 모듈을 불러오고 필요한 Tagging Class인 Twitter를 불러온다. 필요한 텍스트 파일을 불러온 후 ' '.join(twitter.nouns()) 명령어를 통해 명사만 분류한다. print 명령어를 이용하여 분류된 명사들을 볼 수 있다.

```
from konlpy.tag import Twitter
from konlpy.utils import pprint

twitter = Twitter()
f = open('result___yumstone_1.경찰사칭.flac_1541513176_0.txt', 'r', encoding = 'utf-8')
s = f.read()

nouns = ' '.join(twitter.nouns(s))

print('명사 : ' + nouns)
```

[그림 6] 명사 추출 소스코드

아래는 KoNLPy로 명사 추출을 적용한 후의 그림이다. 문장이 아닌 단어만 추출된 것을 볼 수 있다.



[그림 7] KoNLPy 적용 후

2.3.2 단어빈도 - 연문서 빈도

표 1. TF-IDF 정의[2]

수식	정의
TF	$tf(t,d) = \frac{f(t,d)}{\sum f(w,d)}$ $f(t,d)$: 단어 t가 문서 d에 출현한 횟수 $\sum f(w,d)$: 문서 d의 길이
IDF	$idf(t,D) = \log \frac{\lvert D \rvert}{\lvert d \in D : t \in d \rvert}$ $\lvert D \rvert$: 문서 집합 D에 포함된 문서의 개수 $\lvert d \in D : t \in d \rvert$: 문서집합D에 단어 t가 출현한 문서의 개수
TF-IDF	$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$

정보검색과 텍스트 마이닝 분야에서 가중치를 나타내는 것에 활용되고 있으며 여러 개의 문서로 이루어진 문서 집합이 있을 때 등장하는 특정 단어가 그 문서에서 얼마나 중요한 단어인지를 나타내는 통계기반 기술이다.

단어빈도(TF)는 특정 단어가 문서 내에 얼마나 자주 등장하는지를 표현하는 값으로 문서에서 자주 등장하는 단어라면 단어의 중요도를 높게 측정한다.

역문서 빈도(IDF)는 전체문서의 수와 특정 단어가 포함된 문서의 수를 서로 나누어 로그

를 취한 값으로 특정 단어가 모든 문서에서 등장하는 흔한 단어라면 TF-IDF의 가중치를 낮춰 주려는 용도로 사용된다.

위의 자료를 토대로 .txt로 저장된 파일들의 유사도를 비교하여 유형을 나타내 준다. 먼저 실제 사례들의 명사를 분리하여 유사도 비교를 위해 TF-IDF 방식으로 단어 가중치를 조정한 벡터를 생성한다. 코드는 아래와 같다.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from konlpy.tag import Twitter
from konlpy.utils import pprint

twitter = Twitter()

mydoclist = ['result___yumstone_1.경찰사칭.flac_1541513176_0.txt',
             'result___yumstone_3.경찰사칭.flac_1541513418_0.txt',
             'result___yumstone_2.은행사기.flac_1541513320_0.txt',
            ]

doc_nouns_list_t = []

for doc in mydoclist:
    f = open(doc, 'r', encoding = 'utf-8')
    s = f.read()
    nouns = ' '.join(twitter.nouns(s))

    doc_nouns_list_t.append(nouns)
```

[그림 8] 명사 추출 소스코드

from sklearn.feture_extraction.text import TfidfVectorizer, 명령어를 사용하여 sklearn.feture_extraction.text 모듈을 불러온다. 비교할 텍스트 파일들을 mydoclist 리스트로 선언해주고 추출한 명사들을 저장할 리스트 doc_nouns_list_t를 선언한다. 반복문을 이용하여 mydoclist에 있는 텍스트 파일들을 읽어오고 명사를 추출한다. 추출한 명사들은 append 명령어를 이용해 doc_nouns_ list_t에 저장한다.

각 문서를 통해 추출된 명사들은 TF-IDF 방식으로 유사도를 비교하고 결괏값을 사용자에게 제공한다.

```
tfidf_vectorizer = TfidfVectorizer(min_df=1)
tfidf_matrix_t = tfidf_vectorizer.fit_transform(doc_nouns_list_t)

document_distances_t = (tfidf_matrix_t * tfidf_matrix_t.T)

print(document_distances_t.toarray())
```

[그림 9] 유사도 비교 코드

TfidfVectorizer를 이용하여 TF-IDF 방식으로 단어의 가중치를 조정한 BOW 벡터를 생성한다. 아래는 유사도를 비교한 결괏값이다.

```
[[1.         0.59938205 0.19539524]
 [0.59938205 1.         0.10010524]
 [0.19539524 0.10010524 1.        ]]
```

[그림 10] 결과 값

각각 자기 자신과의 유사도 값은 1로 100% 동일한 문서라는 결과가 나왔고 범죄자가 동일하게 경찰을 사칭한 내용의 두 파일의 유사도는 약 0.6, 60%유사도를 보여주고 있다. 경찰을 사칭한 내용의 파일과 은행을 사칭한 내용의 파일은 평균 0.15, 15%가 나온 걸 통해 본 논문에서 제공하는 시스템을 이용한다면 전화금융사기 범죄의 유형과 유사도를 판단하여 사용자에게 제공할 수 있다.

III. 결론

정부에서는 범죄자의 목소리를 공개하여 국민에게 전화금융사기 피해예방요령 교육과 함께 이용하는 등 홍보를 강화하고 있지만, 금융사기 범죄가 갈수록 지능화되고 있어 전화금융사기 범죄를 제거하는 것은 현실적으로 불가능한 것으로 보인다.

현재 구글 플레이스토어에 등록된 피싱 방지 애플리케이션들은 대다수가 전화번호 필터링과 문자메시지 필터링 기능만을 제공한다. 전화금융사기에 이용된 전화번호는 금융감독원이나 통신사 전화번호 데이터베이스에 등록하여 사전에 피싱 범죄를 차단하는 기능을 제공하지만, 전화금융사기에 이용되는 전화 대부분은 인터넷 전화로 발신번호 조작이 가능하므로 문제해결에 한계가 있다고 생각된다.

이에 따라 피해 당사자가 전화 등 정보 통신망을 이용한 금융사기에 당하지 않도록 전화금융사기 범죄에 필요한 특정 단어를 색출하여 기존의 사례들과 비교한 뒤, 이용자들에게 범죄 가능성에 대해 알리고 경고함으로써 전화금융사기 범죄가 현저히 줄어들 것으로 예상한다.

[참고문헌]

[1] 구자윤, "연도별 보이스피싱 피해현황", http://www.fnnews.com/news/201808062154364349

[2] Manning, C. D, Raghavan, P, Schutze, "Introduction to Information Retrieval", Cambridge University Press, pp. 100-123

[3] 신동헌, "TF-IDF 기반 이직 대상 기업 추천 시스템 설계 및 구현", 숭실대학교 소프트웨어 특성화대학원 석사학위 논문, 2017

카드해킹의 문제점과 대비방안

이진우, 이정훈, 이창엽,
노무승, 오현수
울산대송고등학교 3학년, 울산범서고등학교 3학년, 창원경일고등학교 2학년,
창원중앙고등학교 1학년, 부산진고등학교 1학년

Problems and countermeasures against card hacking

Lee Jinu, Lee Jeonghun, Lee Changyeop,
Rho Moo-Seung, Oh Hyensu
Ulsan Daesong High School third grader,
Ulsan Beomseo High School third grader,
Changwon Gyeongil High School second grader,
Changwon Jungang High School first grader,
BUSAN Pusanjin High School first grader

요 약

현재 사용하고 있는 여러 가지 카드들은 자신의 신분을 증명하거나, 혹은 결제, 어떠한 건물에 들어갈 수 있는 권한을 가지고 있는 등, 다양한 용도로 사용 되고 있다. 이러한 카드들이, 다양한 방법으로 복제될 수 있는 문제점이 있어, 이에 저희들은 카드해킹과 관련된 해킹 기법에 대해 조사해보고, 국내에서 어떻게 대응할지, 현 대응방안이 정말로 안전한지, 다시 점검해보고자 한다.

I. 서론

[1]MS카드는 카드 뒷면의 검정 띠 부분이 자기적 성질을 가지거나, 자기 스트라이프라 불리는 자성체를 넣어둔 카드이다. [2]IC 카드와 비교해서, 스키밍 공격에 취약하며, 정보 노출이 쉽고, 외부 자기장의 영향으로 인해, 데이터가 손상되곤 한다. MS카드의 경우 [3]ISO/IEC 7813규격을 준수하며 Track1, Track2, Track3로 정보를 저장할 수 있는 공간이 나뉘는데, Track1 공간에는 영숫자 79자리까지 저장 가능하며, 주로 부가적인 정보를 남길 때 이 트랙을 사용한다. Track2 공간에는 40자리의 숫자를 저장 할 수 있으며, 주로 카드 번호, 결제 금액 등을 저장하는데 사용한다. Track3 공간에는 107자리의 숫자를 저장 할 수 있으며, 주로 긴 숫자 데이터를 처리할 때 사용한다.

[4]RFID란, 사물에 태그를 장비하여, 근거리, 장거리(최대 100m)에서도 사물의 정보를 수집 혹은 저장, 처리함으로써, 원격 카드처리 혹은 관리, 정보의 교환 등 다양한 산업 현장 혹은 일상에서 쓰이고 있다. RFID카드의 종류에는 MIFARE CLASSIC, MIFARE Ultralight, HID, HID iClass, ISO14443a, ISO14443b, ISO1569, SRI512, SRIX4K, Legic, epa,

1) 마그네틱 | Magnetic Stripe
2) Intergrated Circuit
3) 국제표준화기구 및 국제전기기술위원회가 부호화한 국제 표준으로, ATM 또는 신용 카드와 같은 금융 거래 카드의 속성을 규정한다.
4)RFID(Radio-Frequency Identification)는 주파수를 이용해 ID를 식별하는 SYSTEM 으로 일명 전자태그로 불린다.

em410Xm Em4x50, Ti, Hitag/Hitag2, indala, T55xx, FlexPass, VeriChip, PCF7931, Kantech ioProx 등이 있다.

IC카드는 반도체 기반으로 제작된 직접회로(IC)칩이 삽입되어 있는 카드를 말하며, MS카드와 달리, 외부 자기장에 의한 데이터 손상이 없을 뿐만 아니라, 개인정보 저장 및 처리가 가능한 양방향 확인이 가능함으로써, MS카드보다 더 나은 보안성을 가지고 있다. 하지만 보안성이 뛰어난 만큼 RFID, MS카드 보다 단가가 비싸 다른 카드에 비해 잘 사용하지 않는 점이 있다.

위와 같이 다양한 산업 군, 그리고 우리의 일상 속에 깊이 관여되어 있는 카드 기술들은 우리의 삶을 더욱 편리하게 해주었지만, 이를 이용한 크래커들의 공격으로 인해, 편리 하였던 기술들이 오히려 독이 되어, 문제가 발생하고 있어, 카드 해킹 기법에 대하여 연구를 시작하게 되었다.

II. 본론

2.1 카드 해킹 기법의 이해

그림 1 | MS카드 스키밍 장치 'MSR605X'

MS카드의 경우, 피해자가 MS카드의 자기 테이프를 카드 리더기에 긁게 되면, 자기 테이프에 저장한 값이 나오게 되며, 해당 값을 통해, 사용자를 식별하거나, 결제를 진행하게 된다. 이를 이용해 크래커는, 리더기에 카드 정보를 빼돌릴 목적으로 흔히 스키밍 장치라 불리는 또 다른 리더기를 덮어씌우는 형태로 설치하여, 피해자가 MS카드를 긁게 되면, 해당 MS카드의 값이 장치에 들어와 저장하게 되고, 추후 해당 값을 다른 MS카드에 저장시켜 카드 복제를 진행하게 되며, 해당 기법을 '오버레이 스키밍'이라 부른다.

그림 2 | RFID카드 복제 장치 'Proxmark3'

RFID카드와 같이, 비접촉식 카드의 경우, RFID의 동작원리를 먼저 파악하면 어떻게 복제가 가능한지 파악할 수 있다. RFID의 경우, 태그에 목적에 맞는 정보를 입력한 후, 부착하게 되는데, 그 후 리더의 안테나를 통해 발사된 무선 주파수가 태그에 접촉하게 되면, 해당 태그가 주파수에 반응하여 입력된 데이터를 안테나로 전송하게 되며, 안테나는 전송 받은 데이터를 디지털 신호로 변조하여 리더에 전달하게 되고, 리더는 데이터를 해독하여 컴퓨터 혹은 다른 장치로 전달하는 방식으로 진행된다. 해당 과정에서 크래커는 리더를 준비하여, 피해자에게 접근한 뒤, 리더기를 통해 태그의 값을 알아내며, 해당 태그 값을 다른 RFID카드에 저장시켜, 카드 복제를 진행하게 된다. 이러한 기법을 'RFID 스키밍'이라고 부른다.

별도로 위와 같은 기법으로 카드 복제 이후, 카드 프린터기를 이용하여 카드의 표면까지 정상 카드로 위조하는 이른바 사회공학기법을 사용해 타인에게 눈속임을 하여 정상적인 카드인 것처럼 보이게 만들 수 있으며, 이는 신분증 위조, 사내 출입증 복제 등 각종 공, 사문서 위조에 사용될 수 있다.

그림 3 | 카드 프린터기 'TP-9100'

2.2 카드 해킹 기법을 이용한 범죄 사례

MS카드는 5)ATM 카드 출입구와 비슷한 모양의 MS카드 스키머를 정상적인 카드 출입구에 덧붙여 신용카드 정보를 빼가는 방법이 가장 대표적인 사례로, 최근 2018년 11월 4일에 검거된 루마니아인 카드복제 범죄조직원들의 경우 MS카드로도 돈을 인출할 수 있는 Fallback 기능이 있는 신용카드가 주로 해외 신용카드인 점을 악용하여 국내 ATM에서 복제된 해외 신용카드로 현금 인출에 성공하였으며, 인터폴과 한국 국제범죄수사대의 협력으로 체포되었다. 이외로 주유소나 편의점에 위장 취업하여 소비자들에게서 카드를 건네어 받을 때 소형 MS 카드 스키머를 이용하여 카드정보를 저장하여 복제하는 사례나, 업소에서 복제된 카드를 사용하고 사용한 금액의 일부를 수수료로 업소에 지급하고 나머지를 현금으로 돌려받는 일명 '카드깡'에 이용된 사례가 있다. 심지어 2014년 6월 13일, 선불식 충전카드인 '마이비' 교통카드의 RFID 취약성을 악용해 금액을 변조하여 약 1억 8천만 원 상당의 부당이득을 챙긴 조직이 검거되기까지 하였으며, 2015년 1월 30일 KBS뉴스 현장추적에서 복제한 RFID카드를 이용하여 호텔 도어락을 열거나, 정부청사 출입까지 가능한 것에 대하여 보도한 사례가 있다. 전문적인 카드복제 범죄자들의 경우 사회공학기법을 추가적으로 적용하여 카드 프린트기를 사용해 카드에 이미지와 색을 입히고, 카드 양각기로 카드에 볼록한 번호를 새기며, 티핑기를 이용해 카드 번호에 은박을 입히는 등 진짜 신용카드로 보이게끔 다양한 카드복제 장비를 사용하고 있다.

5)현금 자동 입출금기 (Automated Teller Machine)

2.3 IC 카드, 정말 안전한가?

IC카드는 현재, MS, RFID카드의 보안 한계점을 해결하고, 더 많은 데이터를 저장할 수 있는 장점을 통해, 타 방식의 취약한 카드들의 대책방안 중 하나로 나오고 있다. 하지만, 이 IC카드가 정말로 안전하다고 볼 수 있는지 의심이 갈 수 있다고 생각한다. 이 점을 생각하여 관련 사례를 조사해본 결과, IC 카드의 6) Chip & PIN이 한 해킹 그룹에 의해 보안이 깨진 사례가 있었다. 해당 사건의 원리는 IC 칩이 개조되고, IC칩에 FUN이라고 각인한 다른 칩을 덮어 납땜한 상태였으며, 일반 IC 칩보다 0.4~0.7mm 두꺼웠다고 한다. 해당 카드를 엑스선을 통해 분석한 결과, 개조 칩에는 배선이 추가되어 회로가 변경되었으며, 이를 이용하여 7)POS시스템에서 모든 8)PIN코드를 허용 상태로 바꿔버리는 점을 이용한 것이었다. 하지만, 해당 취약점의 경우, 그 당시, PIN인증이 유럽에서 EMV카드의 거래 확인과 분리되어 있다는 사실을 이용하여, 해킹을 진행하였으며, 현재 이 부분은 해결된 상태이다. 이와 별개로 IC카드와 연계된 실제 피해 사례는 없는 것으로 확인 되었기에, 현존하는 카드 보안 체계 중 가장 안전하다고 볼 수 있다.

2.4 카드해킹의 실질적인 대책 방안

[2.4.1] MS 훼손

MS 카드의 경우, 앞에서 살펴보았듯이, 심각한 보안 문제가 존재하며, 스키머에 상당히 취약한 편이다. 이러한 상황에서 업주들은 가성

6)(chip & pin)은 직불카드 또는 신용카드로 물건 값을 계산하는 걸 뜻한다.

7)point of sales system. 판매시점정보관리 시스템

8)PIN(Personal Information Number)본인 확인 수단으로 사용되며 타인의 데이터 접근을 막는다.

비 때문에 마그네틱 카드를 채용하고 있지만, 보안상의 이유로 가능하다면, 마그네틱 카드 사용을 중단하고, IC 카드로 새롭게 카드를 발급하거나, 금융권에서는 마그네틱 + IC 카드를 IC 카드로 바꾸는 등의 조치를 취해야할 것이다. 지금으로써는 마그네틱 + IC 카드를 사용하게 되는 경우, 예를 들어, 편의점 혹은 식당에서 해당 카드를 사용하게 될 경우, 종업원이 이 과정에서 따로 준비한 MS 카드 리더기에 읽게 하거나, POS 기기의 보안적 문제, 네트워크의 문제 등의 사유로 카드 보안의 위협이 되기도 하니, 마그네틱 부분에 네오디뮴 자석을 이용하여, MS 부분을 사용할 수 없게 만들어, IC 카드만 사용하는 것을 추천한다.

[2.4.2] POS기의 정기적인 업데이트 및 관리

실제로, POS 기기로 인해, 카드와 관련된 정보가 유출되기도 하였으며, 보안 패치가 지원되지 않는 구 버전의 [9]OS를 사용하는 등의 관리 문제로 인하여, 다량의 카드 정보가 유출되기도 했다. 이 점을 생각한다면, POS 기기를 사용하는 업체에서는 POS 기기의 보안 패치를 정기적으로 적용하고, 지원이 종료된 OS의 경우, 지원이 진행되는 OS로 변경을 하여, 예방 조치를 취해야할 것이다.

[2.4.3] RFID 차폐 필름

RFID 스키밍으로 인해, 외국에서는 많은 카드 복제가 발생하기도 했고, 실제로 사회공학에 활용되어, 보안 사고로 이어지기도 한다. 이를 사전에 방지하기 위해서는, RFID 공격기법의 특징상, 전파만 막아도 이를 예방할 수 있기에, 저렴한 방법으로는 쿠킹호일을 보호하고자 하는 카드에 씌워 전파의 영향을 받지 않도록 하는 방법이 있으며, 해당 원리를 이용하는 차폐 필름 혹은 관련 제품을 사용하는 방법이 있다. [EX] 지갑 / 여권 케이스 / ETC]

III. 결론

본 논문에서는 결론적으로 MS와 RFID가 들어간 카드를 채택하고 있는 카드회사 및 기업은 비용이 조금 더 들더라도 IC카드로 전환해야 될 필요가 있다. 또한 사람들에게 해당 카드가 취약하다는 사실을 알릴 필요가 있으며, 이미 MS, RFID가 들어간 카드를 사용 중인 경우 최대한 카드 복제 예방방법을 숙지하여, 사용에 주의를 해야 된다. 경비원, 주점 종업원과 같이 카드를 검증하는 사람들은 사회공학에 대비하여 카드를 주의 깊게 봐야 되며, 복제된 카드를 사용하여 피해를 보는 쪽은 대부분 업주 측이므로 카드가 조금이라도 일반적인 카드와 다르다면 바로 의심을 하여 조치를 취해야 한다.

9) **운영 체제**(運營 體制, Operating System)는 사용자가 컴퓨터를 쉽게 다루게 해주는 인터페이스다.

[참고문헌]

[1] '멤버십 카드 위조해 현금 빼낸 루마니아인 구속',한겨레신문,2018.11.04.,

http://www.hani.co.kr/arti/society/society_general/868697.html?_fr=mt2

[2] "ATM기 '고품질 스키머'에 고객들은 속수무책", RadioKorea뉴스, 2016.01.09.,

https://www.radiokorea.com/news/article.php?uid=235386

[3] '신용카드 IC칩 보안, 어떻게 뚫었을까'

, TechHolic뉴스, 2015.11.02.,

http://www.techholic.co.kr/news/articleView.html?idxno=42664

[4] "[현장추적] '카드 출입증' 1분이면 복제 정부청사도 뚫려", KBS뉴스, 2015.01.30.,

https://www.youtube.com/watch?v=X7Q3pRRQf7o

[5] "마이비카드 해킹해 잔액 조작 일당, 1억 8천여만원 챙겨", 보안뉴스, 2014.06.13.,

https://www.boannews.com/media/view.asp?idx=41469&kind=1

[6] information technology Identification cardsFinancial transaction cards, ISO/IEC 7813,2006
https://www.iso.org/standard/43317.html

스마트 헬스케어 서비스를 위한 IoT 센서기반 플랫폼 서비스

이원영, 김지영, 강희진, 한수민
서울여자대학교 정보보호학과

IoT Senser Based Platform services for Smart Health Care Services

Gil Wonyoung Lee, Jiyoung Kim, Heejin Kang, Sumin Han
*Division of information security, Seoul Woman University.

요 약

스마트 헬스케어는 건강관련서비스와 의료IT가 융합된 종합의료서비스이다. 스마트 헬스케어는 의료비 절감과 사회경제적 비용 감소라는 경제산업적 파급효과가 있지만, 이를 위한 투자 비용 및 서비스에 대한 접근성의 한계라는 문제가 뒤따른다. 본 연구에서는 현 시점의 애플리케이션의 단점을 보완하여, 다양한 의료용 센서를 수용할 수 있는 통합 애플리케이션을 개발하는 것을 연구의 목표로 한다. 본 연구의 세부 기능은 다음과 같다. 사용자의 편리성을 위하여 물리적인 연결 대신 블루투스 통신을 통하여 건강 정보를 앱 내부에 저장한다. 또한, 사용자가 자신의 건강정보를 쉽게 파악할 수 있도록 시각적인 인터페이스를 제공한다. 데이터를 효율적으로 관리하기 위해 앱 내의 데이터베이스를 구축하고 이의 보안성을 강화한다. IoT 센서기반 플랫폼 서비스를 개발함으로써, 손쉬운 센서 전환, 데이터 보안 강화, 다양한 데이터의 사용, 위급 시 SOS 전송 및 알람 기능과 관련된 여러 센서를 다룰 수 있다. 본 연구의 기여점은 편리성, 투자 비용 및 사용자의 비용적 측면의 한계의 극복에 있다.

핵심어: 스마트 헬스케어, 아두이노, 플랫폼 서비스, 센서기반, 편리성

I. 서론

의료서비스의 패러다임이 치료, 병원에서 소비자 중심으로 변화하면서 스마트 헬스케어 시장이 급격하게 성장하고 있다. 이러한 급격한 변화를 겪고 있는 헬스케어 내에서 생태계 활성화를 위해서는 비용적인 문제라는 한계를 극복해야 한다. 미국 정부에서는 비용 효율성이 높은 의료를 통해 절감된 의료재정 혹은 경제적 효과를 다양한 지불 방식으로 배분함으로써 생태계의 순환을 목표로 하고 있다. 스마트 헬스케어 생태계 활성화를 위해 비용적인 문제라는 한계를 극복해야한다.

본 연구에서는 첫 번째 단계로 근본적인 의료기기의 비용 절감과 투자비용의 감소를 보완하기 위하여 아두이노를 이용하여 다양한 외부의료센서들을 하나의 애플리케이션으로 통합하여 사용할 수 있는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 헬스케어 서비스 시장 동향을 살펴본 후 3장에서는 시스템 구성 및 설계를 설명하고 4장에서는 구현결과를 제시한 후 결론을 내고자 한다.

II. 헬스케어 서비스 시장동향

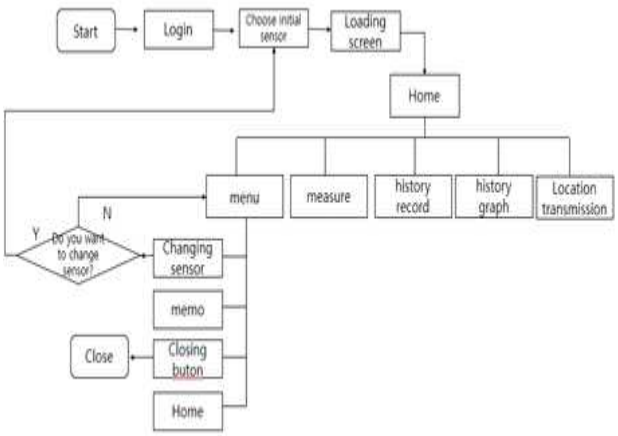
세계 각국에서 정부 차원에서 스마트 헬스케어 산업 육성책을 추진 하고 있다. 기존 병원이나 제약사 등 의료 산업에서도 ICT기업과 협업하

여 신규 사업에 진출하는 모습을 보이고 있다. [1] 이에 글로벌 스마트 헬스케어 시장규모는 지속적인 성장을 보일 전망이다. 한국보건산업진흥원에 따르면 2014년 기준 210억 달러에 머물렀던 글로벌 스마트 헬스케어 시장규모가 2020년에는 1015억 달러 규모가 되면서 약 4.8배의 성장을 보일 것으로 전망했다. [2] 2017년 12월 18일 산업통상자원부가 발표한 '새 정부의 산업정책 방향'의 내용에 따르면 '5대 신산업 선도 프로젝트에 바이오,헬스 분야를 포함시키고 예산을 많이 증가시켰다. [3]

그러나 헬스케어 어플리케이션 상황을 보면, 의학 분야보다는 피트니스와 건강분야에 집중되어 있다.

III. 시스템 구성 및 설계

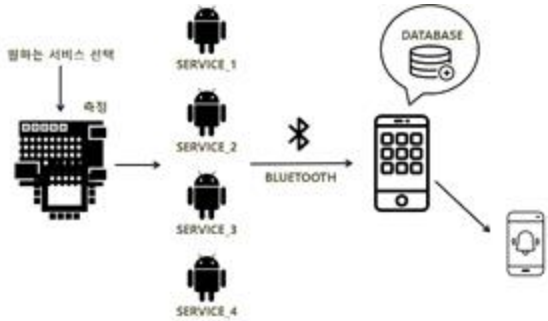
3.1 애플리케이션 진행 순서



[그림 1 Flowchart]

해당 애플리케이션을 처음 들어갈 경우, 가장 첫 번째로 나오는 것은 로그인 화면이다. 만일 아이디가 없을 시 회원가입 란을 통하여 회원가입 할 수 있다. 로그인 후, 사용할 센서를 선택하라는 화면이 뜬다. 화면에서 센서 선택 후 선택완료 버튼을 누르면 홈 화면으로 이동 후 각각의 기능을 수행한다.

3.2 데이터 흐름도



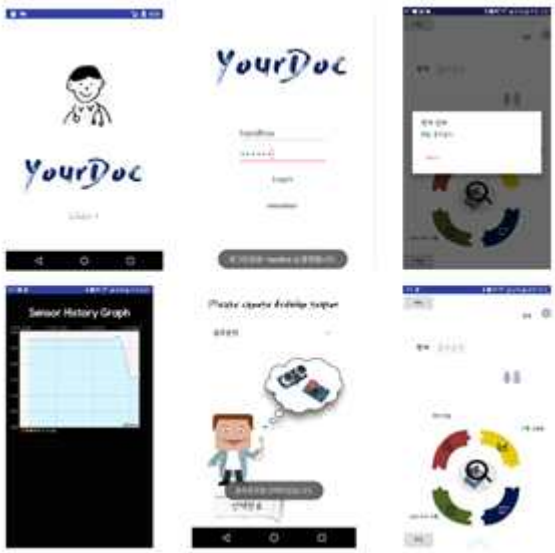
[그림 2 Dataflow]

구체적인 데이터 전송 과정은 다음과 같다. 사용자가 앱에서 원하는 센서를 선택하면 센서의 데이터가 아두이노에 들어온다.

이 수치들은 블루투스 통신을 통하여 앱 내부에 저장되게 되는데 이때 사용자가 선택한 센서에 따라 각각의 고정된 센서 아이디가 할당된다. 할당된 아이디와 함께 센서 값은 VALUE 라는 DB 테이블에 저장되게 되고 사용자는 DB에 저장된 값을 통해 다양한 기능들을 이용할 수 있다. 즉, 측정한 값들을 이용해 데이터의 특징을 파악하여 측정값의 센서 종류를 알맞게 정하고 해당되는 데이터 제공 서비스(로그, 그래프, 알람)에 따라 데이터를 처리하여 사용자에게 필요한 상태와 정보들을 보여준다.

IV.구현

4.1 애플리케이션 구현



[그림 3 애플리케이션 구성도]

4.1.1 사용자 중심 인터페이스

센서를 통해 받아온 데이터들은 다양한 형태로 사용자에게 보여지는데 기록을 잴 때마다 하나씩 로그블록이 쌓이는 "나의 수치 기록"과 변화 추이를 한눈에 볼 수 있는 "기록그래프"가 있다.

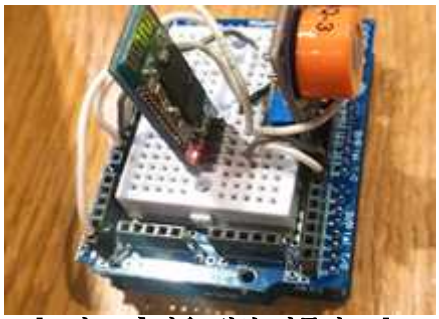
사용자에게 위치정보 권한을 요구한 뒤에 접근을 허용하면 애플리케이션의 위치가 보이고 이는 사용자가 위급하다고 여길 시에 저장된 번호로 위치정보를 담은 SOS문자를 전송한다.

센서 변경은 스피너를 사용하여 손쉽게 변경이 가능하고, 센서를 선택(혹은 변경)과 측정직후에 우선적으로 현재 상태와 상태별 조치를 권고하는 팝업 알림이 뜬다. 이는 사용자에게 경각심을 주기 위함이다.

4.1.2 데이터베이스

사용자의 데이터를 안전하게 관리하기 위해 앱 내에 데이터베이스를 구축하여 그 안에서 관리한다.

4.2 아두이노 구현



[그림 4 측정을 위한 아두이노]

현재 시장 조사 결과, 같은 종류의 센서라 하더라도 제조업체마다 달라 시중에 있는 모든 센서를 호환하기 현실적으로 힘들다는 문제점을 발견했다. 따라서 본 연구에서는 최소한의 요구사항을 설정하여 많은 센서를 호환할 수 있도록 하였다. 요구사항은 다음과 같다.

1. 같은 전압을 사용할 것
2. 입력신호가 동일할 것
(디지털신호 또는 아날로그신호)
3. 저항 회로구성이 유사할 것
4. 아날로그 0번핀을 사용할 것.

기능확장을 위해 아두이노 쉴드를 이용한다. 또한 4핀 짜리 핀 헤더 소켓을 연결하여 모든 센서를 통용할 수 있도록 하였다. 이를 통해 저항, 볼트, 아날로그/데이터 등을 다룰 수 있는 발판을 만들었다.

V.결론

본 연구에서는 현재 의료분야 헬스케어 어플리케이션이 발전이 더딘 원인인 비용적인 측면, 보안성과 편의성을 보완한 어플리케이션 개발을 통해 이용을 활성화 시키는 데에 목적을 둔다.

비용적인 면에서 여러 의료 센서를 사용할 수 있기 때문에, 새로운 센서가 등장해도 매번 다른 애플리케이션을 다운받을 필요가 없어 비용 절감을 할 수 있다.

이 애플리케이션은 사용자가 수치를 눈으로 확인하기 쉬운 인터페이스를 제공한다. 실시간으로 센서에서 받아오는 수치들을 가공해 변화하는 것을 로그블록과 그래프를 통해 확인할 수 있다. 추후에 수치 관련 다른 기능이 필요할 시 수식만 새롭게 작성하면 얼마든지 다른 기능을 구현할 수 있어 편의성을 증가시킨다.

의료 정보는 모든 사용자에게 가장 예민한 부분이므로 사용자만 데이터를 저장할 수 있도록 앱 내부에 데이터베이스를 구축하였다. 따라서 APK 파일이 지워지면 데이터베이스도 같이 사라짐으로 보안성을 높인다.

마지막으로 이 애플리케이션은 하나의 플랫폼을 가진 소프트웨어이므로, 스마트 폰에 내장된 센서 혹은 애플리케이션이 허용 가능한 외부 연결 센서를 사용하고자 하는 회사는 이 애플리케이션을 통해 고객에게 데이터 정보를 제공할 수 있어 가용성과 효율성을 높인다.

[참고문헌]

[1]https://home.kpmg.com/content/dam/kpmg/kr/pdf/kr-issuemonitor79-smart-healthcare-201801.pdf

[2]한국보건산업진흥원 연구결과보고서 (2018.04.26)

[3]산업통상자원부 '새 정부의 산업정책 방향' (2017.12)

체크리스트 기반 소프트웨어 설계보안 연구 : 소프트웨어 개발보안제도를 중심으로

성선화[1*], 최진영[2#]
고려대학교 정보보호대학원 사이버보안학과

A Study of Checklist based Secure Design for Secure Coding Policy

Sun-Hwa Sung, Jin-Young Choi
Graduate School of Information Security, Korea University

요 약

보안침해사고로 인한 공공의 피해와 확산 방지를 위해 2012년부터 SW개발보안제도를 시행하고 설계단계 및 구현단계에서 SW개발보안 적용을 의무화하여 공공 정보시스템의 시큐어 SW구축 확산에 도움을 주고 있다. 하지만 일부 공공 정보시스템만 의무적 적용 대상이며, SW개발보안 진단인력 부족, SW보안약점 진단원 역할의 한계 등으로 제도 확산에 어려움이 있다. 본 논문에서는 SW 보안취약점 발생예방과 개인정보보호 방안을 정보시스템 설계단계에서부터 함께 고려되어야 할 중요사항임을 인식하고, SW개발보안제도와 개인정보보호제도에서 요구하는 보안요구사항을 접목하여 설계보안 진단체크리스트를 구성하였고 실제 설계 산출물을 이용하여 활용가능성 및 설계보안 진단 효과를 평가하였다. 본 논문에서 제안하는 체크리스트는 설계단계의 SW개발보안 활용에 도움이 될 것으로 생각한다.

I. 서론

1. 연구 배경

설계보안(Secure by design)은 시스템 구축 시 보안을 우선적으로 고려해 설계한다는 의미이다. 국내외 대규모 정보유출사고 발생 이후 설계보안의 중요성에 대해 부각됐고, 지금은 IT 업계뿐 만 아니라 전 산업군에서 일반적인 설계 트렌드로 자리 잡아 가고 있다.

소프트웨어(SW) 개발보안은 보안약점 및 보안취약점 발생 최소화를 위해 소프트웨어개발 전 과정에서 실행되어야 하는 일련의 보안강화 활동을 의미이며, 최근 사이버보안 공격의 약 75%가 소프트웨어 보안취약점을 노린 것으로 분석되고 있어 SW개발보안이 적용된 시큐어 SW 구축의 중요성은 더욱 높아지고 있다. 2012년 행정자치부에서 『행정기관 및 공공기관 정보시스템 구축 운영지침』을 통해 SW개발 시 소스코드의 보안약점 제거를 의무화하는 SW개발보안제도를 시행하였고 SW보안약점 진단원 제도 도입 및 인력 양성, 시큐어코딩 관련 교육 지속적 실시, SW개발보안 가이드 배포 등 SW개발보안의 확산을 위해 노력하고 있다. 2016년 말 소스코드의 보안약점 제거만으로는 한계가 존재하여 설계단계까지 SW개발제도를 확대하여 시행중에 있다. 이는 국내에서도 설계보안에 대한 중요성을 SW개발보안제도에 반영한 결과라고 볼 수 있다.

최근 행정안전부 개인정보보호위원회에서는 정보보호주체 권리 보장을 위해 제품 및 서비스 설계단계에서부터 개인정보보호에 최적화된 설계를 의미하는 Privacy by design(이하 "개인정보보호 최적화설계")을 고려하도록 적용하는 방안을 검토하고 있는데, 개인정보 활용목적, 범위, 기록유지, 안전조치, 비식별화 등의 보호조치를 제품 및 서비스 설계단계부터 고려하여야 한다는 의미이다.

본 논문에서는 설계보안과 개인정보보호 최적화설계를 함께 접목하여 정보시스템 구축의 설계단계에서 반드시 고려되어야 하는 SW개발보안 적용 항목에 대한 연구를 진행하고, 그 결과물로 설계보안 진단 체크리스트를 구성, 현업에서 설계 산출물 점검 및 진단 등에 활용할 수 있도록 제안하고자 한다.

II. 본론

1. 설계보안 진단 체크리스트의 필요성

첫째, 보안은 개발초기부터 고려되어야 효과적이다. 구축초기부터 단계별 필요한 정보보호 조치를 식별·계획하고 이를 이행한다면 보안위협 최소화 및 보안수준을 크게 향상시킬 수 있다. 또한 미국 국립표준기술연구소(NIST) 연구결과에 의하면, 설계단계에서 발생한 결함을 운영단계에서 발견하고 제거하는 비용은 설계단계에서 수정하는 비용보다 60배 이상 증가하므로 개발초기부터 보안을 고려한다면 취약점 조치비용을 크게 절감시키는 효과도 가져 올 수 있다.

둘째, 정보시스템 구축 설계단계의 SW개발보안 적용 여부를 점검하고 결과 정량이 가능한 점검도구가 필요하다. 구현단계의 SW개발보안 적용여부 점검을 위해 다양한 정적·동적 분석도구들이 개발·배포되고 있는 것에 비해, 설계보안 적용여부를 점검하고 진단할 유용한 도구가 없는 것이 현실이다.

셋째, 정보시스템 개발초기부터 고려되어야 할 다양한 보안요구항목을 설계보안 진단항목에 접목할 필요가 있다. 특히 최근 EU의 GDPR 발의 등으로 그 어느 때보다 중요성이 부각되고 있는 개인정보보호를 위한 최소한의 기술적·물리적 보호조치도 함께 고려하여 비교적 안전한 시스템 구축을 위한 노력과 글로벌 침해사고에 대한 대비도 필요하다.

이와 같은 필요성에 따라, 설계단계의 보안요구항목 적용여부를 쉽게 점검·진단할 수 있도록 설계보안 진단체크리스트를 구성하여 제안한다.

2. 설계보안 진단 체크리스트의 구성

현재 국내에서는 개인정보보호법, 정보통신망법, 금융거래법 등 다양한 법, 법령, 관련지침에서 중요정보 식별 및 정보보호 방안에 대해 규정하고 있으며, 이 법규 사항 중 설계단계에서 의무적으로 고려되어야 할 보안요구사항들이 다수 있다.

본 논문에서 제안한 설계보안 진단 체크리스트는 전자정부 SW개발·운영자를 위한 소프트웨어 개발보안 가이드[2], 개인정보보호 자율점검 가이드라인[3] 및 The IEEE Center for Secure Design에서 발표한 설계보안 결함유발항목 Top10[4]을 기초자료로 하여 세부점검항목을 구성하였으며, 진단 단위별로 시스템 단위 진단체크리스트와 프로그램 단위 진단체크리스트로 분류하였다.

시스템 단위 진단체크리스트는 시스템 전체적으로 영향력을 끼치는 보안요구항목 진단을 위해서다. 주로 중요정보 관리정책, 공통 라이브러리·API 도입, 접근권한설계, 세션통제 등에 대한 진단 항목들로 구성되어 있다. [Table 1 참고]

Table 1. 시스템 단위 진단체크리스트

관련법	보안요구항목의 부문	세부점검사항
합계		49개
공공 정보시스템구축 운영지침	- 입력데이터 검증·표현 - 보안기능 - 세션통제 - 에러처리	8 17 10 4
개인정보 보호법	- 개인정보파기 - 안전조치의무	2 7
기타	- 외부API 보안영향	1

프로그램 단위 진단체크리스트는 각 프로그램마다 준수하여야 할 보안요구항목 진단을 위해서이며, 입출력값 검증, 인증수행 제한, 중요정보 저장 및 전송 등에 대한 진단 항목들로 구성되어 있다. [Table 2 참고]

Table 2. 프로그램 단위 진단체크리스트

관련법	보안요구항목의 부문	세부점검사항
합계		37개
공공 정보시스템구축 운영지침	- 입력데이터 검증·표현 - 보안기능 - 세션통제	22 14 1

표 [Table 3]은 진단체크리스트의 내용 중 일부를 발췌한 것이다.

Table 3. 진단체크리스트 상세내용

보안요구항목	세부점검항목
DBMS 조회 및 결과 검증	애플리케이션에서 DB연결을 수행시 최소권한의 계정을 사용하는가?
입력 및 출력 값 검증	중앙 집중식(입력값 필터, MVC인터셉트,Validator컴포넌트) 입력값/출력값 검증 메커니즘을 적용하였는가?
접근권한 관리 및 접근 통제	전보·퇴직 등 인사 이동으로 취급자가 변경 될 경우 접근 권한 변경 또는 말소되도록 설계하였는가?
개인정보 파기	보유기간 경과, 처리 목적 달성 후 지체 없이(보유기간 종료일로부터 5일 이내) 개인정보가 파기되도록 설계되었는가?
보안기능 동작에 사용되는 입력값검증	클라이언트에서 수행하는 외부입력값 검증작업을 서버측의 비즈니스 로직에서 재검증을 실시하였는가?
업로드·다운로드 파일검증	파일 다운로드 요청전, 악성코드 또는 바이러스 검사 수행뒤 다운로드 되도록 설계하였는가?
인증 수행 제한	로그인 시도횟수를 3~5번으로 제한하였는가?

3. 체크리스트 활용한 진단 평가

설계보안 진단체크리스트의 실제 활용가능 여부 및 효과성 평가를 위해 A사의 사업비 규모 5천만~3억원 사이의 설계 산출물 3개를 선정하고, 선정된 설계 산출물 중 화면설계서, ERD, 개발표준정의서를 참고하여 설계보안 진단을 수행하였으며, PM 및 시스템운영자와의 개별인터뷰로 일부 항목을 보완하였다. 정보시스템 별로 시스템 단위 진단체크리스트 이용하여 각 1회씩 진단했으며, 전체 349개 메뉴에 대해 프로그램 단위 진단체크리스트로 진단을 수행하였다. [Table 4 참고]

Table 4. A사 보안요구항목별 설계보안 준수율

관련법	부문	보안요구항목	준수율
평균			74%
공공정보시스템구축운영지침	입력 데이터 검증 및 표현	DBMS조회 및 결과검증	100
		입력 및 출력값 검증	67
		디렉터리 서비스 조회 및 결과 검증	
		웹 서비스 요청 및 결과 검증	99
		웹 기반 중요기능 수행 요청 유효성검증	50
		허용된 범위 내 메모리 접근	-
		업로드·다운로드 파일검증	74
		XML 조회 및 결과 검증	100
		시스템자원 접근 및 명령어 수행 입력값 검증	-
		HTTP프로토콜 유효성 검증	-
		보안기능 동작에 사용되는 입력값 검증	10
	보안기능	중요정보 저장	16
		인증 대상 방식	100
		인증 수행 제한	30
		비밀번호 관리	97
		중요 정보 전송	67
		중요자원 접근 통제	93
		암호키 관리	100
		암호연산	100
	예외처리	예외처리	83
	세션통제	세션 통제	79
개인정보보호법	안전조치 의무	접근권한 관리 및 접근 통제	100
	개인정보 파기	개인정보파기	100
기타	외부API 보안	외부API의 보안영향 최소화	0

4. 진단결과 평가

A사 진단을 완료한 결과, 보안요구사항에 대한 평균 준수율은 시스템 단위 평가에서는 83%, 프로그램 단위 평가에서는 61% , 전체 74%로 나타났으며, 진단에서 파악된 주요 사항은 아래와 같다.

누락되는 보안요구항목은 전 프로그램에 걸쳐 반복적으로 누락되는 패턴을 보였으며, 시스템 구축 시 전자정부프레임워크와 중앙집중식 필터 등을 적극 활용한 결과 SQL삽입, XML삽입, 크로스사이트스크립트(XSS) 공격 등의 잘 알려진 입·출력 값에 의해 발생할 수 있는 보안취약점은 비교적 잘 대비하고 있었고 개인정보보호를 위한 시스템 요구사항도 설계단계에서부터 잘 반영하고 있었다.

주요 준수율이 낮은 항목은 외부 입력값의 서버측 재검증, 중요정보 저장 시 메모리 초기화 및 웹캐싱 방지, 인증수행 제한, 패스워드 변경 후 세션 재할당, 외부API 보안영향 검증 등 이었다. A사의 낮은 준수율을 보이는 항목들은 대부분 논리적 오류에 해당하는 것이라 구현단계에서 사용되는 자동진단도구에서도 검출되기가 어려우므로 설계단계에서부터 결함발생을 방지하는 것이 중요하다.

결론적으로, 만일 A사가 진단 체크리스트를 활용하여 설계보안 진단을 수행하고 발견된 보안 결함 26%를 보완하였다면 이후 단계에서 수정 시 발생할 수 있는 많은 시간소요와 비용을 크게 절감시키고 보안취약점 발생을 미연에 방지하는 기회를 가졌을 것이다.

III. 결론

1. 연구 성과

본 논문은 설계보안 진단 체크리스트를 제안하고 평가하여 그 성과를 측정하였으며, 측정된 성과는 다음과 같다.

첫째, 제시된 설계보안 진단 체크리스트에는 소프트웨어 개발보안제도에서 제시하는 분석·설계단계의 SW개발보안 요구항목 뿐만 아니라 개인정보보호법에서 규정하고 있는 보안요구항목 중 시스템 개발초기부터 반드시 고려되어야 하는 항목들도 추가하여, 국내 주요 정보보호제도에서 제시한 보안요구사항을 설계단계에서부터 포괄적이고 체계적으로 점검하고 진단하는데 활용할 수 있다.

둘째, 구축중인 정보시스템의 설계업무 담당자가 설계보안 적용여부를 자체 점검해 볼 수 있어 구현단계로 진입하기 전에 보안 결함을 보완할 수 있는 기회를 제공한다. 특히 SW개발보안제도 비의무 대상인 정보시스템 구축·운영 시에도 자체 설계단계 설계보안 진단도구로 활용하여 일정수준의 설계보안 적용을 달성하도록 지원할 수 있다.

셋째, 정보보호 관련제도 등에서 제시하고 있는 보안요구항목을 86개 세부점검사항으로 상세히 도출하여, 보안전문가 수준의 지식을 보유하고 있지 않은 현업담당자여도 설계단계의 보안요구항목 적용 여부를 점검하거나 진단 시에 활용할 수 있다.

2. 연구 결론

본 논문에서 제시한 진단 체크리스트는 소프트웨어 개발보안제도에 개인정보보호제도의 보안요구항목을 접목하였고, 보안전문가 수준의 지식을 보유하지 않은 현업 담당자도 어렵지 않게 설계단계의 설계보안 이행여부를 점검하고 진단에 활용할 수 있도록 상세히 점검항목을 구성하였으며, 이의 활용성 및 효과성 확인을 위한 평가도 완료하였다.

결론적으로, 본 논문의 진단 체크리스트는 현업의 설계담당자에게 설계보안 적용 시 구체적 이행방안을 제시하는 좋은 길잡이가 될 것으로 예상하며, 이러한 시도는 설계단계부터 SW개발보안 및 개인정보보호 최적화설계 적용의 활성화를 유도하고 각종 정보보호 제도와 글로벌 환경변화에 대한 능동적 대처와 시큐어 SW구축 확산에 도움을 줄 수 있을 것으로 생각한다.

[참고문헌]

[1] 과학기술정보통신부·한국인터넷진흥원, "정보보호 사전점검 안내서", 2018

[2] 행정안전부, "전자정부 SW 개발·운영자를 위한 소프트웨어 개발보안 가이드", 2017

[3] 행정안전부, "개인정보보호 자율점검 가이드라인", 2015

[4] https://cybersecurity.ieee.org/center-for-secure-design

[5] 행정안전부, "공개SW를 활용한 소프트웨어 개발보안 검증가이드", 2016

[6] 김영숙, "공공 정보시스템 개발 설계산출물에 대한 SW개발보안적용 진단을 위한 연구", 동국대학교, 2017

[7] 과학기술정보통신부·소프트웨어정책연구소, "2017 소프트웨어산업 연간보고서", 2017

[8] 국가정보원·과학기술정보통신부·방송통신위원회·행정안전부·금융위원회, "2018 국가정보보호백서", 2018

DexGuard 난독화 도구 특성 분석*

김규호, 신용구, 이선준, 이정현
숭실대학교 소프트웨어학부

Feature Analysis of DexGuard Obfuscation Tool

Kyu-Ho Kim, YongGu Shin, SunJun Lee, and Jeong Hyun Yi
School of Software, SoongSil University.

요 약

모바일 앱의 활성화에 비례하여 모바일 악성코드가 빠른 속도록 증가하고, 그 중에서도 안드로이드를 대상으로 한 악성코드는 기하급수적으로 증가하고 있다. 최근에는 이러한 악성코드가 백신에 탐지되지 않도록 난독화를 적용하는 경우가 대부분이다. 이에 대비하여 악성코드 분석하기 위해서는 난독화 해독을 위한 효율적인 역난독화 기술 개발이 시급한 실정이다. 따라서 본 논문에서는 역난독화 기술 개발에 선행하여 현재 상용화되어 있는 소스코드 난독화 도구인 DexGuard를 대상으로 난독화 기법별로 세부 기능을 분석하고자 한다. 이를 통해, 향후 난독화가 적용된 악성코드를 자동 분석할 수 있는 기반이 마련될 것으로 기대한다.

I. 서론

현대 사회에서는 수많은 모바일 장비들을 활용하여 다양한 작업을 할 수 있다. 그중에서도 안드로이드 OS의 시장 점유율은 근래에 86%에 육박한다. 이에 따라 많은 사람들이 안드로이드 앱 시장을 사용한다.[1]

안드로이드 앱 시장은 제품 출시가 자유로운 정책을 가지고 있기 때문에 많은 제품들이 쉽게 출시될 수 있지만 출시된 제품을 통해 사용자의 정보가 유출되고 기기의 안정성이 저하되는 등, 악성코드가 확산되기 쉽다. 특히, 출시가 자유롭다는 점과 디컴파일을 통한 분석이 용이한 Dalvik Byte Code의 특징을 악용하여 기존의 코드를 분석하여 중간에 악성행위를 하는 코드를 삽입한 뒤 다시 재배포하는 리패키징 공격이 많이 발생하고[2] 이를 방지하기 위해 기존의 코드를 분석하기 어렵게 만드는 코드 보호 기법[3]이 등장했다.

그러나 이런 코드 보호 기법의 분석을 어렵게 한다는 점을 악용하여 악성코드의 악성행위를 숨기기 위해 활용한 지능형 악성코드가 등장하였고 이에 따라 악성코드에 대한 대응이 점차 느려졌다.[5]

본 논문에서는 이러한 코드 보호 기법 중에서 분석 속도를 저하시키는 난독화 기법에 대해 분석하기 위해 상용 소스코드 난독화 도구 중 하나인 DexGuard[6]를 대상으로 세부 기능에 대해 분석한다.

II. 배경 지식

2.1 Java Reflection API

Reflection은 프로그램이 동적으로 자기 자신의 구조 및 동작을 관리, 수정하는 것을 의미하며, 이를 통해 프로그램은 컴파일 단계에서 알 수 이름을 알 수 없는 인터페이스, 필드, 메소드에 대해 실행 중에 동적으로 조작하는 것이 가능하다.

Android의 Virtual Machine인 Dalvik Virtual Machine[7]은 Java의 Virtual Machine을 기초로 설계되었기 때문에 Java에 존재하는 Reflection 기능을 모두 갖추고 있다. 이를 활용한다면 메소드를 코드에서 직접 호출하지 않고 실행 중에 클래스의 구조를 분석한 뒤 필요한 메소드를 추출하여 간접적으로 호출할 수 있다.

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

2.2 코드 보호 기법 분류

코드를 대상으로 역공학 분석을 어렵게 만드는 기술이다. 분석기법이 적용되는 시점에 따라 정적 보호 기법과 동적 보호 기법으로 나뉘게 된다[4]. 정적 보호 기법에는 난독화, 패킹, 위변조 탐지 등의 기법이 있으며, 동적 보호 기법에는 루팅 탐지, 에뮬레이터 탐지, 디버거 탐지 등의 기법이 있다. 본 논문에서는 소스코드를 기반으로 한 정적 보호 기법인 소스코드 난독화 기법을 대상으로 한다.

III. 난독화 기법별 기능 분석

난독화 기법은 코드 보호 기법 중 정적분석을 어렵게 만들기 위한 기법이다. DexGuard는 안드로이드 플랫폼에서 활용되는 Dalvik Byte Code에 대한 난독화 기능을 제공한다.

본 논문에서는 DexGuard의 난독화 기법 중, 소스코드의 정적분석을 방해하는 기법들만을 다룬다. 난독화 기법들을 식별자를 변조하는 Renaming, 메소드 호출부를 변조하는 API Hiding, 문자열 데이터 값을 변조하는 String Encryption, Class 데이터를 변조하는 Class Encryption으로 분류하고자 한다.

분석은 Dalvik Byte Code를 decompile하여 나타난 smali 코드를 바탕으로 분석하였다.

```
.method public compareKey(String)Z
        .registers 4
const/4  v0, 0
const-string  v1, "Hello World!"
if-ne  p1, v1, :1C
new-instance  v1, SecretClass
invoke-direct  SecretClass-><init>()V,
v1
invoke-virtual
SecretClass->secretMethod(String)V,
v1, p1
const/4  v0, 1
return  v0
.end method
```

[그림 1] 난독화 적용 이전 Smali 코드

3.1 Identifier Renaming

Identifier Renaming 기법은 정적 코드 분석에 큰 도움이 되는 변수 명, 클래스 명, 메소드 명 등을 의미정보를 포함하지 않는 값으로 변경하여 가독성을 떨어뜨린다. Renaming 기법의 경우 가독성에는 영향을 주지만 코드 흐름에는 변화를 주지 않기 때문에 복호화 과정이 존재하지 않는다. DexGuard를 포함한 거의 모든 난독화 도구에서 기본적으로 제공해준다.

[그림 2]에서 확인할 수 있듯이 클래스나 메소드의 식별자가 'T', '''등으로 변경되었을 뿐 실행 흐름상에는 변화가 없음을 확인할 수 있다.

```
.method public ´(String)Z
        .registers 4
const/4  v1, 0
const-string  v0, "Hello World!"
if-ne  p1, v0, :1C
new-instance  v0, T
invoke-direct  T-><init>()V, v0
invoke-virtual  T->´(String)V, v0, p1
const/4  v1, 1
return  v1
.end method
```

[그림 2] Identifier Renaming 적용이후 Smali 코드

3.2 API Hiding

API Hiding 기법은 Java Reflection API를 활용하여 정적 코드 분석 시에 어떤 API가 호출되는지에 대한 분석을 어렵게 한다. JAVA Reflection API의 한계로 인해 문자열 데이터를 활용하여 분석을 할 수 있다.

```
.method public ´(String)Z
        .registers 10
const/4              v7, 0
const-string          v0, "Hello World!"
new-instance          v1, SecretClass
invoke-direct
SecretClass-><init>()V, v1
const/4              v0, 1
new-array             v2, v0, [Object
const/4              v0, 0
aput-object           p1, v2, v0
const-string          v0, "o.SecretClass"
invoke-static
Class->forName(String)Class, v0
move-result-object  v0
const-string          v3, "SecretMethod´"
const/4              v4, 1
new-array             v4, v4, [Class
const-class           v5, String
const/4              v6, 0
aput-object           v5, v4, v6
invoke-virtual
Class->getMethod(String,
[Class)Method, v0, v3, v4
move-result-object  v0
invoke-virtual
Method->invoke(Object, [Object)Object,
v0, v1, v2
const/4              v7, 1
return               v7
.end method
```

[그림 3] API Hiding 적용이후 Smali 코드

[그림 3]의 예시코드는 짧기 때문에 위에서 언급한 JAVA Reflection API의 한계점을 쉽게 확인할 수 있다. 그러나 이러한 한계점도 Renaming 기법이나 String Encryption 기법, Class Encryption 기법 등과 함께 쓰인다면 분석에 많은 어려움을 줄 수 있다.

3.3 String Encryption

String Encryption 기법은 코드에 존재하는 문자열 데이터를 암호화하는 난독화 기법으로 데이터 호출 시에 지정된 복호화 메소드를 통해 데이터를 복호화

하여 활용한다. 내부에 새로이 복호화 메소드가 추가되는 것을 확인할 수 있다.

```
.method public compareKey(String)Z
        .registers 6
const/4            v3, 0
const/4            v0, 0
const/4            v1, 0
const/4            v2, 0
invoke-static       MainActivity->$(B,
B, I)String, v0, v1, v2
move-result-object  v0
invoke-virtual
String->intern()String, v0
move-result-object  v0
if-ne              p1, v0, :2E
:1C
new-instance        v0, SecretClass
invoke-direct
SecretClass-><init>()V, v0
invoke-virtual
SecretClass->secretMethod(String)V, v0,
p1
const/4            v3, 1
:2E
return             v3
.end method
```

[그림 4] String Encryption 적용이후 Smali 코드

```
.method private static $(B, B, I)String
        .registers 9
mul-int/lit8        p2, p2, 4
rsub-int/lit8       p2, p2, 72
mul-int/lit8        p0, p0, 3
rsub-int/lit8       p0, p0, 12
mul-int/lit8        p1, p1, 4
rsub-int/lit8       p1, p1, 4
sget-object         v4,
MainActivity->$:[B
const/4            v5, 0
new-instance        v0, String
new-array          v1, p0, [B
if-nez             v4, :38
move               v2, p0
move               v3, p1
:2E
add-int/lit8        p1, p1, 1
add-int/2addr       v2, v3
add-int/lit8        p2, v2, -12
:38
move               v2, v5
add-int/lit8        v5, v5, 1
int-to-byte         v3, p2
aput-byte          v3, v1, v2
const/4            v2, 0
invoke-direct       String-><init>([B,
I)V, v0, v1, v2
return-object       v0
:52
move               v2, p2
aget-byte          v3, v4, p1
goto               :2E
.end method
```

[그림 5] 복호용 메소드의 Smali 코드

[그림 4]를 통해 "Hello World!"라는 문자열 데이터가 호출된 시점에서 복호화 메소드인 $가 호출됨을 알 수 있다. String Encryption의 경우 실행에 활용되는 데이터 값을 변조했기 때문에 정상적인 실행을 위해서는 반드시 복호화하는 과정을 필요로 한다. 이를 위해 [그림 5]와 같은 복호용 코드가 반드시 포함되어 있다.

3.4 Class Encryption

Class Encryption 기법은 Class를 암호화하는 기법이다. 암호화된 Class는 정적 파일구조상에서는 확인이 힘들며 복호화클래스가 추가되고 암호화된 클래스의 호출부에 복호화클래스가 들어가게된다.

앱 실행 시 동적으로 복호화하는 클래스에서 클래스로더를 통해 암호화되어 있던 클래스가 빌드되는 것을 확인할 수 있다.

```
.method public ´(String)Z
        .registers 10
const/4            v7, 0
const-string        v0, "Hello World!"
const-string        v0, "o.T"
invoke-static       T$ˈ->´(String)Class,
v0
move-result-object  v0
const/4            v1, 0
invoke-virtual
Class->getDeclaredConstructor([Class)C
onstructor, v0, v1
move-result-object  v0
const/4            v1, 0
invoke-virtual
Constructor->newInstance([Object)Objec
t, v0, v1
move-result-object  v1
const/4            v0, 1
new-array           v2, v0, [Object
const/4            v0, 0
aput-object         p1, v2, v0
const-string        v0, "o.T"
invoke-static       T$ˈ->´(String)Class,
v0
move-result-object  v0
const-string        v3, "´"
const/4            v4, 1
new-array           v4, v4, [Class
const-class         v5, String
const/4            v6, 0
aput-object         v5, v4, v6
invoke-virtual
Class->getMethod(String,
[Class)Method, v0, v3, v4
move-result-object  v0
invoke-virtual
Method->invoke(Object, [Object)Object,
v0, v1, v2
const/4            v7, 1
return             v7
.end method
```

[그림 6] Class Encryption 적용이후 Smali 코드

[그림 6]에서는 'T$ˈ'라는 클래스를 통해서 'T'클래스에 접근하는데 'T'클래스는 Renaming된 SecretClass이며 ClassLoader를 통해서 간접 접근하기 때문에 정적분석을 통해서는 이 클래스에 대한 정보를 확인하기 어렵다.

IV. 결론

본 논문에서는 안드로이드 기반의 소스코드 난독화 기법을 제공해주는 DexGuard라는 도구의 난독화 기능에 대해 분석하였다. 이러한 기능에 대한 분석은 코드 보호 기법이 적용된 지능형 악성코드를 자동으로 분석하기 위해 선행적으로 수행하여야 하는 역난독화 기법 개발에 많이 도움이 될 것으로 기대된다.

[참고문헌]

〔1〕 https://www.gartner.com/en/newsroom/press-releases/2018-05-29-gartner-says-worldwide-sales-of-smartphones-returned-to-growth-in-first-quarter-of-2018

〔2〕 Jin-Hyuk Jung, Ju Young Kim, Hyeong-Chan Lee, and Jeong Hyun Yi, "Repackaging Attack on Android Banking Applications and Its Countermeasures",Wireless Personal Communications, Vol.73, No.4, pp.1421-1437, Dec. 2013.

〔3〕 Haehyun Cho, Jongsu Lim, Hyunki Kim, and Jeong Hyun Yi, "Anti-Debugging Scheme For Protecting Mobile Apps on Android Platform", Journal of Supercomputing, Vol.72, No.1, pp. 232-246, Jan. 2016.

〔4〕 Jongsu Lim, Yonggu Shin, Sunjun Lee, Kyuho Kim and Jeong Hyun Yi, "Survey of Dynamic Anti-Analysis Schemes for Mobile Malware", Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, Vol.9, No.3, pp. 39-49, Sep. 2018.

〔5〕 FireEye, M-Trends Reports: M-Trends 2018, The Trends Behind Today's Breaches and Cyber Attacks

〔6〕 https://www.guardsquare.com/en/products/dexguard

〔7〕 https://source.android.com/devices/tech/dalvik

의료정보 공유 플랫폼을 위한 하이퍼레저 패브릭 응용 방안

이민기*, 민서현*, 지청민**, 홍만표*

*아주대학교 사이버보안학과, **아주대학교 정보컴퓨터공학과

Medical information sharing platform architecture based on Hyperledger Fabric

Minki Lee*, Seohyeon Min*, Cheongmin Ji**, and Manpyo Hong*

*Dept. of Cyber Security, Ajou University, **Dept. of Computer Engineering, Ajou University

요 약

정보통신 기술의 발전에 따라 데이터의 양이 많아지고 이를 공유하여 활용하고자 하는 연구가 활발히 진행되고 있다. 특히 의료분야에서는 의료정보를 활용하여 병원 간의 의료정보 공유 및 병원과 연구소 간의 의료정보 공유를 통해 환자의 진료비를 절감과 연구소의 의료정보를 공유를 통한 수월한 연구를 기대하고 있다. 하지만 의료정보 공유는 개인정보보호법, 의료법에 저촉되는 사안과 데이터 무결성, 외부로부터 들어올 수 있는 보안 취약점 때문에 쉽게 구현하기 어려운 실정이다[1]. 이를 블록체인을 통하여 해결하려고 할 때 데이터 무결성과 보안 취약점은 해결할 수 있으나 블록체인에 기록된 데이터는 삭제를 하지 못하기 때문에 개인정보 삭제권에 관한 법률이 저촉되는 문제점이 발생한다. 따라서 본 논문에서는 하이퍼레저 패브릭의 private data collection을 이용하여 분산된 데이터베이스, 데이터 무결성, 데이터 기밀성, 데이터 삭제권을 보장하는 플랫폼을 구상하였다.

I. 서론

현재 의료기관은 시스템 간 진료정보 교류가 제한돼 있다. 환자가 병원을 옮길 때마다 CD나 문서 등의 형태로 다시 제출해야 한다. 2014년 기준 CT, MRI 중복 촬영으로 월 평균 16억원이 낭비된다[1]. 이러한 문제를 해결하기 위해 의료정보를 공유하려고 하는 시도를 하고 있다. 예를 들어 메디블록[2]과 같이 환자의 개인정보 주체를 보장하면서 환자, 의료공급자, 의료 연구자에게 자유로운 데이터 공유를 보장하기 위한 블록체인 플랫폼이 존재한다.

블록체인은 데이터 무결성을 위해 데이터를 분산화하고 정해진 알고리즘의 합의에 따라 유지하고 동기화하는 시스템을 말한다. 블록체인 네트워크 참여에 제약이 없는 시스템을 비허가형 또는 퍼블릭 블록체인이라 하고 블록체인 네트워크 참여 시 검증을 받는 과정이 있다면 허가형 또는 프라이빗 블록체인이라 한다. 비트코인, 이더리움과 같은 시스템은 비허가형 블록체인, 하이퍼레저 패브릭, R3가 있다. 금융의 탈중앙화를 위해 나온 것이 블록체인의 시초였지만 블록체인의 장점을 통해 다양한 시스템으로 파생되고 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 의료정보의 공유 시 발생하는 개인정보보호법과 의료법에 대하여 살펴보고 이를 해결하는 방안의 예시인 블록체인 플랫폼 하이퍼레저 패브릭을 살펴본 다. Ⅲ장에서 하이퍼레저 패브릭 기반 의료정보 공유 시스템을 설명하고 Ⅳ장에서 결론과 추후연구 방안을 서술한다.

II. 배경

2.1 의료정보 공유

정보통신의 기술이 발전함에 따라 많은 데이터가 생성되었고 이는 의료정보에도 영향을 끼쳤다. 한 환자에 대하여 여러 병원의 협진이 필요한 경우나 어떤 병원의 의료정보를 바탕으로 임상실험 및 연구를 유연하게 시행하기 위해 의료정보 데이터를 공유하려고 하는 움직임이 생겼다[3]. 이 때 중요한 것은 환자의 개인정보에 대한 권리를 최대한 보장하기 위해 의료정보의 기밀성과 무결성을 보장해야 하는 것과 유럽연합의 GDPR(General Data Protection Regulation)을 비롯한 개인정보보호법의 컴플라이언스 충족이다[4].

의료법 제21조 2항에 의하면 의료인, 의료기관의 장 및 의료기관 종사자는 환자가 아닌 다른 사람에게 환자에 관한 기록을 열람하게 하거나 그 사본을 내주는 등 내용을 확인할 수 있게 할 수 없다. 하지만 개인정보보호법 제18조 2항 4호에 따르면 학술연

구의 목적을 위해 개인정보를 이용하기 위해서는 특정 개인을 알아볼 수 없는 형태로 만들어 사용할 수 있다[5]. 이를 준수하기 위해 개인정보를 알아볼 수 없는 형태로 만드는 익명화 과정을 거치게 되면 데이터가 훼손되어 임상실험 및 연구에 적합하지 않게 된다[6]. 환자의 데이터를 익명화 없이 사용하기 위해서는 결국 환자에게 개인정보 활용을 동의를 받아야 한다.

본 논문에서는 환자에게 개인정보 활용 동의를 받았다는 가정 하에 각종 기관 간에 의료정보를 안전하게 공유하는 블록체인 플랫폼을 만드는 방법을 설명하고자 한다. 블록체인 기술을 도입하면 의료정보 기록의 블록체인 연결 구조로 의료 데이터의 무결성이 보장되고 분산된 데이터베이스 구조 덕분에 외부로부터 중앙 데이터베이스 공격 시 발생하는 문제점도 해결된다. 의료 정보의 이동 시 환자의 동의를 얻을 수 있도록 설계를 하면 개인정보의 주체성도 보장된다.

2.2 하이퍼레저 패브릭

하이퍼레저는 리눅스 재단에서 주도한 B2B, B2C 사업을 위해 만들어진 오픈 소스 프로젝트이며 그 하위 프로젝트로 하이퍼레저 패브릭 프레임워크가 있다. 하이퍼레저 패브릭은 신원이 확인된 클라이언트를 네트워크에 참여시키는 허가형 블록체인이다. 하이퍼레저 패브릭의 네트워크는 여러 개의 채널로 이루어지며 각 채널은 다시 별도의 네트워크를 이룬다. 이 때 각 채널끼리의 통신 내용은 서로 독립되어 있으며, 이에 따라 기밀성이 보장된다. 네트워크 참여자인 피어는 여러 채널에 참여할 수 있으며 채널에 참여하면 MSP(Membership Service Provider)를 통하여 다른 피어에게 조직(Organization)을 할당해 준다. 할당 받은 조직은 각 채널에서 규정한 채널 정책에서 권한을 준 조직이어야 참여할 수 있다.

Fig 1은 하이퍼레저 패브릭 네트워크에 들어온 피어의 거래가 어떻게 처리되는지 보여준다. 피어는 체인코드(스마트 컨트랙트)와 장부(Ledger)를 가지고 있으며 네트워크 밖의 클라이언트 애플리케이션과 연결되어 있다. 클라이언트 애플리케이션이 하이퍼레저 패브릭 채널내의 다른 피어와 거래를 하기 위해 보내는 신호인 트랜잭션을 호출하게 되면 호출을 받은 피어의 체인코드를 실행해야 하는데 이 때 실행

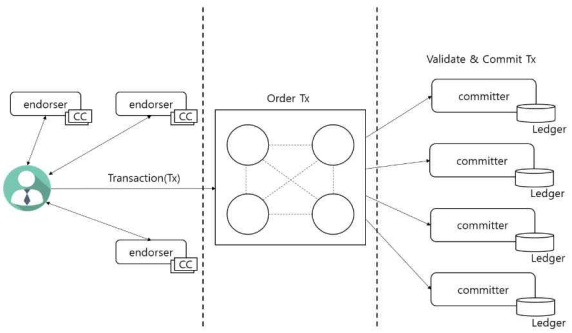


Fig 1. 하이퍼레저 패브릭 네트워크 거래처리

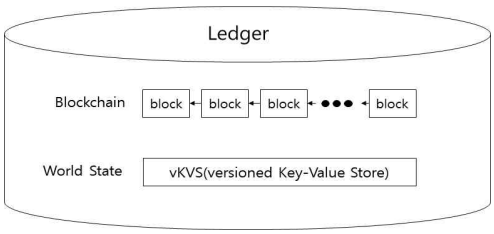


Fig 2. 하이퍼레저 패브릭 장부 구조

결과가 전체 네트워크에 끼치는 악영향을 확인하기 위해 미리 체인코드를 실행하는 역할을 하는 보증(Endorsing) 노드에게 체인코드 실행을 의뢰한다. 보증 노드는 받은 체인코드를 실행하게 되며 트랜잭션과 실행 결과를 다시 피어에게 돌려준다. 피어는 다시 이를 순서화(Ordering) 서비스에게 보내며 순서화 서비스는 다른 피어들이 실행한 서로 다른 트랜잭션들의 순서를 결정하며 마지막으로 커미터(Commiter)에게 이 결과를 보낸다. 커미터는 장부에 데이터를 쓰는데 문제가 없는지 확인한 후 모든 피어가 가진 장부에 트랜잭션 결과를 배포하게 된다.

Fig 2는 하이퍼레저 패브릭의 장부 구조를 보여준다. 피어의 장부는 블록체인과 상태(State) 데이터베이스로 이루어져 있다. 블록체인은 트랜잭션의 기록, 이전 블록의 해시 값이 저장된 블록이 서로 연결되어 있는 방식으로 이루어져 있다. 이는 생성된 모든 트랜잭션들의 무결성을 보장해준다. 상태 데이터베이스는 키-값-버전 형태로 저장되어 있다.

이러한 구조를 통해 하이퍼레저 패브릭은 특수 목적을 가진 조직들끼리 연합을 이루거나 하나의 데이터 공유 플랫폼을 만들기 적합하다. 특히 금융이나 의료정보와 같이 빠른 데이터 공유와 데이터의 기밀성이 보장되어야 하는 곳에 적합하다. 하지만 블록체인과 World State로 이루어진 장부를 사용하게 된다면 채널 안 모든 피어의 트랜잭션 기록으로 어떤 데이터가 오고 갔는지 참조할 수 있게 된다.

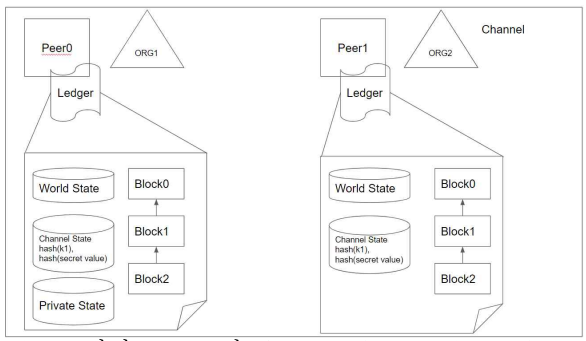


Fig 3. 피어 Ledger의 Channel State, Private State

2.3 Private data collection

하이퍼레저 패브릭 1.2버전부터 private data collection 자료구조를 지원한다[8]. 기존 하이퍼레저 패브릭은 서로 다른 조직간 트랜잭션 내용의 기밀성을 보장하려 할 때마다 새로운 채널을 만들어야 했

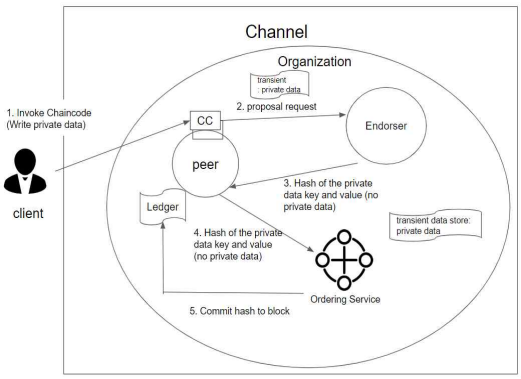


Fig 4. Private data 트랜잭션 흐름

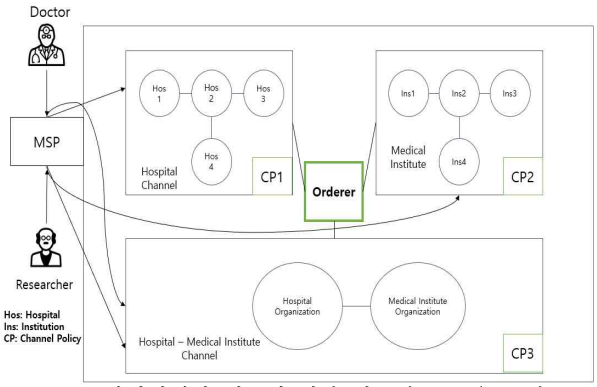


Fig 5. 하이퍼레저 패브릭 기반 의료정보 공유 플랫폼

는데 이는 네트워크에 부담이 되었다. 이에 따라 같은 채널에 있으면서 데이터에 대한 접근 권한을 다르게 설정하여 데이터의 기밀을 보장하는 private data collection이 업데이트 되었다.

Fig 3의 채널에는 조직 1에 속하는 피어 0과 조직 2에 속하는 피어 1이 있다. 피어 0과 1의 장부 모두 World State, 블록체인, 채널 안 Private State의 자료에 접근할 키를 가진 Channel State을 가지고 있으나 Private State의 접근 권한은 피어 0에게 있기 때문에 Private State는 피어 1에 존재하지 않는다. 따라서 Private State을 읽거나 쓰는 작업은 피어 0만 가능하며 만약 피어 0이 속한 조직 1이 Private State에 피어 1이 속한 조직 2의 접근을 허용하면 피어 1의 장부에도 Private State가 복사된다. 조직 2도 마찬가지로 조직 2의 Private State와 접근 권한을 정의할 수 있다.

1.2 이전 버전에서는 피어의 체인코드를 호출할 때마다 순서화 서비스에 트랜잭션이 전달되고 커미터가 이를 모든 피어에게 뿌려주는 방식인 트랜잭션 흐름을 가졌는데 private data collection을 사용하게 되면 순서화 서비스에는 private data의 해시 값이 전달되고 순서화 서비스는 이 해시한 키와 해시한 데이터에 관한 트랜잭션 기록의 순서를 결정하고 채널 안 모든 피어에게 전달하게 된다. 해시 값이 아닌 실제 private data의 키와 데이터는 보증 노드에서 임시적으로 관리하며 자격 증명이 된 피어의 트랜잭션 흐름이 끝나면 임시적으로 저장한 키와 데이터는 피어의 Channel State와 Private State에 저장된다.

III. 하이퍼레저 패브릭 응용 방안

본 논문에서는 위와 같은 하이퍼레저 패브릭의 특징을 갖고 의료정보 공유 플랫폼을 다음과 같이 제안한다. Fig 5와 같이 의료정보 공유의 주체를 병원과 연구소라 할 때 병원 간 정보공유, 연구소 간 정보공유, 병원 연구소간 정보 공유를 위한 세 개의 채널을 만들 수 있다. 우선 이 하이퍼레저 패브릭 네트워크에 참여하려면 조직별로 만들어진 CA를 통해 인증 받아야 하며 이는 MSP를 통해 이루어진다. MSP를 통하여 병원이나 연구소 조직을 할당 받았다면 각 채널 규정에 따라 접근 권한이 있는 채널에 참여가 가능하다. 채널 정책에는 조직 내 데이터를 어떻게 받고 처리할지 체인코드로 정의할 수 있어 효율적인 데이터 처리와 공유를 할 수 있게 된다. 합의 알고리즘은 병원과 연구소에서 원하는 방식대로 만들 수 있어 각 조직별 순서화 서비스가 나올 수 있으나 편의상 한 개로 가정한다.

Fig 6은 병원 - 연구소 채널에서 일어나는 데이터 흐름을 보여준다. 병원-연구소의 거래 흐름은 다음과 같다. 우선 환자의 의료정보 제공 동의를 받은 의사가 하이퍼레저 패브릭의 병원 피어의 체인코드를 호출한다. 병원은 이 기록에 대하여 연구소에서 필요로 하는 의료 기록을 저장한 데이터베이스 Health Record와 환자의 개인 신상을 비롯한 모든 정보가 담긴 데이터베이스 Health Record Details를 만들어 같은 채널 내라도 조직별 접근권한을 다르게 한다. 조직 별 각 private data collection에 들어갈 JSON 형식은 체인코드로 정의할 수 있다. 본 논문은 병원에서 기록한 의료 기록은 이름, 나이, 진단, 처방이고 개인정보 보호를 위해 연구소에 제공할 데이터에는 환자의 이름을 빼야하는 상황을 가정한다.

Fig 7은 Golang으로 작성된 체인코드 내에 하이퍼레저 패브릭 private collection에 저장할 형식인 JSON을 선언하는 부분이다. "healthRecord" 구조체는 환자의 이름이 없으며 연구소만 열람할 수 있게 설정하며 "healthRecordDetails"는 환자의 이름을 포함하며 오직 병원만이 열람할 수 있도록 접근 권한을 정의할 수 있다.

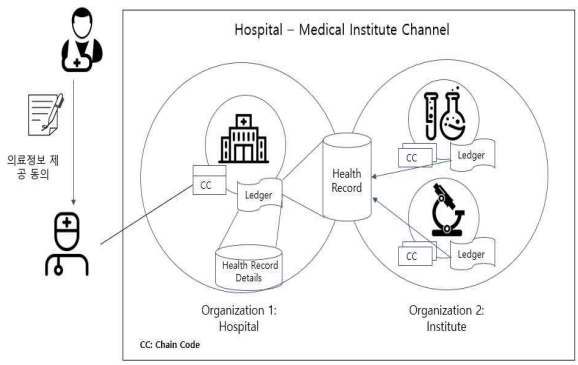


Fig 6. 채널 안 Private State를 이용한 거래 흐름

```
// 체인코드 중 JSON 정의 부분
import ("encoding/json")

type healthRecord struct {
  ObjectType string `json:"docType"`
  Age        int    `json:"age"`
  Diagnosis  string `json:"diagnosis"`
  Prescription string `json:"prescription"`
}

type healthRecordDetails struct {
  ObjectType string `json:"docType"`
  Name       string `json:"name"`
  Age        int    `json:"age"`
  Diagnosis  string `json:"diagnosis"`
  Prescription string `json:"prescription"`
}
```

Fig 7. Private collection JSON (Golang)

```
// collections_config.json
[
 {
  "name": "healthRecord",
  "policy": "OR('HospitalMSP.member', 'Institute
                MSP.member')",
  "requiredPeerCount": 3,
  "maxPeerCount": 10,
  "blockToLive": 1000000
},
 {
  "name": "healthRecordDetails",
  "policy": "OR('HospitalMSP.member')",
  "requiredPeerCount": 4,
  "maxPeerCount": 8,
  "blockToLive": 1000000
 }
]
```

Fig 8. Collection 접근 권한 설정

Fig 8은 collection의 각 이름을 나타내는 "name", 조직의 접근 권한 정책을 나타내는 "policy", collection의 private data를 사용하며 이를 유지하는 최소 피어의 개수 "requiredPeerCount", 최대 피어의 개수 "maxPeerCount"와 데이터가 기록된 시점을 기준으로 몇 번의 데이터 트랜잭션 후 자동으로 데이터가 삭제되어야 하는지 나타내는 "blockToLive"가 설정한 코드를 나타낸다. 병원은 "HospitalMSP" 연구소는 "InstituteMSP"이다. "policy"에서 각 private collection에 대한 접근 권한을 설정한다. 이에 따라 각 private collection을 읽고 쓰는 것에 대한 권한 설정을 할 수 있고 private collection의 기능에 따라 데이터 삭제도 가능하다. 나머지 옵션은 병원과 연구소의 상황에 맞게 설정할 수 있다.

Ⅳ. 결론

본 논문에서는 의료 정보 공유의 필요성과 관련된 법조항을 살펴본 후 하이퍼레저의 private data collection의 특징을 바탕으로 데이터의 접근 통제와 기밀성 그리고 데이터 삭제를 가능한 의료 정보 공유 플랫폼 구현 방안을 설명하였다. 하이퍼레저 패브릭에 기반한 병원, 의료 연구기관들 사이의 데이터 공유 플랫폼은 데이터의 무결성과 기밀성을 보장함과 더불어 네트워크 접근통제까지 이루어져 안전한 데이터 공유 플랫폼이 될 것으로 기대된다.

하이퍼레저 패브릭은 키-값 쌍으로 데이터를 저장하는 JSON 방식의 자료구조를 이용하여 데이터베이스에 저장한다. 따라서 병원들의 의료기록을 공유하기 위해 사용하는 전자의무기록(EHR)을 하이퍼레저 패브릭의 데이터베이스에 저장하기 위한 자료구조 형식에 관한 협의 및 통일에 관한 연구가 필요하다.

[Acknowledgement]

본 연구는 보건복지부의 재원으로 한국보건산업진흥원의 보건의료기술연구개발사업 지원에 의하여 이루어진 것임(과제고유번호 : HI18C0316)

[참고문헌]

[1] CT·MRI 중복 촬영, 3년간 34.5% 증가…월 16억 낭비, <http://news.jtbc.joins.com/article/article.aspx?news_id=NB10579846>, (2018.11.01)
[2] MEDIBLOC WHITEPAPER, <https://medibloc-homepage.oss-us-west-1.aliyuncs.com/whitepaper/medibloc_whitepaper_kr.pdf> (2018.11.01.)
[3] Myung-Kyu-Yi. "A Study on Security Weakness and Threats in Personal Health Record Services", The Journal of The Institute of Internet, Broadcasting and Communication (IIBC) Vol. 15, No. 6, pp.163-171.
[4] Mostert, Menno, et al. "Big Data in medical research and EU data protection law: challenges to the consent or anonymise approach." European Journal of Human Genetics 24.7 (2016): 956.
[5] 생명윤리법, <http://www.law.go.kr/> (2018.10.26.)
[6] 개인정보 비식별 조치 가이드라인, <https://www.privacy.go.kr/> (2018.10.26.)
[7] IPFS, <https://ipfs.io/#uses> (2018.11.01.)
[8] Hyperledger Fabric - Read the Docs, <https://hyperledger-fabric.readthedocs.io/en/release-1.3/> (2018.10.29.)

클라우드 환경에서 CP-ABE 방식을 활용한 안전한 데이터 접근 기법에 관한 연구+

황용운*, 이임영**
순천향대학교 컴퓨터학과

A study on Secure Data Access Scheme using CP-ABE in Cloud Environments

Yong-Woon Hwang†, Im-Yeong Lee**
Dept of Computer Science and Engineering, Soonchunhyang University*,**

요 약

최근 클라우드 컴퓨팅의 발달로 인해 사람들은 클라우드 스토리지에 데이터를 저장하거나 공유할 수 있다. 하지만 클라우드 환경에서도 다양한 보안위협이 존재하는데, 악의적인 사용자로 인해 스토리지에 저장된 데이터가 유출될 수 있다. 이에 클라우드 환경에서 안전하게 데이터 접근하기 위한 다양한 보안 기술이 필요하다. 그 중 속성기반 암호인 CP-ABE 방식을 적용한 기법들이 연구되고 있지만, 다양한 보안위협에 취약한 방식들과 효율성이 부족한 방식들이 존재한다. 본 논문에서는 클라우드 환경에서의 보안위협을 해결하고자 클라우드 환경에서 CP-ABE 방식을 활용하여 인가된 사용자만이 안전하게 클라우드 스토리지에 접근할 수 있으며, 속성 해지 시 기존 방식들에 비해 연산의 효율성을 높이는 접근제어 기술을 제안하고자 한다.

I. 서론

최근 클라우드 컴퓨팅의 발달로 사람들은 클라우드 스토리지에 데이터를 저장하거나 다른 사람들과 공유할 수 있다. 하지만 클라우드 환경에서도 서비스 제공업체를 완전히 신뢰할 수 없으며, 악의적인 사용자로 인해 데이터가 유출되거나 손실될 수 있는 보안위협이 존재한다. 따라서 클라우드 환경에서 데이터 소유자의 데이터를 안전하게 보호해야하며, 클라우드 스토리지에 안전하게 접근하기 위한 보안 기술이 필요하다. 이에 본 논문에서는 클라우드 환경에서 속성기반 암호를 활용하여 인가된 사용자만이 안전하게 클라우드 스토리지에 접근할 수 있으며, 사용자 탈퇴 시 탈퇴 리스트를 사용하여 탈퇴된 사용자들의 접근을 차단한다. 또한 AC(Access control)에서 사용자의 속성과 암호문에 지정된 접근구조와 비교하여 일치하면 1차 복호화를 진행하기 때문에 기존방식들과 비교해 사용자가 암호화된 데이터를 복호화하는데 효율적인 연산을 제공한다.

+ 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음
(IITP-2018-2015-0-00403)

II. 관련연구

2.1 속성기반 암호

속성기반암호는 각 개체가 가지고 있는 속성(소속, 직무) 정보의 집합과 이를 기반으로 생성한 접근구조를 바탕으로 암복호화를 실시하는 방식이다. 그 중 CP-ABE(Ciphertext-Policy Attribute Based Encryption) 방식은 암호문을 생성할 때 송신자가 수신자의 속성을 기반으로 접근구조를 지정하여 암호화된 데이터와 함께 수신자에게 전송하면, 수신자는 자신의 속성 집합을 기반으로 암호문을 복호화 한다[1].

2.2 클라우드 환경에서의 CP-ABE 방식

현재까지 클라우드 환경에서 CP-ABE방식의 기반인 데이터 접근기법에 관한 연구들은 계속해서 제안되었지만, 다양한 보안위협에 취약한 방식들과 효율성이 부족한 방식들이 존재한다. 특히 2012년에 sekhar scheme[2]과 2015년 Zhu scheme[3]에서 탈퇴된 사용자가 다른 사용자의 속성을 가지고 데이터에 접근할 수 있는 위장공격이 발생할 수 있다. 이에 사용자 탈퇴 시 속성철회기법을 적용한 연구가 2012년 Xu scheme[4], 2013년 Yang scheme[5], 2016년 Ramesh scheme[6] 등이 존재한다. 하지만 Xu scheme과 Yang scheme은 사용자 탈퇴시 다른 사용자의 키와 기존에 저장된 암호문을 업데이트하는 비효율성을 가진다. Ramesh scheme에서도 속성 철회 시 속성을 관리하는 AA(Attribute Authority)에서 다른 사용자의 속성 공개키를 지속적으로 업데이트 하는 비효율성을 가진다.

III. 제안방식

본 장에서는 클라우드 환경에서 CP-ABE 기법을 활용한 데이터 접근제어 기법을 제안한다. [그림 1]은 본 제안방식이 적용되는 환경을 시나리오로 나타낸 그림이다.

3.1 시스템 계수

본 제안방식의 시스템 계수는 다음과 같다.

- TTP : 신뢰할 수 있는 3기관(관리자)
- $Storage$: 데이터를 관리하는 서버
- $Access\ Control$: 사용자 접근제어 관리
- PK, MK : 공개 파라미터, 마스터키
- SK : 사용자 비밀키(암호문 복호화키)
- A_U, S : 사용자 속성데이터, 속성데이터 집합
- AS : 접근구조
- ak : 사용자 속성 만족시 생성되는 키값
- $nonce\ value$: 사용자마다 주어지는 랜덤값
- CT, C : 암호데이터, 1차 복호화된 데이터

3.2 시스템 초기화 단계

Step 1. 사용자는 TTP에게 등록메시지를 보내 등록을 요청하면 TTP는 사용자리스트에 사용자에 대한 정보를 등록 후 공개 파라미터와 마스터키 (PK, MK)를 생성한다.

- Setup단계 : 공개파라미터 PK, MK 생성

Step 2. 사용자 속성데이터와 PK, MK를 통해 비밀키 SK를 생성한다. 이후 데이터 소유자에

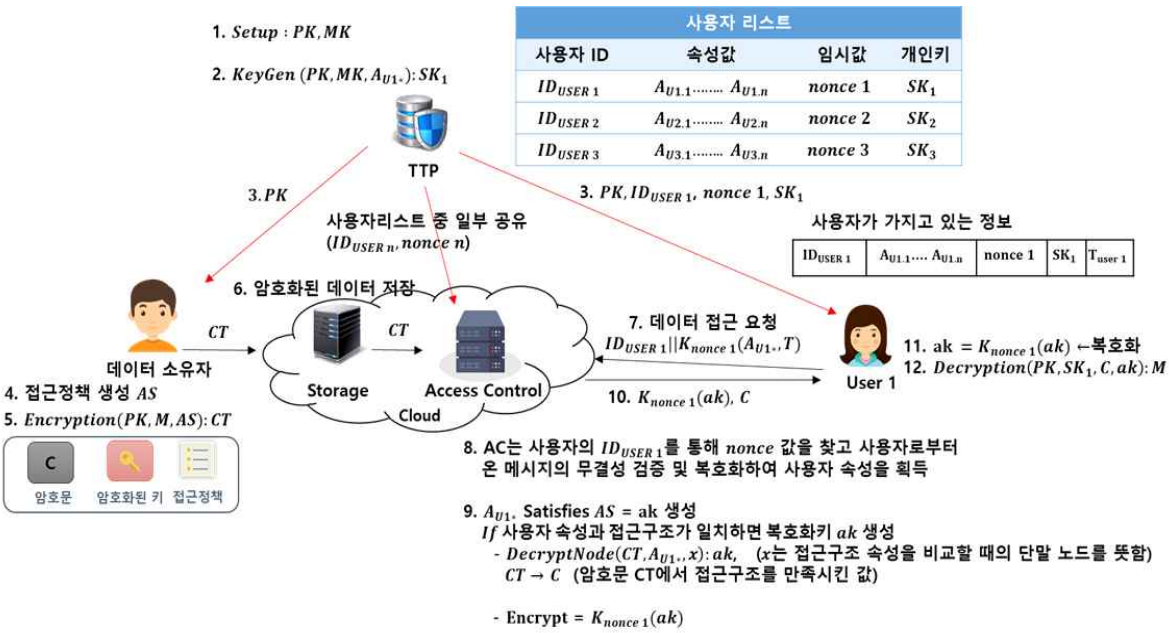


그림 1. 본 제안방식 시나리오

게 PK를 사용자에게 PK와 사용자리스트에 등록된 사용자 $ID_{USER}, Nonce\,value, SK$를 안전한 채널로 전송해준다.

• $KeyGen(PK, MK, A_{U^*})$: 속성의 집합 A_U와 마스터키, PK를 통해 비밀키 SK를 생성

3.3 데이터 암호화 단계

Step 3. 데이터 소유자는 자신이 공유할 데이터 m에 접근할 수 있는 사용자의 속성들을 기반으로 접근구조 AS를 지정한다. 이후 공개파라미터와 접근구조(PK,AS)로 데이터 m을 암호화하여 클라우드 스토리지에 전송하며, 스토리지는 암호화된 데이터를 저장한다.

• $Encrypt(PK,AS,m)$: 공개파라미터 PK와 접근구조 AS로 데이터 암호화 단계

3.4 사용자 데이터 접근단계

Step 4. 암호호된 데이터에 접근을 요청하는 사용자는 TTP로부터 받은 $Nonce\,value$으로 자신의 속성집합을 암호화하여 자신의 식별자와 함께 AC에게 전송한다.

• $K_{nonce}(A_{U^*}T)$: 사용자가 AC에 접근요청단계

Step 5. AC는 사용자로부터 받은 $K_{nonce}(A_{U^*}, T)$를 복호화하고, 사용자의 속성과 암호문에 지정된 접근구조 AS를 비교하여 만족하면 데이터 복호시 필요한 키 ak를 생성한다. 이후 사용자의 $Nonce\,value$으로 ak를 암호화하여 1차 복호화된 암호문 C와 함께 사용자에게 전송해준다.

• $A_{U^*}\,Saticifies\,AS$?

• $DecyptNode(CT, A_{U^*}, x) = ak$: x는 접근구조 트리의 노드로서 사용자 속성과 비교하며 1차 복호화 진행하여 복호화키 ak 생성

• $K_{nonce}(ak), C$: 속성을 만족한 사용자에게 1차 복호화된 암호문과 복호키 전송

Step 6. 사용자는 AC로부터 받은 $K_{nonce}(ak)$를 복호화하여 ak를 획득한다 이후 암호문 C를 비밀키 SK, ak, 공개파라미터 pk를 통해 복호화를 진행하여 메시지 m을 획득한다.

• $Decrypt(PK, SK, C, ak)$: 암호문에 지정된 속성을 가진 사용자는 비밀키 SK, ak로 복호화 단계

3.5 사용자 속성 해지단계

사용자 탈퇴 시 TTP는 속성해지 리스트에서 탈퇴된 사용자의 정보를 등록하고, 탈퇴된 사용자의 $Nonce\,value$를 변경한다. 그리고 사용자리스트와 동기화를 진행한 후 이를 AC와 공유한다. 이후 탈퇴된 사용자가 기존에 가지고 있던 $Nonce\,value$를 기반으로 접근 요청 시 AC는 사용자의 $Nonce\,value$을 모르기 때문에 사용자로부터 받은 $K_{nonce}(A_{U^*}, T)$를 복호화 할 수 없다. 이에 AC는 사용자에게 접근실패 메시지를 전송함으로써 탈퇴된 사용자의 접근을 차단한다.

IV. 제안방식 분석

· 사용자 공모공격, 위장공격 : 본 제안방식은 TTP에서 비밀키 생성시 속성과 랜덤함수를 기반으로 비밀키를 생성하기 때문에 사용자의 속성만으로 비밀키 SK를 유추할 수 없다. 또한 사용자와 AC만이 알고 있는 nonce값을 대칭키로 사용하여 데이터 접근을 요청하고, 데이터 복호시 키 ak도 같이 사용하기 때문에 사용자들간의 공모를 통해 SK를 유추하더라도 데이터 m을 복호화 할 수 없다. 또한 다른 사람의 속성을 사칭에서 접근하는 위장공격에 안전하다.

· 인가되지 않은 사용자의 접근제어 : 본 제안방식에서 사용자가 데이터에 접근하기 위해 초기에 TTP에서 등록절차를 거친다. 또한 스토리지에 저장된 암호화된 데이터에 접근하기 위해 AC에 접근을 요청하게 된다. AC는 사용자의 속성과 암호화된 데이터에 지정된 접근구조 속성을 비교하여 일치하는 사용자에게 암호문을 전송해준다. 이에 본 제안방식에서 인가되지 않은 사용자의 접근을 차단하며, 이에 데이터에 대한 무결성 및 기밀성을 보장 할 수 있다.

본 제안방식은 기존 클라우드 환경에서의 CP-ABE 방식과 비교해 다음과 같은 장점을 가진다. sekhar scheme과 Zhu scheme에서 발생되는 위장공격에 대해 안전하다. 또한 Xu scheme와 Yang scheme, Ramesh scheme에서 사용자 탈퇴 시 다른 사용자의 키와 기존에 저장된 암호문을 업데이트하는 비효율성을 가진다. 이를 본 제안방식에서 사용자리스트와 AC의 추가로 탈퇴된 사용자의 접근을 관리함으로써 사용자 탈퇴 시 3.5의 사용자 속성해지 단계와 같이 진행된다. 이에 기존방식에서 사용자 탈퇴 시 다른 사용자의 키와 서버에 저장된 암호문을 업데이트 하는 비효율적인 연산량을 효율적으로 줄일 수 있다. 그리고 기존방식은 암호화된 데이터를 사용자가 복호화하지만 본 제안방식은 복호화의 연산량의 일부를 AC에서 진행된다. 이에 사용자가 암호화된 데이터를 복호화하는데 필요한 연산량을 기본 방식에 비해 효율적으로 줄일 수 있다.

V. 결론

클라우드 환경에서 발생하는 보안위협을 해결하기 위해 본 논문에서는 CP-ABE 기법을 활용한 데이터 접근제어 기법을 제안하였다. 제안방식은 사용자가 클라우드 스토리지에 접근하기 위해 TTP로부터 부여 받은 *Nonce value*을 대칭키로 활용함으로써 기존 방식에서 다른 사용자의 속성을 통해 발생할 수 있는 위장공격을 해결할 수 있다. 또한 사용자가 탈퇴되거나 추가 시 TTP에서 관리하고 있는 사용자 리스트의 변경으로 사용자가 클라우드 스토리지에 접근할 수 있는 권한을 AC에서 부여함으로 다른 사용자의 키와 암호문 업데이트에 영향을 주지 않는다. 이에 기존 방식에 비해 효율적으로 속성해지를 할 수 있다.

향후 연구로는 기존 방식과의 연산량을 비교하여 본 제안방식의 효율성을 높이는 방향으로 연구를 진행 할 것이다. 추가적으로 암호화된 데이터에 접근하는 사용자에게 익명성을 부여하여 사용자에 프라이버시를 보호할 것이며, 문제 발생 시 추적성을 통해 이를 해결 할 수 있는 연구가 필요한 것으로 사료된다.

[참고문헌]

[1] 박광용, 송유진. "속성기반 암호기술." 정보보호학회지, 제 20호 2권, pp. 85-92, (2010)

[2] Sekhar, B. Raja, et al.: CP-ABE based encryption for secured cloud storage access. International Journal of Scientific & Engineering Research, vol. 3, no. 9, pp. 1-5, (2012)

[3] Zhu, S., Yang, X.: Protecting data in cloud environment with attribute-based encryption. International Journal of Grid and Utility Computing, vol. 6, no. 2, pp. 91-97, (2015)

[4] Xu, Z., Martin, K. M.: Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In Trust, Security and Privacy in Computing and Communications, 2012 IEEE 11th International Conference on, pp. 844-849, (2012)

[5] Yang, K., Jia, X.: Attribute-based Fine-Grained Access Control with Efficient Revocation in Cloud Storage Systems. Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, pp. 523-528, (2013)

[6] Ramesh, D., Priya, R.: Multi-authority scheme based CP-ABE with attribute revocation for cloud data storage. In Microelectronics, Computing and Communications (MicroCom), 2016 International Conference on, IEEE, pp.1-4. (2016)

블록체인 키 관리서버를 이용한 PGP+

노창현, 이임영
순천향대학교 컴퓨터소프트웨어공학과

PGP using blockchain based key management server

Chang-Hyun Roh, Im-Yeong Lee
Dept of Computer Software Engineering, Soonchunhyang University

요 약

전자우편은 인터넷의 발달로 인해 많은 사람들이 통신망을 이용하여 메시지를 주고받는 우편 방식을 말한다. 이러한 전자우편 환경에서는 프라이버시와 기밀성을 보장하기 위해 S/MIME나 PGP같은 암호화 표준들이 사용되고 있다. 하지만, PGP에서는 공개키를 키 서버에 저장하고 다른 기관의 서버와 동기화하여 사용하고 있어서 실시간으로 동기화 되지 않으며, 사용자가 지정하여 사용하는 키 서버가 정지될 경우 공개키의 검색, 등록, 취소를 원활히 진행할 수 없는 문제가 있다.

이러한 문제를 해결하기 위해 본 논문에서는 PGP에서 키 관리 서버를 블록체인 네트워크로 구성하여 등록과 삭제를 실시간으로 모든 서버들이 동기화할 수 있도록 하며, 저장된 공개키의 무결성을 제공하는 시스템을 제안한다.

I. 서론

전자우편(e-mail)은 인터넷의 발달과 함께 발전했으며, 웹 브라우저를 통한 웹 메일(Web Mail)의 등장으로 많은 사용자들이 어려움 없이 사용할 수 있게 되었다. 요즈음에는 SNS(Social Network Service)나 메신저의 발달로 개인의 사용량이 이전보다 많이 줄어들었지만, 기업이나 일부 사용자들은 여전히 전자우편을 활발하게 사용하고 있다[1]. 많은 사람들이 전자우편을 사용함에 따라 전자우편에 대한 기밀성을 보장하는 연구와 표준 제정도 많이 진행되었다. 전자우편의 보안은 두 가지로 나뉘는데 통신상의 안전한 연결망을 확보하는 것과 전자우편의 내용 자체를 암호화하는 방법이다[2]. 안전한 연결망을 확보하는 방법에는 대표적으로 S-HTTP (Secure HTTP), SSL(Secure Socket Layer)의 방법이 있다. 전자우편의 내용 자체를 암호화하는 방식은 S / MIME(Secure / Multipurpose Internet Mail Extensions)과 PGP(Pretty Good Privacy)가 대표적이며 그 중 높은 보안성을 제공하는 것이 PGP이다[3].

PGP는 PKI(Public Key Infrastructure)기반의 알고리즘을 이용하고 수신자의 공개키로 암호화하여 전자우편의 기밀성, 무결성, 인증을 제공한다. 하지만, 공개키를 관리하는데 있어 키 관리 서버들을 이용하기 때문에 동기화의 문제와 단일 지점 실패 오류의 문제도 존재한다. 이러한 PGP 키 관리 서버의 문제점인 동기화의 문제와 단일 지점 실패 오류의 문제를 블록체인 기술로 해결할 수 있다.

블록체인은 분산원장기술 중 P2P(Peer-to-Peer)의 방식의 암호 화폐에서 처음 등장하였으며, 모든 노드가 같은 데이터의 모임인 블록을

+ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00156, 블록체인의 정보보호 프레임워크 및 평가 방법 개발)

이전블록과 다음 블록에 연결하여 무결성을 제공하는 기술이다[4].

이러한 PGP 키 관리 서버의 문제를 해결하기 위해 본 논문에서는 키 관리 서버들을 블록체인 네트워크로 구성하는 방식을 제안한다.

II. 관련연구

2.1 PGP

PGP는 Phil Zimmermann이 1991년에 발표하였다. PGP는 전자우편과 파일을 암호화하여 기밀성과 인증 서비스를 제공한다. 1991년 처음 발표 후 계속하여 발전하여 여러 버전으로 수정되어 왔으며, 2001년에는 IETF(Internet Engineering Task Force)에서 표준으로 지정하여 전자우편 암호화의 표준으로 사용되고 있다[5]. PGP의 동작 방식은 전송하고자 하는 메시지에 암호 알고리즘을 적용하여 암호화한다.

PGP에서의 암호화는 대칭키 암호화와 공개키 암호화 방식이 모두 사용된다. 메시지를 암호화할 때에는 랜덤하게 생성한 세션키를 이용하여 대칭키 암호화 방식으로 진행하며, 공개키 암호화 방식은 세션키를 수신자의 공개키로 암호화하여 메시지 암호화 후 세션키를 암호화하여 수신자에게 전송한다. 수신 받은 수신자는 자신의 개인키로 세션키를 복호화한 후 다시 세션키로 메시지를 복호화하여 진행한다.

위의 동작 방식에서 세션키는 사용자 본인이 직접 생성하여 사용할 수 있지만, 수신자의 공개키는 직접 알아내어야 한다. 공개키를 알아내는 방법은 수신자에게 직접 전송받거나, 공개키 서버에 등록하여 접근하는 방식이 있다. 대부분의 사용자들은 온라인상의 공개키 서버에 접근하여 수신자의 ID로 검색 후 공개키를 습득 하여 PGP 전송 방식을 진행한다.

2.2 Blockchain

블록체인은 2009년 Satoshi Nakamoto가 비트코인이라는 암호 화폐에서 처음 제안하였다. 비트코인은 권력이 집중화된 은행과 같은 기관 없이 사용자들끼리 화폐를 발행하고 거래를 진행하는 구조를 가지고 있다. P2P 네트워크에서 모든 참여자들이 같은 거래 데이터를 소유하면서 네트워크를 유지한다. 블록체인의 특징은 모든 네트워크 참여자들이 같은 데이터를 블록의 형식으로 가지고 있으며, 이 블록들을 블록의 해시 값으로 연결하여 데이터의 위·변조를 불가능하게 한다. 블록체인은 환경에 따라 Public, Private, Consortium의 분류로 나눌 수 있으며, Public 블록체인은 누구나 네트워크에 참여하여 블록의 생성을 할 수 있다. Private나 Consortium은 참여자를 제안하여 네트워크를 구성하며, 지정된 참여자나 기관만이 블록을 생성하고 검증할 수 있다.

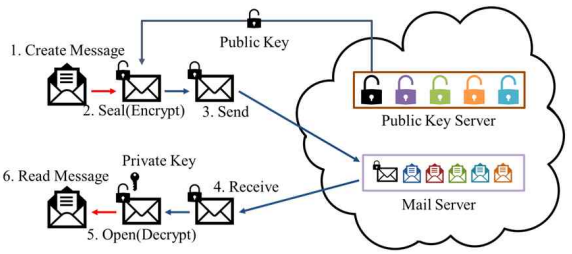
III. 기존방식

기존 PGP의 키 관리 방식은 공개키 서버에 키를 저장하고 다른 공개키 서버와 주기적인 동기화를 통해 키를 분산 저장한다.

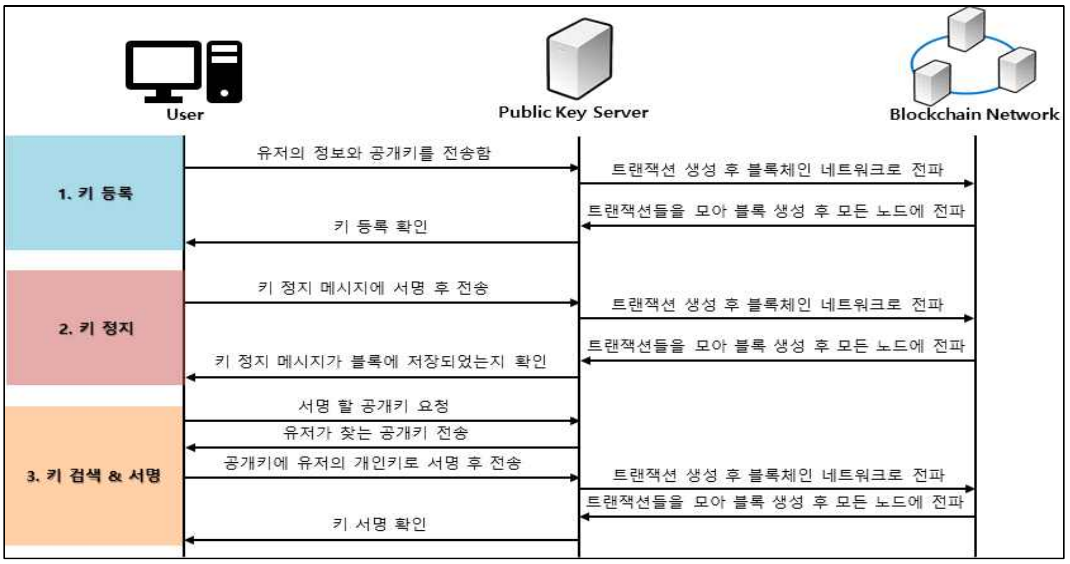
기존 방식의 공개키 서버는 중앙 집중형 구조를 가지고 있어 키가 한 곳에 저장된다는 단점이 있다. 또한, 다른 기관의 공개키 서버와 동기화를 진행하지만 실시간으로 이루어지는 것이 아닌, 60 ~ 90분 정도가 소요된다.

이러한 문제점을 해결하기 위해 비트코인 블록체인 플랫폼에 PGP를 적용하려는 연구가 제시되었다[6]. 하지만, 암호화폐 기반의 블록체인에 PGP의 데이터를 추가하기 때문에 블록의 크기가 커지는 문제가 있으며, 비트코인은 튜링 완전 언어의 스크립트를 제공하지 않아 확장성의 문제도 가지고 있다.

본 논문은 블록체인 네트워크로 공개키 관리 서버를 구성하여 키의 무결성을 제공하며, 어느 한 지점이 동작을 하지 않더라도 다른 노드들로인해 공개키 서버의 서비스가 계속해서 제공하여 가용성을 보장하는 방식을 제안한다.



(그림 1) PGP에서의 통신 방식



(그림 2) 본 제안방식의 전체 시나리오

IV. 제안방식

본 제안방식에서는 기존방식에서 설명한 키 등록, 키 정지, 키 서명, 키 검색 기능을 블록체인 네트워크에서 진행할 수 있도록 한다. 전체 시나리오는 (그림 2)와 같다.

4.1 시스템 계수

- * : 참여객체(U :사용자, PKS : Public Key Server)
- PK_U : 사용자 U의 공개키
- SK_U : 사용자 U의 개인키
- $Info_U$: 사용자 U의 공개키에 들어간 정보
- $Block_i$: i 번째 블록
- Sig_{SK_U} : 사용자 개인키로 서명
- TS : 트랜잭션
- $R.Cert$: 공개키 정지 인증서

4.2 공개키 등록

공개키 등록은 유저가 공개·개인키 쌍을 생성한 후 공개키 서버에 유저정보와 공개키를 서버로 보내고 공개키 서버는 트랜잭션을 생성하여 전파한다. 블록체인 네트워크는 합의 알고리즘을 통해 블록을 생성하고 검증과 블록 전파를 통해 공개키 등록을 완료한다.

4.3 공개키 정지

공개키 정지는 사용자가 공개키 서버에 키 정지 인증서를 보내 저장 되어있는 공개키를 사용할 수 없도록 한다. 사용자는 정지 인증서에 자신의 개인키로 서명을 진행하고 공개키 서버로 전송하고, 공개키 서버는 정지 인증서로 트랜잭션을 만들어 블록체인 네트워크에 전파한다. 그 후 블록체인 네트워크는 블록을 생성 후 블록에 연결하여 키가 정지되었다는 것을 모든 노드에 저장한다.

1) $R.Cert_U$ 생성 후 개인키로 서명 후 $Sig_{SK_U}(R.Cert_U)$ 생성
2) $R.Cert_U$, $Sig_{SK_U}(R.Cert_U)$를 PKS로 전송
3) PKS는 $Sig_{SK_U}(R.Cert_U)$를 PK_U로 검증
4) 검증 후 $R.Cert_U$를 삽입하여 TS 생성
5) 블록체인 노드는 TS를 1분 간격으로 수집
6) 수집된 TS들을 모아 $Block_i$을 생성
7) $Block_i$을 검증 후 블록체인 네트워크에 전파
8) PKS는 유저에게 블록의 헤더를 전송

4.4 공개키 서명

공개키 서명은 유저의 공개키 고리에 있는 다른 사용자의 공개키를 신뢰할 때 진행한다. 유저는 공개키 고리에 있는 신뢰할 다른 사용

자의 공개키에 서명을 진행하고 공개키 서버에 전송한다. 공개키 서버는 다른 사용자의 공개키와 서명을 수신하고 이전에 서명을 진행하였는지 이전 블록들을 통해 확인한 후, 트랜잭션을 생성하고 전파하여 블록을 생성하고 블록체인에 연결하여 서명했음을 알린다.

1) U는 서명할 PK_i로 $Sig_{SK_U}(PK_i)$ 생성
2) PK_i, $Sig_{SK_U}(PK_i)$를 PKS로 전송
3) PKS는 $Sig_{SK_U}(PK_i)$를 PK_i로 검증
4) PKS는 $Block$에서 $Sig_{SK_U}(PK_i)$가 중복되는지 검사
5) 중복이 되지 않으면 TS에 PK_i, $Sig_{SK_U}(PK_i)$를 삽입하여 생성 후 블록체인 네트워크로 전파
6) 블록체인 노드는 TS를 1분 간격으로 수집
7) TS수집 후 $Block_i$을 생성
8) $Block_i$을 검증 후 블록체인 네트워크에 전파
9) PKS는 U에게 생성된 블록의 헤더를 전송

4.5 공개키 검색

공개키 검색은 유저가 자신의 공개키 고리에 다른 사용자를 추가할 때 사용한다. 유저는 다른 사용자의 정보(유저 ID, 키 ID, 전자우편 주소 등)의 데이터로 공개키 서버에 검색 요청을 한다. 검색요청 후 공개키 서버는 동기화된 블록을 맨 뒤에서부터 검색하여 공개키와 다른 사용자의 서명들을 모두 수집하여 유저에게 전송하고 유저는 공개키 고리에 추가한다.

1) U는 다른 사용자의 정보 $Info_i$ 를 습득함
2) U는 다른 사용자의 정보 $Info_i$로 PKS에 검색 요청
3) PKS는 $Block$에서 $Info_i$를 검색
4) 성공한 경우 PK_i와 $Sig_{SK_U}(PK_i)$들을 U에게 회신
5) U는 자신의 공개키 고리에 데이터를 추가

V. 결론

본 논문에서는 기존 PGP의 공개키 서버에서 블록체인 네트워크로 구성된 공개키 서버를 제안하였다. 기존에는 기관 별로 운영하던 서버가 정지될 경우 서비스를 제공할 수 없었던 단일 지점 실패 오류의 문제가 있었다. 또한, 각 기관에서 운영하던 공개키 서버는 실시간으로 동기화를 제공하지 않았기에 키를 안전하게 보관할 수 없었다.

이러한 기존 공개키 서버의 단점을 블록체인 네트워크로 구성하여 단일 지점 실패 오류의 문제를 해결하였고, 실시간으로 모든 서버들이 동기화되기 때문에 무결성을 제공하는 탈 중앙화 된 시스템을 제안하였다. 향후 본 제안방식에서 기존 블록체인의 문제점인 동기화로 인한 트래픽 관리의 어려움과 네트워크 유지를 위한 막대한 연산량의 문제가 있지만 안전한 키 관리의 장점이 더 큰 것으로 생각한다. 또한, 유저들의 공개키 고리의 관리를 스마트 컨트랙트로 자동으로 다른 사용자의 공개키를 신뢰할 수 있는 방식의 연구가 필요하다고 생각한다.

[참고문헌]

[1] Radicati Group, "Eamil Statistics Report, 2017 - 2021", Radicati Group, Feb 6, 2017

[2] 박동욱, 박재희, 김진상, 김일민 "PGP 방식을 이용한 웹 기반 전자우편 보안 시스템", 한국정보처리학회 논문지, 제8-C권, 제1호, 2001.

[3] 이임영, 최용락, 전정훈, 김윤정, 조태남, 김승주, 이재광, 김호원, 신상욱, 소우영, 김성렬, 이주영 공역 "컴퓨터 보안과 암호 5th Edition", 도서출판 그린, 2011.

[4] S. Nakamoto. "Bitcoin : A Peer-to-peer Electronic Cash System", bitcoin.org, 2009.

[5] IETF, "MIME Security with OpenPGP", RFC3156, Aug, 2001.

[6] Wilson, Duane, and Giuseppe Ateniese. "From pretty good to great: Enhancing PGP using bitcoin and the blockchain." International conference on network and system security. Springer, Cham, 2015.

컨소시엄 블록체인환경에서의 Privacy를 보장하는 안전한 통신시스템에 관한 연구+

라경진*, 이임영**
순천향대학교 컴퓨터학과

A Study on Secure Communication System to Ensure Privacy in Consortium Blockchain Environment

Gyeong-Jin Ra*, Im-Yeong Lee**
Dept of Computer Science and Engineering, Soonchunhyang University*,**

요 약

블록체인은 DLT(Decentralized Ledger Technology) 기술 중의 하나로, 참여노드가 모두 동일한 원장을 기록 및 공유하는 기술이다. 따라서 발생한 트랜잭션을 모두 볼 수 있기 때문에 Privacy 문제가 발생한다. 이를 위해 Monero, Z-Cash 등은 암호학적 기술을 적용하여 Privacy를 보장하는 블록체인 기반의 코인을 제안하였다. 하지만 컨소시엄 블록체인은 허가된 멤버로 구성된 여러 그룹이 하나의 원장을 공유해야 하므로 멤버확인과 블록체인 생성이 분리된 효율적인 시스템이 필요하다. 본 논문에서는 컨소시엄 블록체인 환경에 적합하도록 ECC기반의 OTK(One Time Key)와 Root Signature를 통해 사용자를 인증하고, Full Node의 익명신용장을 적용하여 Anonymous ID를 발급, 신원 추적을 어렵게 하여 Privacy가 향상된 통신시스템을 제안한다.

I. 서론

최근 비트코인의 등장과 함께 블록체인은 분산원장으로써 기록을 추가만 가능하고 이를 모든 참여원이 공유한다[1]. 따라서 탈중앙적이며 위조와 변조로부터 강력한 무결성 및 신뢰성을 제공함에 따라 각광받고 있다. 블록체인은 참여 형태에 따라 나뉘며, 누구나 참여 가능한 퍼블릭 블록체인과 허가된 멤버만이 참여 가능한 프라이빗 블록체인, 컨소시엄 블록체인으로 구성된다. 그 중 컨소시엄 블록체인은 허가된 멤버와 이로 구성된 그룹끼리 원장을 블록체인 형태로 공유한다. 따라서 멤버의 인증을 수행하면서 다른 그룹멤버간의 Privacy는 보장해야한다. 따라서 본 논문에서는 컨소시엄 블록체인 환경에 적합하도록 ECC기반의 OTK(One Time Key)와 Root Signature를 통해 사용자를 인증하고, Full Node의 익명신용장을 적용하여 Anonymous라는 익명 ID를 발급, 신원 추적을 어렵게 하여 Privacy가 향상된 통신시스템을 제안한다.

+ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00156, 블록체인의 정보보호 프레임워크 및 평가 방법 개발)

II. 관련 연구

2.1 블록체인

블록체인은 DLT 기술 중의 하나로, 참여 노드가 모두 동일한 원장을 기록 및 소지하는 자료구조 기술이다. 하지만 기존 데이터베이스와 다르게 원장은 거래의 추가만이 가능하고 수정과 삭제가 불가능하다. 이는2가지 특성 때문이다. 첫째, 블록체인의 블록 구성 시 해시 계산으로 비가역성을 가진다. 둘째, 이전블록의 내용이 다음 블록의 내용으로 체인을 형성한 후 모든 노드가 동일하게 갖고 있다. 따라서 현실

적으로 위조와 변조가 어려워 매우 강력한 투명성을 가지고 있다. 또한 블록체인 참여 노드 모두가 기록을 동일하게 저장하므로 중앙의 기관 없이 네트워크 형성 및 유지가 가능하여 탈중앙화가 가능하다[2]. 따라서 중앙집중형 구조에서의 단일지점장애를 피하고 중개인의 수수료 및 절차가 최소화 된다. 블록체인은 참여 형태에 따라 나뉘며, 누구나 참여 가능한 퍼블릭 블록체인과 허가된 멤버만이 참여 가능한 프라이빗 블록체인, 컨소시엄 블록체인으로 구성된다. 그 중 컨소시엄 블록체인은 허가된 멤버와 이로 구성된 그룹끼리 원장을 블록체인 형태로 공유한다.

2.2 ECC(Elliptic curve cryptography)

ECC는 타원곡선 이론에 기반한 공개키 암호 방식이다. 타원곡선은 타원곡선의 한 점으로 가져와 y^2=x^3+ax+b형태의 방정식으로 정의되는 곡선으로, 공개키 체계를 수행하기 위해 특별한 실수 범주에 속하게 한다. 초기 공개 키 암호 방식은 아주 큰 정수를 2개 이상의 소수로 나누는 것이 오래 걸리는 것에 기반을 두고 있다[3]. 타원곡선 암호 또한 알려진 특정한 점에 대한 무작위 타원 곡선의 이산 로그를 찾는 것이 오래 걸린다는 점에서 착안하였다. 이산로그는 이산대수로도 불리며 a^x=b를 만족하는 x를 가리킨다. 여기서 a와 x를 안다면 b를 구하는 것은 매우 쉽지만, a,b만 알 경우, x를 구하는 문제는 매우 어려운 이산대수 문제의 성질을 이용하여 공개키-개인키를 생성 및 이용한다. 이산 대수 문제(Discrete Logarithm Problem)는 모듈러 연산으로 나머지만을 취하는 잉여계시스템에서의 위수를 구하는 것이 어렵다는 문제다. 타원곡선 암호가 RSA나 Elgamal 암호와 같은 기존 공개 키 암호 방식에 비하여 갖는 가장 대표적인 장점은 보다 짧은 키를 사용하면서도 그와 비슷한 수준의 안전성을 제공하는 것이다[2].

2.3 Root Signature

Root Signature는 2017년 Yuan et.at 등이 소개한 scheme으로, Blockchain에서의 다중 Transaction을 하나의 단일 Transaction처럼 Signature를 형성한다[4]. 또한 Input Transaction을 모아 단일 Output Transaction을 생성하는 과정에서 익명성을 제공한다. 이는 ECC의 압축 성질을 이용하여 단일메세지를 생성할 수 있고, 동형 및 겹선형 사상을 이용하여 익명성을 가지기 때문이다[4]. 따라서 본 제안방식에서는 이를 착안하여 일회용 공개키와 Blockchain Server의 글로벌타임스탬프 Transaction의 검증용 단일 메시지를 생성하도록 Root Signature를 제안한다.

III. 보안요구사항

본 연구에서 제안하는 컨소시엄 블록체인에서의 Privacy를 보장하는 안전한 통신시스템은 다음과 같은 보안 요소가 요구된다[5].

- 사용자 인증(Authentication): 정당한 사용자는 사용자의 공개키-개인키를 통해 네트워크의 올바른 사용자임을 보장하여야 한다.
- 신뢰성(Reliability) : 네트워크 참여자는 네트워크의 무결성과 가용성을 보장받아 전체 네트워크를 신뢰할 수 있어야한다.
- 효율성(Efficiency) : 다중블룸필터 기반 인증 기술은 안전하면서 전체 처리량의 오버헤드를 최소화하여 효율적으로 계산 및 가용이 되어야 한다.
- 프라이버시(Privacy) : 컨소시엄 블록체인 내 다른 그룹원 멤버의 트랜잭션 검증 및 인증은 요청자의 정보를 노출시키지 않으며 수행되어야 한다.

IV. 제안방식

본 논문의 제안방식은 인증서기반의 사용자 인증을 위해 사용자의 인증서 발급요청과정, 발급, 인증서 검증 과정으로 구성된다. 객체는 사용자, 컨소시엄 블록체인 네트워크이며 블록체인의 스마트컨트랙트에 따라 구현된다.

4.1 Key generation & Request public key certificate phase

이 단계는 사용자가 인증을 위해 일회용 키를 생성하고 이를 블록체인 서버에 일회용 공개키 검증용 키와 함께 인증서 요청을 하는 단계이다.

Step 1. Client A는 공개키 ID_{PA}와 개인키 ID_{SA}, 인증용 공개키 PK_{Ai}와 개인키 SK_{Ai}를 생성한다.

Step 2. 이후 공개키 검증용 σ를 생성한다.

$P_i = H(r * ID_{PB}) * G + ID_{PA}, \sigma_A = \sum_1^n P^{i\,SK_{Ai}}$

Step 2. Client A는 인증서정보 $(ID_{PA}, \sigma, t_0, PK_{Ai})$을 Blockchain Server에 보내, 인증서발급을 요청한다.

4.2 Public key certificate issuing phase

이 단계는 사용자의 일회용 공개키 검증용 키를 검증한 후 사용자의 ID와 블록체인 서버 ID를 포함하여 인증서를 발급하는 단계이다. 블록체인 서버는 각각의 사용자를 단일노드로하여 인증서발급 트랜잭션을 하나의 Root signautre로 집합하여 블록체인 서버의 Singature를 생성한다.

Step 1. Blockchain Server는 Client A가 보낸 인증서 정보의 일회용 공개키 PK_{Ai}에 따른 R을 검증한다.

$P_i' = H(R * ID_{SB}) * G + ID_{PA}$, 로 부터 $(\sigma, g_2) = \prod_{i=i}^{K} e(P_i, PK_{Ai})$ 일 경우 검증이 성공한다.

$$ID_{SB} * R = . ID_{SB} * r * G = . ID_{SB} * r$$

$$E(\sigma, g_2) = e\left(P^i, g_2^{SK_{Ai}}\right) = (P_i', PK_{Ai}) = OK$$

Step 2. Blockchain Server는 client A의 검증이 이루어졌을 경우, 자신의 SK_{Bi}:을 통해 서명한다.

$$H(r * ID_{PA}) * G + ID_{PB}$$

Step 3. Blockchain Server는 다수의 Client의 인증서 서명정보를 하나의 서명 σ으로 생성한다.

$$P_i = H(r * ID_{PB}) * G + ID_{PA},\ \sigma_B = \sum_1^n P^{i\,SK_{Bi}}$$

Step 4. Blockchain Server로 구성된 Blockchain Network는 Blockchain Server의 단일 서명을 검증 한다.

$$ID_{SA} * R = . ID_{SA} * r * G = . ID_{SA} * r$$

$$E(\sigma, g_2) = e\left(P^i, g_2^{SK_{Bi}}\right) = (P_i', PK_{Bi}) = OK$$

Step 5. 검증이 이루어졌을 경우, Blockchain Network는 Blockchain Server의 단일 서명을 모아 Hash Tree의 Leafnode로 하여 인증서 관리를 수행한다.

Step 6. Blockchain Server는 ID와 검증 σ_B가 포함된 인증서 $(ID_{PA}, \sigma_A, t_0, PK_{Ai}, ID_{PB}, \sigma_B)$를 Client A에 반환한다.

Step 7. Client A는 Blockchain Server는 ID와 검증 σ_B 가 포함된 인증서를 검증한다.

$$ID_{SA} * R = . ID_{SA} * r * G = . ID_{SA} * r$$

$$E(\sigma, g_2) = e\left(P^i, g_2^{SK_{Bi}}\right) = (P_i', PK_{Bi}) = OK$$

4.3 Signing a documents phase

이 단계는 사용자가 인증을 위한 일회용 키와 공개키 인증서와 서명할 메시지를 포함하여 블록체인 서버에 서명을 요청하면, 블록체인 서버는 이를 검증 및 Global Timestamp를 생성하여 메시지에 대한 무결성을 보장하는 유일한 값을 반환하는 단계이다.

Step 1. Client A는 Blockchain Server에 자신의 인증서 정보 와 함께 일회용 개인키 $(ID_{PA}, \sigma_A, t_0, PK_{Ai}, ID_{PB}, \sigma_B), SK_{Ai}$를 전송하여 인증을 수행하고, 동시에 개인키, 메시지 pair $x = H(m, SK_{Ai})$를 보내어 서명을 요청한다.

Step 4. Blockchian Server는 일회용 개인키 SK_{Ai}와 공개된 공개키 PK_{Ai} 로 $(PK_{Ai} = g_2^{SK_{Ai}})$, Client A를 확인한 후 Blockchain Network의 Hash Tree를 통해 공개키 인증서의 유효성을 검증한다.

Step 5. Blockchain Server는 pair $x = H(m, SK_{Ai})$와 Anonymous ID를 추가하여 $x = h(m, SK_{i,} A_{IDA})$를 생성, 이를 Global Hash Tree를 생성한 후 Root Value에 세계협정시간을 Lingking 하여 Global Times

stamp Value S_t을 생성한다.

Step 6. Blockchain Server는 Client A의 공개키를 통하여 Anonymous ID를 암호화 $E=(PK_A, A_{IDA})$하여 전송한다.

Step 7. Blockchain Network는 내부 합의 Consensus를 통해 $\mathrm{Token}(S_t, SK_{Bi})$를 Commit 한 이후 Blockchain Server 및 Client A에 Broad Cast 한다.

4.4 Signature verification phase

이 단계는 Global Timestamp를 포함하는 서명 메시지를 검증자(수신자)에게 전송한 후 검증자는 이를 블록체인 서버에 유효성을 검증하는 단계이다.

Step 1. Client A는 자신 Anonymous ID와 개인키 SK_{Ai}, 메시지의 유일한 Token 값 (S_t, SK_{Bi})과 Verifier의 공개키로 암호화 한 $E(ID_{PV}(M))$을 전송한다. 이때 A_{IDA}를 암호화하여 같이 전송한다.

Step 2. Verifier는 ID_{SV} 로 이를 복호화하여 메시지를 획득한 후, Blockchain Network에 Client A의 인증서와 Token 값 $x=h(m, SK_{i,} A_{IDA})$ 검증 요청한다.

Step 3. Blockchain은 인증서의 유효성을 Hash Tree로 검증하고, Token 값을 검증 할 수 있는 유효 Hash Path를 Verifier에 반환한다.

Step 4. Verifier은 Hash Path를 통해 Blockchain의 Token의 유효성을 검증 할 수 있다.

V. 제안방식 분석

• 사용자 인증(Authentication): 정당한 사용자는 해시체인 및 해시트리로 생성한 공개키-개인키 쌍과 글로벌 타임스탬프가 포함 된 공개키 인증서를 통해 올바른 사용자임을 보장한다.

• 상호 인증(Mutual Authentication): 블록체인 서버, 참여 사용자는 메시지 생성 시 마다 서명을 붙여 서로 간 인증을 제공한다.

• 신뢰성(Reliability) : 네트워크 참여자는 컨소시엄 블록체인을 통해 위·변조로부터 안전하고 Fault Tolerance를 가진 신뢰 네트워크를 형성한다.

• 효율성(Efficiency) : 사용자 및 인증 네트워크는 Root Signature를 활용하여 OTK를 효율적이고 안전하게 검증한다.

• 프라이버시(Privacy) : 컨소시엄 블록체인 내 다른 그룹원간의 Anonymous ID를 활용하여, 사용자의 신원추적을 막아 Privacy를 향상시켰다.

VI. 결론

본 논문에서 컨소시엄 블록체인에서의 Privacy를 보장하는 안전한 통신시스템을 제안하였다. 제안방식분석에 따라 보안요구사항을 만족하면서 Anonymous ID를 통해 Privacy를 높이고 ECC기반의 OTK, Root Signature을 통해 효율성을 개선시켰다. 향후 구체화 된 환경에 적용하여 필요한 기반 프로토콜을 구현 및 실제 구현까지 확대 적용이 필요할 것으로 생각된다.

[참고문헌]

[1] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system", 2008.

[2] 이경현, 이임영, 신상욱, "비트코인과 암호화폐 기술", 한티미디어, 2018.

[3] D. Hankerson, A. J. Menezes, S. Vanstone, "Guide to elliptic curve cryptography", Springer Science & Business Media, 2006.

[4] C.Yuan, M. X.Xu, X. M. Si, "Research on a New Signature Scheme on Blockchain",Security and Communication Networks, 2017.

[5] 이임영 외 11명, "컴퓨터 보안과 암호", 그린출판사, 2011

크로스 아키텍처 바이너리 간 함수 유사도 분석 기법

심규원* 김지민** 지청민** 홍만표*

*아주대학교 사이버보안학과 **컴퓨터공학과

A Study on the Analysis of Function Similarity Between Cross-architecture Binary

Gyu-Won Sim* Ji-Min Kim** Cheong-Min Ji** Man-Pyo Hong*

*Cyber Security, **Dept. of Computer Engineering AJOU University

요 약

기존의 바이너리 간의 유사도 분석을 위한 도구들은 크게 바이너리의 의미론(Semantic)적인 특징과 구문론(Syntax) 적인 특징을 이용하는 기법이 있다. 구문론적 특징을 이용하는 매칭 기법은 비교적 높은 정확도를 가지나 컴파일러 버전, 최적화 옵션, 아키텍처의 차이가 생길 경우 식별률이 저하되는 한계를 가진다. 반면, 의미론적인 특징을 이용하는 매칭 기법은 컴파일 환경에 비교적 영향이 적으나 매칭의 신뢰도가 낮고, 식별 가능한 함수의 수가 제한적이다. 본 논문에서는 유사 함수 식별을 위한 기법으로 바이너리 내의 함수가 호출하는 함수의 반환 타입, 인자 타입, 인자의 개수와 같은 의미론적인 특징을 이용하여 유사 함수를 식별하는 기법 및 식별된 함수를 이용하여 추가적인 함수를 식별할 수 있는 매칭 기법을 제안한다. 이를 기존의 정적 바이너리 유사도 분석 도구인 Diaphora에 구현하여 ARM, MIPS, PowerPC 아키텍처 바이너리를 대상으로 성능을 평가한다.

Ⅰ. 서론

임베디드 및 IoT 기술의 발달로 IoT 장비의 실제 가정 내 보급이 증가함에 따라 이에 대한 공격 시도와 보안 사고가 연이어 발생하고 있다. 따라서 IoT 장비와 같은 임베디드 시스템의 안전성을 보장할 수 있는 대책이 필요하지만 IoT 장비와 같은 임베디드 시스템은 제한된 성능 및 자원으로 인해 기존의 PC 환경에서 사용하는 보안 기술을 적용하기에는 어려움이 존재한다. 따라서 임베디드 시스템에서는 취약점 분석을 통한 소프트웨어 자체의 안전성을 보장해야 한다. 그러나 임베디드 소프트웨어는 소스코드가 공개되지 않는 closed-source의 형태로 배포되어 소스코드 기반의 취약점 분석에 제한이 있으며, 또한 Anti-Reverse Engineering 기법으로 함수의 이름 및 변수 정보를 갖는 심볼이 제거된 형태로 배포되기 때문에 바이너리 코드 기반의 취약점 분석에도 어려움이 존재한다.[1-3] 이러한 심볼 정보가 제거된 바이너리의 취약점 분석을 위해서는 기존의 심볼 정보를 식별한 바이너리와 비교하여 심볼 정보를 복원하는 방법으로 취약점 분석을 진행하며 이를 위한 바이너리 간의 유사도 분석 도구로는 대표적으로 Bindiff와 Diaphora가 있다. 그러나 기존의 도구는 유사 함수 식별 기준으로 사용하는 대부분의 heuristic에 바이너리의 기계 종속적인 특성을 이용하기 때문에 바이너리의 컴파일 환경이 동일 함수 식별에 많은 영향을 주며, 특히 아키텍처가 달라졌을 경우 유사 함수 식별률이 현저하게 낮아지는 한계가 존재한다. 임베디드 장비의 특성상 다양한 아키텍처의 프로세서를 사용하는 경우가 많아 아키텍처가 달라진 경우에도 유사 함수를 식별하는 방법이 필요하다.

본 논문에서는 아키텍처가 서로 다른 바이너리에서 유사 함수를 식별하는 데 활용이 가능한 함수 내에서 호출하는 함수들의 반환형, 인자 타입, 인자의 개수 정보와 식별된 함수를 이용하여 추가적인 함수 식별이 가능한 매칭 기법을 제안하고 이를 기존 바이너리 디핑 도구인 Diaphora에 적용한다.

본 논문은 다음과 같이 구성된다. 2장은 기존의 정적 유사도 분석 도구를 살펴보고, 3장에서 크로스 아키텍처 바이너리 간 유사 함수 식별이 가능한 매칭 기법을 제안하고 4장에서 기존 Diaphora와 비교 분석 및 평가에 대해 서술하였다. 5장에서는 본 연구의 결론을 제시하였다.

Ⅱ. 기존 정적 유사도 분석 도구

2.1. Bindiff

Bindiff는 Zynamics 社의 바이너리 유사도 분석 도구로 IDA의 Plugin 형태로 작동한다. 기본적으로 함수의 CFG(Control-Flow-Graph) 기반으로 유사도를 분석하며 basic block, basic block간의 edge 수, 다른 함수를 호출하는 횟수 등을 기준으로 유사 함수를 식별한다. 그러나 Closed-source 소프트웨어로 분석에 한계가 있으므로 본 논문에서는 사용하지 않는다.

2.2. Diaphora

Diaphora는 Open-source 소프트웨어로 바이너리 간 함수 유사도 분석을 위한 도구이다. Diaphora는 IDA Pro에서 스크립트의 형태로 동작하며 최초 실행 시 [그림 1]과 같이 바이너리로부터 유사도 식별에 필요한 데이터를 추출하여 SQLite 데이터베이스에 저장한다. 바이너리로부터 추출하는 데이터는 IDA Pro에서 분석된 CFG 상의 basic block(node)의 개수, basic block 간의 edge 수, 명령어의 수, Hex-rays에 의해 디컴파일된 pseudocode 등이 있다. 이후 추출된 속성 값을 비교하는 방식으로 유사도를 식별하며 매칭이 발생할 경우 ratio를 산출하여 그 신뢰도를 평가한다. 발생한 매칭은 ratio에 따라서 best, partial, reliable 세 경우로 분류되며 총 41개의 heuristic이 존재한다.

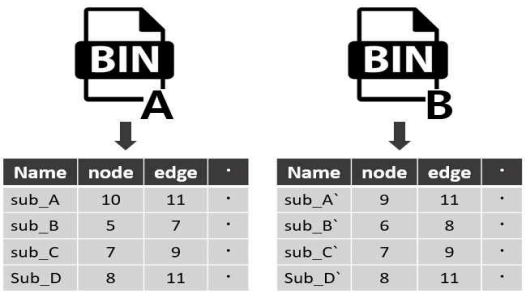


그림 1. Diaphora가 생성하는 SQLite

Ⅲ. 제안 방법

기존의 Diaphora 도구의 바이너리의 의미론적 특징을 이용하는 heuristc들은 바이너리 내에서 basic block의 개수나 특정 분기문의 개수 등 수치적인 특징(numeric features)에 기반을 둔다. 그러나 이러한 수치적인 특징에 기반을 두는 heuristic들은 컴파일 옵션이나 컴파일러 버전, 아키텍처 등의 기계 코드가 생성되는 과정에 차이가 생길 경우 식별률이 낮아지는 문제가 발생한다.[5] 이는 동일한 코드가 컴

그림 2. 함수 프로토타입 추출

파일러의 버전 또는 컴파일 옵션에 의해 기계 코드가 생성될 때 basic block의 수에 차이가 생겨 CFG 표현이 달라지고, 아키텍처가 다를 경우엔 사용하는 명령어 셋이 다르고 함수 호출 문법과 함수 오프셋에도 차이가 생겨 디핑에 사용하는 수치적 특징들이 서로 달라지기 때문이다.[6]

3.1. 호출 함수 프로토타입 비교

본 매칭 기법은 분석 대상 함수와 해당 함수 내에서 호출하는 함수들의 리턴 타입, 인자 타입, 인자 개수를 비교하여 함수 내에서 같은 함수를 호출하는지를 판단하여 유사도를 식별한다. [그림 2]와 같이 분석 대상 함수 내에서 호출하는 함수들의 프로토타입 정보를 디컴파일러를 이용하여 추출하며 디컴파일러는 Hex-rays에 비해 지원하는 아키텍처가 다양한 Avast 社의 Retargetable Decompiler(Retdec)을 이용한다.

IDA Pro에서 심볼 정보가 제거된 바이너리의 함수는 함수의 주소 값으로 만들어진 'sub_XXXX'와 같은 형태의 이름이 붙는다. 함수의 프로토타입에서 이러한 이름은 유사도를 분석할 때 의미가 없으므로 일반화를 수행하여 제거한다. 일반화는 [그림 3]과 같다. 첫 번째는 함수의 리턴 타입, 두 번째는 인자의 타입, 세 번째는 인자의 개수를 의미한다.

위와 같은 방법으로 매칭을 할 경우 특정 함수 A가 호출하는 함수가 오직 한 개라면, A 함수는 다른 여러 함수들과 일치하는 것으로 잘못 도출될 가능성이 크다. 이러한 False positive를 줄이고자 함수 내에서 호출하는 함수의 개수가 한 개일 경우에 한해서 추가적인 식별 방법을 사용한다. 예를 들어 [그림 4]와 같이 func_A 함수 내에서 사용되는 함수가 fun

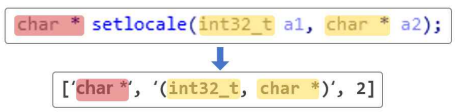


그림 3. 함수 프로토타입 일반화

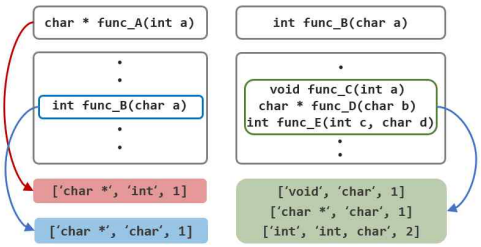


그림 4. 함수 내에서 호출하는 함수가 한 개일 경우

c_B 하나일 경우엔 단순히 일반화된 프로토타입을 매칭하는 것이 아닌 func_B 함수 내에서 호출하는 함수들의 프로토타입 정보를 추가적으로 추출하여 모두 일치할 경우에만 일치하는 것으로 판별한다.

그 외에 함수 내에서 호출하는 함수의 개수가 두 개 이상일 경우엔 [그림 5]와 같이 분석 대상 함수와 해당 함수 내에서 호출하는 함수들의 프로토타입을 일반화하며, 일반화된 프로토타입의 비교를 진행하여 함수의 유사도를 판단한다.

3.2. 식별된 함수를 이용한 매칭

동일 함수일 경우 해당 함수 내에서 호출하는 함수 또한 일치할 가능성이 크다. 따라서 기존에 식별된 함수 내에서 호출하는 함수들 중 매칭이 발생하지 않은 함수들에 대해서 추가적인 비교를 통해 함수 식별이 가능하다. [그림 6]은 실제 동일한 리눅스 Coreutils의 소스코드를 컴파일러 버전 gcc 8.2와 5.4 환경에서 컴파일 된 바이너리를 Diaphora를 이용하여 디핑을 한 결과이며 하단에 위치한 'gettext_quote' 함수와 중간의 'quotearg_buffer_restyled' 함수는 비교 대상 바이너리에 동일 함수가 존재했지만 식별되지 않은 함수이다. 그러나 해당 호출 그래프에서 부모가 되는 'quotearg_buffer' 함수는 디핑에 의해 식별이 된 상태고 해당 함수가 호출하는 함수는 한 개이며 식별되지 않은 함수이다. [그림 6]과 같이 부모 함수가 식별되었고 자식 함수들이 식별되지 않았을 경우 식별되지 않은 자식 함수들은 서로 동일하다고 볼 수 있으며 함수 내에 여러 함수가 식별되지 않았을 경우 3장 3절에서 설명하는 신뢰도 평가 방법에 의해 신뢰도 있는 매칭을 결정한다.

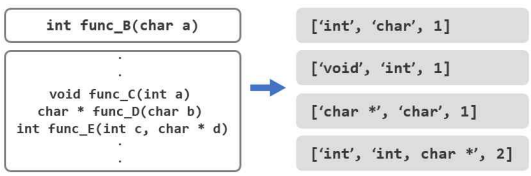


그림 6 함수 내에서 호출하는 함수가 여러 개일 경우

그림 5. Diaphora가 식별한 함수를 이용한 매칭 예

3.3 중복 매칭 선정 기준

제안한 매칭 기법은 특정 함수가 비교 대상 바이너리의 여러 함수와 조건을 동시에 충족해서 중복으로 매칭이 발생할 수 있다. Diaphora에서는 신뢰도 높은 heuristic을 우선 검사하고 그 신뢰도에 따라 ratio를 산출하여 중복 매칭으로부터 신뢰도 있는 매칭을 판별한다. Diaphora에서 ratio는 IDA Pro의 Hex-rays에 의해 디컴파일된 pseudocode와 CFG에 의해 계산되는 값과 어셈블리 명령어를 이용하여 계산한다. Hex-rays는 지원하는 아키텍처가 x86, x64, ARM32, ARM64로 한계가 있으며, 어셈블리와 CFG 또한 바이너리의 아키텍처에 영향을 받기 때문에 본 매칭 기법에서는 사용이 어렵다. 따라서 본 매칭 기법에서는 중복 매칭이 발생했을 경우 매칭된 함수 간의 basic block 수의 차이가 가장 적은 매칭이 실제 유사한 함수로 평가하며 신뢰도를 둔다.

Ⅳ. 제안 기법의 성능 평가

앞서 제안한 매칭 기법의 성능을 평가하기 위한 방법으로 Diaphora에 해당 매칭 조건을 heuristic 형태로 구현하여 추가하였고 선행 연구를 참고하여 특정 실험 바이너리들을 다음과 같이 선정하였다. [6-8]

- GNU Coreutils - dir, du, mv 바이너리
  1) Coreutils arm과 mips 비교
  2) Coreutils arm과 powerpc 비교
  3) Coreutils mips와 powerpc 비교

[그림 7]이 실험 결과를 나타낸다. 기존의 Diapho-ra와 제안한 매칭 기법을 적용한 Diaphora의 실험 결과 false positive를 제외하고 실제로 일치하는 함수의 개수를 기준으로 ARM과 MIPS에서는 약 4배, ARM과 PowerPC에서는 1.5배 정도의 성능 향상과. MIPS와 PowerPC에서는 약 1.4배 정도의 성능 향상을 보인다. ARM과 Powerpc, MIPS와 PowerPC의 바이너리 디핑에서는 ARM과 MIPS의 경우에 비해 다소 적은 함수가 추가로 식별된 것을 확인할 수 있는데, 이는 제안한 매칭 기법 또한 완전히 기계에 독립적

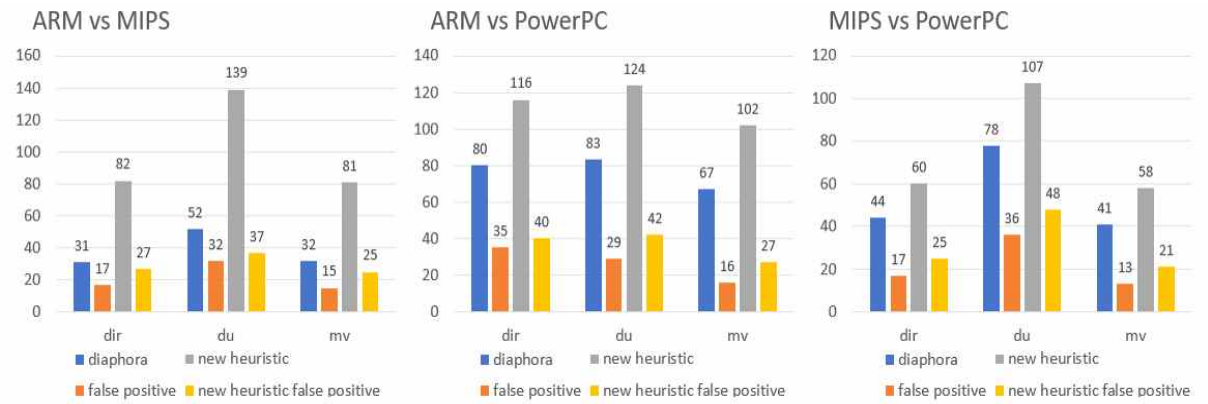


그림 7. 크로스 아키텍처 간 바이너리 디핑 결과

인 특성을 가지지 않으며 아키텍처에 따라 디컴파일된 함수의 프로토타입에 차이가 생기기 때문으로 보인다.

V. 결론 및 추후 연구

본 연구에서는 심볼 정보가 제거된 임베디드 소프트웨어를 기존의 분석이 완료된 바이너리와 비교를 통해 심볼 정보를 복원할 수 있는 바이너리 간의 유사 함수 식별 기법을 제안한다. 함수 내에서 호출하는 함수의 리턴 타입, 인자의 개수 정보를 이용한 매칭 기법과 기존의 식별된 함수를 이용하여 추가 함수를 식별하는 방법을 제안하였으며 제안한 기법은 함수의 의미론적인 특징을 이용하여 유사 함수를 식별하는 방법으로 크로스 아키텍처 간의 바이너리에서도 유사 함수를 식별하는데 활용이 가능하다. 제안한 매칭 기법을 적용하여 실험한 결과 기존의 바이너리 디핑 도구인 Diaphora 대비 모든 크로스 아키텍처 바이너리 간 디핑에서 성능 향상을 보였다. 본 연구를 통해 심볼 정보가 제거된 임베디드 소프트웨어의 취약점 분석에 있어서 기존의 심볼 정보를 식별한 크로스 아키텍처의 바이너리를 디핑에 활용하여 심볼 정보 복원이 가능하다.

ACKNOWLEDGEMENTS

본 연구는 과학기술정보통신부 및 국가보안기술연구소의 위탁연구개발사업의 연구결과로 수행되었음 (2018-059)

[참고문헌]

[1] 정영준, 임동혁, 서영빈, 김재명, 임베디드 운영체제 보안 기술 동향, 전자통신동향분석 제 23권 제 1호, February, 2008

[2] Zhi-Kai Zhang, Zhi-Kai Zhang, IoT Security: Ongoing Challenges and Research Opportunities, 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, November, 2014

[3] Md. Mahmud Hossain, Maziar Fotouhi, Ragib Hasan, Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things, 2015 IEEE World Congress on Services, 2015

[4] Misbah ul Iman, A F M Ishaq, Anti-reversing as a tool to protect intellectual property, 2010 Second International Conference on Engineering System Management and Applications, 2010

[5] Computer Security: 23rd European Symposium on Research in Computer Security, 80p, 23rd European Symposium on Research in Computer Security, September, 2018

[6] Jannik Pewny, Behrad Garmany, Robert Gawlik, Christian Rossow, Thorsten Holz, Cross-Architecture Bug Search in Binary Executables, IEEE Symposium on Security and Privacy, 2015

[7] Fabio Pagani, Matteo Dell'sAmico and Davide Balzarotti, Beyond Precision and Recall: Understanding Uses (and Misuses) of Similarity Hashes in Binary Analysis, 8th ACM Conference on Data and Application Security and Privacy, March, 2018

[8] Mahinthan Chandramohan, Yinxing Xue, Zhengzi Xu, Yang Liu, Chia Yuan Cho, Hee Beng Kuan Tan, BinGo: Cross-Architecture Cross-OS Binary Search, Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016

효율적인 한국 암호 모듈 검증 제도를 위한 암호 모듈 자동 검증 시스템

박태환* 안규황** 권혁동** 서화정** 김호원*†

*부산대학교 대학원 전기전자컴퓨터공학과

**한성대학교 대학원 정보시스템공학과

Automatic Cryptographic Module Verification System for Efficient Korean Cryptographic Module Verification System

Taehwan Park* Kyuhwang An** Hyeokdong Kwon* Hwajeong Seo** Howon Kim*†

*Department of Electrical and Computer Engineering, graduate school of Pusan National University.

**IT Department, graduate school of Hansung University.

요 약

최근 ICMC 2018 미국 국립표준연구소(NIST)에서 암호 모듈 자동 검증 도구 개발 내용에 대한 발표가 있었으며, 이를 위한 ACVP(Automated Cryptographic Validation Protocol)을 제시하고 있다. 이를 통한 암호 모듈 검증의 효율성 강화가 기대되고 있다. 암호 모듈 검증과 관련된 이러한 변화에 맞추어 본 논문은 암호 모듈 검증 효율성 강화를 위한 한국형 암호 모듈 자동 검증 시스템을 제시하며, 국산 해시함수 LSH-224/256/384/512와 블록암호 LEA-128/192/256 ECB, CBC, CTR 모드에 적용하였다.

I. 서론

오늘날, 암호 모듈 검증 제도는 미국 국립표준연구소(NIST)와 캐나다 통신보안기구(CSE)를 중심으로 CMVP(Cryptograph Module Validation Program)이 활발히 이루어지고 있다. 암호 모듈 구현 적합성 검증은 검증대상 암호 알고리즘 유형 및 특성에 따라 시험검증이 이루어지며, 대칭키 암호의 경우, KAT(Known Answer Test), MMT(Multiple Message Test), MCT(MonteCarlo Test)등을 수행하게 된다. 암호 모듈 구현 적합성 검증 시, 시험기관은 벤더(시험신청기관)에서 요청한 검증대상 암호 알고리즘 별 구현 적합성 확인 파일(Facts 파일), 구현 적합성 요청 파일(Request 파일), 구현 적합성 답변 파일(Response 파일)을 바탕으로 구현 적합성 검증을 실시하며, 시험기관과 벤더 간의 상호 요청 및 대응, 검증 기간이 필요하게 되며, 구현 적합성 검증에 있을 경우, 벤더에서 수정 작업 수행하는 시간 또한 필요하게 된다. 이러한 문제를 해결하고, 암호 모듈 검증 제도의 효율성 강화를 위해, 미국 NIST에서는 ACVP 및 관련 사항을 구축하고 있으며, 이에 대한 내용을 ICMC 2018에서 발표하였다.

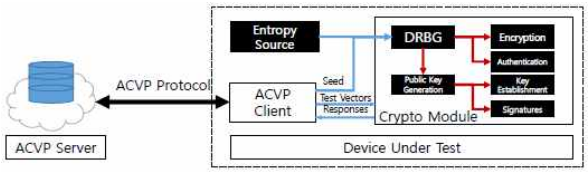
ACVP와 같은 암호 모듈 검증제도의 효율성 강화 연구 동향을 바탕으로 본 논문은 한국 암호 모듈 검증 제도의 효율성 강화를 위한 한국형 암호 모듈 자동 검증 시스템을 제시하고자 한다.

II. 관련 연구 및 동향

2.1. ACVP

ACVP(Automated Cryptographic Validation Protocol) [1]은 미국 NIST에서 주도하여 개발 중인 자동화 암호 검증 프로토콜이다. 최근 급속한 ICT 기술 발전으로 인해, 암호 모듈 검증 신청 건수의 급증으로 기존의 인적 자원 기반의 암호 모듈 검증에 있어 시간 지연 및 어려

움이 발생함에 따라 이를 해결하기 위한 방안으로 연구 개발 중에 있다. 그림 1은 NIST ACVP의 서버 클라이언트 구조를 나타낸다.



Fig, 1. NIST CAVP 서버-클라이언트 구조

그림 1에서 ACVP 서버는 Web 호스트 서비스를 제공하며, 검증 대상 암호 알고리즘 별 ACV 프로토콜에서 정해진 JSON 포맷에 따라 테스트벡터(기존 암호 모듈 구현 적합성 검증 시, FACTS 파일과 REQUEST 파일에 해당)를 생성한다. 그리고 ACV Protocol을 기반으로 ACVP 클라이언트에서 전달 받은 테스트벡터 수행 결과물(기존 암호 모듈 구현 적합성 검증 시, RESPONSE 파일에 해당)에 대한 검증 기능을 수행한다.

2.2. KCMVP

그림 2는 KCMVP(Korea Cryptography Module Validation Program)에 있어 벤더의 검증대상 구현 결과물에 대한 암호 알고리즘 시험 방식을 나타내고 있다. 아래의 그림에서 시험 기관은 KCMVP에서 검증대상 암호 알고리즘 별 구현 적합성 검증을 위한 FACTS 파일(벤더로부터 받은 RESPONSE 파일 검증 확인용), REQUEST 파일(벤더에게 전달하여 벤더의 암호 알고리즘 구현 적합성 확인을 위한 키, 평문, 초기화 벡터 등의 정보를 가지고 있음)들을 가지고 있다. 시험 기관은 KCMVP를 신청한 벤더들 중 암호 구현 적합성 검증이 필요한 벤더에게 REQUEST 파일을 전달하며, 벤더는 이에 대한 RESPONSE 파일을 생성하여 시험 기관에 제출한다. 이후, 시험기관은 벤더로부터 전달 받은 RESPONSE 파일에 대해 검증도구(KCAVS)를 활용하여 FACTS 파일과 비교 검증을 수행하게 된다. 이러한 일련의 과정에 있어 벤더 측에서 시험 기관의 REQUEST 파일에 대한 RESPONSE 파일 생성하는데 소요되는 시간, 시험 기관 측에서의 구현 적합성 검증 결과가 벤더의 구현물에 대한 구현 적합성 문제가 있는 경우, 벤더 측에 전달 및 벤더 쪽에서 구현 결과물 수정에 있어서 시간이 소요되게 된다는 문제점을 가지고 있다. 따라서 KCMVP의 전반적인 효율성 강화를 위해 이러한 문제점 해결이 필요할 것으로 보인다.

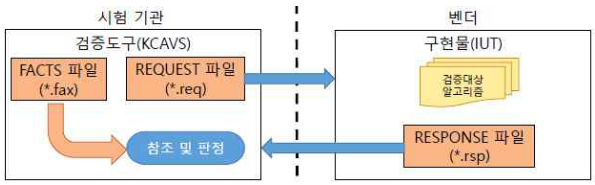


Fig. 2. KCMVP 암호 알고리즘 시험 방식

III. 제안 기법

미국 NIST는 암호 모듈 검증 제도의 효율성 강화를 위한 ACVP[1]를 개발 중에 있으며, 관련 제도상에 적용할 예정이다. 그러나 한국 암호 모듈 검증제도[2]의 경우 벤더 측에서 검증하는데 소요시간이 많이 걸린다는 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해, 본 논문에서는 KCMVP의 효율성 강화를 위한 한국형 암호 모듈 자동 검증 시스템을 제안하며, 국산 해시함수 LSH-224/256/384/512와 블록암호 LEA-128/192/256 ECB, CBC, CTR 모드에 대하여 각각 실제로 적용하였다.

3.1. 제안 시스템 구조 및 동작 원리

그림 3은 제안 시스템의 구조를 나타내고 있다. 시험 기관 쪽에서는 제안 시스템을 웹 서버 형태로 운영한다. 제안하는 시스템에서 지원하는 검증 대상 암호 알고리즘 유형 별 FACTS, REQUEST 파일 생성 기능, 벤더 측에서의 검증 대상 암호 알고리즘 구현 물 업로드 기능, 벤더 구현 물 검증 및 검증 결과 송신 기능을 제공한다. 제안 시스템 상에서의 검증 대상 암호 알고리즘 유형 별 FACTS, REQUEST 파일 생성 기능은 벤더 측에서 구현 적합성 검증을 하고자하는 검증 대상 암호 알고리즘 유형을 선택함으로써, 해당하는 검증 대상 암호 알고리즘 유형에 맞는 FACTS, REQUEST 파일들이 생성된다.

제안 시스템 상에서의 벤더 측에서의 검증

대상 암호 알고리즘 구현 물 업로드 기능은 제안 시스템에서 허용하는 파일 유형(.zip, .c, .h 파일)에 맞춰진 벤더측 검증 대상 암호 구현물을 업로드 하는 기능을 제공한다. 파일 업로드 시, 웹 서버 측에서는 허용하는 포맷의 파일만 업로드가 가능하다.

마지막으로 제안 시스템 상에서의 벤더 측 구현 물에 대한 검증 및 검증 결과 송신 기능은 앞선 구 현물 업로드 기능을 통해 받은 벤더 구현 물에 대해 .zip 파일인 경우, 압축 해제를 실시하며, .c/.h 파일에 대한 컴파일 및 컴파일 결과물을 실행한다. (벤더 구현 물은 기존에 정해진 REQUEST, RESPONSE 파일 양식을 따른다는 가정) 벤더 측 구현 결과물에 대한 실행을 통해 얻은 RESPONSE 파일들과 기존에 생성된 FACTS 파일과의 비교를 통한 구현 적합성 검증을 실시하며, 이에 대한 결과를 웹상에 알려주는 기능(결과 송신)을 수행하게 된다.

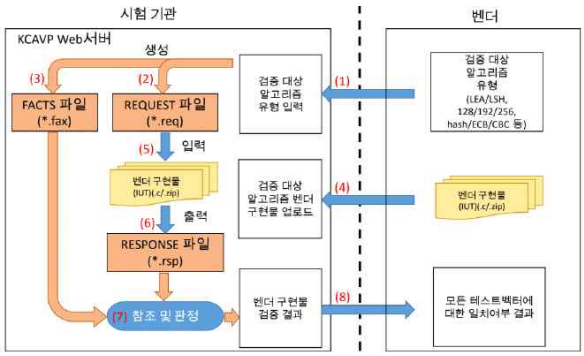


Fig. 3. 제안 시스템 구조

표 1은 본 제안 시스템 상에서 지원하는 검증 대상 암호 알고리즘을 나타내고 있다.

Table. 1. 제안시스템 상에서 지원하는 검증대상 암호 알고리즘

지원 검증대상 암호 알고리즘 유형			지원 검증대상 암호 알고리즘 명
비밀키	블록암호		LEA-128/192/256
	블록암호 운영모드	기밀성	ECB, CBC, CTR
해시함수			LSH-224/256/384/512

LEA-128/192/256 블록암호의 ECB, CBC, CTR 운영 모드를 지원하고 있으며, LSH-224/256/384/512 해시 함수를 지원하고 있다. 관련 테스트 유형으로는 블록암호에 대한 KAT(Known Answer Test) 기능과 해시함수에 대한 짧은 길이의 메시지, 긴 길이의 메시지, 임의의 길이에 대한 메시지 테스트를 지원한다.

3.2. 제안 시스템 동작 결과

그림 4는 제안 시스템의 메인 화면을 나타내고 있다. 제안 시스템의 메인화면에서 사용자가 선택한 검증대상 알고리즘 유형에 따라 화면이 변경된다. 사용자는 구현 적합성 검증을 받고자 하는 암호 알고리즘의 유형, 출력 비트, 운영 모드 등을 선택한 후, Send to Server 버튼을 클릭하게 되면, 서버에서 해당하는 검증대상 암호 알고리즘에 대한 FACTS, REQUEST 파일을 생성하게 된다. 검증대상 암호 알고리즘에 대한 FACTS, REQUEST 파일을 요청한 이후 웹 서버 하단의 파일 업로드를 통해, .zip, .c, .h 파일을 업로드 할 수 있으며, 사용자(벤더)에서 업로드한 구현물에 대한 컴파일 수행 시 컴파일 에러가 발생하는 경우 컴파일 에러 알림 기능과 메인 페이지로 돌아가는 버튼을 제공한다.

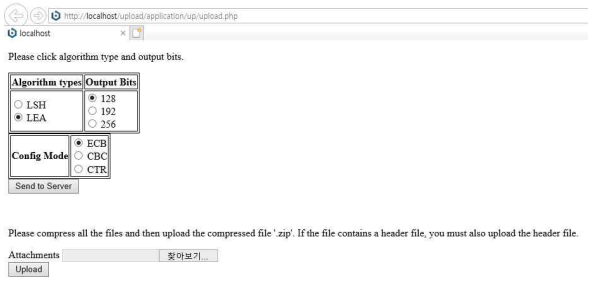


Fig. 4. 제안 시스템 메인화면(LEA 블록암호)

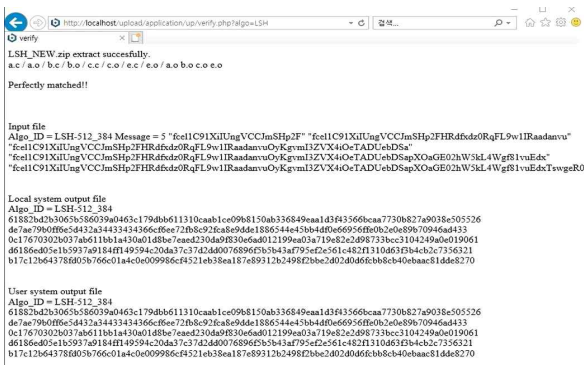


Fig. 5. 제안 시스템 상에서의 LSH 해시함수 수행 결과 비교 검증 화면

그림 5는 사용자(벤더 측)에서 LSH-512/384 해시 함수에 대한 구현 적합성 검증을 요청하여, 구현 결과물을 .zip 파일 형태로 업로드한 후, 서버측에서 구현물 컴파일/수행을 통해 얻은 RESPONSE 파일과 사용자 요청에 따라 미리 생성된 FACTS 파일과의 구현 적합성 비교 검증 결과를 알려주는 화면이다.

IV. 제안 방식 평가 및 비교

사용자(벤더) 및 시험 기관 측면에서의 효율성 강화 측면, 다양한 유형의 암호 모듈 적용 가능성 측면, 한국 암호 모듈 검증 제도 적용 가능성 관점에서 본 논문에서 제안하는 방식과 기존 관련 연구 및 방식간의 비교를 진행하였다.

Table. 2. 기존 관련 연구 및 방식과의 비교

연구 방식	사용자 효율성 강화	시험 기관 효율성 강화	다양한 유형의 암호 모듈 적용 가능성	한국 암호 모듈 검증 제도 적용 가능성
ACVP	O	O	O	X
KCMVP	-	-	O	O
제안 기법	O	O	X	O

미국 NIST의 ACVP는 사용자(벤더), 시험기관 측면에서의 효율성 강화와 다양한 유형의 암호 모듈 적용 가능성이 높다. 하지만 한국 암호 모듈 검증제도(KCMVP)와 검증 대상 암호 알고리즘과 제도적 측면에서 차이가 있기 때문에 적용이 불가능하다는 단점을 가지고 있다.

KCMVP에서의 암호 모듈 구현 적합성 검증 방식은 다양한 유형의 암호 모듈 적용이 가능하며, 한국 암호 모듈 검증 제도(KCMVP)에서의 검증 대상 암호 알고리즘에 대한 구현 적합성 검증이 가능함으로써 적용이 가능하다는 장점을 가지지만, 현재의 암호 모듈 구현 적합성 검증에 있어서 사용자(벤더) 측에서의 구현 적합성 검증 및 검증 결과에 따른 수정 보완 작업에 있어서 많은 시간이 소요된다는 단점을 가지고 있다.

본 논문에서의 제안 기법은 웹 서버 형태의 암호 모듈 구현 적합성 검증 시스템을 통해, 사용자(벤더) 측에서 구현 적합성 검증을 받고자 하는 암호 알고리즘에 대한 구현 적합성 검증 및 검증 결과를 받을 수 있으며, 이를 통해 기존 방식에 비해 사용자(벤더) 측면에서의 수정 및 보완 작업의 시간을 줄일수 있다는 장점(사용자 측면에서의 효율성 강화)과 시험기관 측면에서는 시험 검증 신청 벤더별 구현 적합성 검증 수행 및 벤더 측에서 알리는 시간을 절약할 수 있다는 장범이 있으며, LEA 블록암호와 LSH 해시함수에 대한 구현 적합성 검증 기능을 제공함으로써 KCMVP에 적용이 가능하다. 하지만 PC환경에 대한 암호 소프트웨어 모듈 외의 타 유형의 암호 모듈에 대해 지원하지 않는다는 단점이 있다.

V. 결론

본 논문에서는 미국 NIST의 ACVP와 한국 암호 모듈 검증 제도(KCMVP)에 대한 연구를 살펴보았으며, 시험 기관과 사용자(벤더) 간의 암호 모듈 구현 적합성 검증 시, 시간이 많이 소요된다는 문제점을 해결하기 위해 본 논문에서는 웹 서버 기반의 암호 모듈 구현 적합성 검증 시스템을 제안하였다.

제안하는 기법은 벤더별 구현 적합성 검증 수행 및 벤더 측에서 알리는 과정에서 소요되는 시간을 절약할 수 있다는 장점을 통해, 효율적인 암호 모듈 구현 적합성 검증이 가능해진다. 본 논문에서 제안하는 시스템의 소스는 Github[1]에서 확인할 수 있으며, LSH와 LEA에 실제로 적용하여 동작 수행하는 동영상은 Youtube[2]에서 확인이 가능하다.

[참고문헌]

[1] NIST, "Automated Cryptographic Validation Protocol(ACVP)", https://github.com/usnistgov/ACVP, June 2018.

[2] National Intelligence Agency, "Cryptographic Module Vlidation", http://www.nis.go.kr/AF/1_7_3_1.do

1) Github: https://github.com/kyu-h/Kcryptoforum_CAVP
2) Youtube: https://youtu.be/e-cjDbVDOcw

DQ-Net : 동적 QR 코드 기반 Wi-Fi 인증 메커니즘

양유진, 임정현, 최정은, 서화정*1)

한성대학교 IT융합공학부

DQ-Net : Dynamic QR code-based Wi-Fi authentication mechanism

YuJin Yang, JeongHyun Lim, JeongEun Choi, HwaJeong Seo*

Division of IT Convergence Engineering, Hansung University.

요 약

무선 네트워크 기술인 Wi-Fi(Wireless Fidelity) 사용 시, 대부분의 이용자들이 SSID(Service Set Identifier)의 기본값을 그대로 사용함으로써, 관련된 정보를 알고 있는 해커들에게 공격당할 가능성이 높아진다. 번거로움 때문에 초기의 비밀번호를 그대로 사용하면, 비밀번호를 공유하는 사람이 늘어나면서 시간이 지날수록 보안성이 떨어지는 문제점이 발생한다. 본 논문에서는 이러한 보안 취약점 해결과 동시에 사용 편의성을 향상시키는 방법을 제안한다.

I. 서론

현재 Wi-Fi가 널리 사용되고 있음에도 불구하고 Wi-Fi 보안에 대한 사람들의 관심은 턱없이 부족하다. 그렇기에 사람들은 암호가 설정 되어 있지 않은 Wi-Fi를 아무 의심 없이 사용한다. 이는 나쁜 의도를 가진 사용자에게 해킹당할 위험성을 증가시킨다.[5] 또한 암호가 설정되어 있더라도 주기적으로 변경하지 않기에 보안성은 점차 취약해진다.

스마트폰 이용 확대와 함께 QR 코드(Quick Response Codes) 스캐너 프로그램이 다양하게 개발되면서 QR 코드의 활용이 늘고 있다. 하지만 QR 코드는 사람이 인식할 수 없기 때문에 안전성 검증이 쉽지 않고, 누군가 악의적으로 변경할 가능성이 있다. [2]

본 논문에서는 강원대학교 이남세 석사가 제안한 PS-Net[1](개인별 보안 Wi-Fi 네트워크)에서 아이디어를 착안하였다. PS-Net에 상업적 측면을 추가하고, 보안을 더욱 강화하는 방안을 제시한다. 사용자 중심 네트워크로 동적 QR 코드를 이용함으로써 악의적으로 QR 코드를 변경하는 것을 방지할 수 있다.

1.1 관련 연구

Fig. 1. Composition of Ps-Net using QR codes

사용자는 정보를 입력하여 보안을 설정함으로써 사용자 맞춤 가상 Wi-Fi 네트워크를 형성할 수 있다. 우선 QR 코드화한 AP(Access Point) 공개키(public key)를 기기로 인식하여 키를 취득하고, 어플을 이용하여 사용자로부터 SSID와 비밀번호를 입력받는다. 이후 AP 공개키와 입력받은 정보를 AP에 전달한다. 이 정보를 AP에 전송하기 위해서 public channel이 사용되는데, 이는 보안에 취약하다. 따라서 Fig. 1의 ③에서 RSA 암호화 알고리즘을 사용해 암호화를 해야 한다. 이후 AP는 개인키(private key)를 사용하여 암호화된 정보를 해독한다. 복호화된 정보를 이용하여 새로운 가상 AP를 생성하고, 사용자는 가상 Wi-Fi 네트워크를 이용할 수 있게 된다.

* 교신저자 : hwajeong84@gmail.com

II. 본론

기존의 Wi-Fi 인증 방식은 네트워크에 설정된 비밀번호를 입력하는 '네트워크 중심 인증 방식'인 반면에, PS-Net과 DQ-Net은 '사용자 중심 인증 방식'으로, 사용자가 SSID와 암호를 설정하여 네트워크를 사용한다. 하지만 DQ-Net은 PS-Net과 달리 어플을 설치하지 않고 public channel인 일회성 사이트를 제공한다. 이 사이트는 인증만을 위해 사용하는 것이기 때문에 용도를 다하면 자동으로 소멸한다. 또, PS-Net은 AP 기기 근처에 종이로 출력된 QR 코드를 공개하는 반면, DQ-Net은 디스플레이가 설치된 AP 기기 화면에서 QR 코드를 제공한다.

2.1 DQ-Net 구성 제안

Fig. 2. Suggestions of DQ-Net composition
① Output dynamic QR codes to the display
② Scan the QR codes with the Smart Phone
→ Obtain AP public key & Link to disposable page
③ Enter the information(SSID, P/W) at the site AP via public channel
④ Decode the data with AP private key
→ Create and provide virtual Wi-Fi Networks
→ Cease to exist the page

Fig. 2는 동적 QR 코드와 일회용 사이트를 이용한 개인별 네트워크인 DQ-Net의 연결 과정을 보여준다. AP는 공개키와 일회용 URL을 동적 QR 코드로 생성하여 디스플레이에 출력한다. 사용자는 자신의 기기로 이를 스캔하여 AP의 공개키를 얻게 되고, 자동적으로 일회성 페이지에 연결된다. 사용자가 일회성 페이지에서 요구하는 SSID와 암호를 입력하면, 이 정보들은 RSA 암호화 되어 public channel을 통해 AP로 전달된다. AP는 개인키를 이용하여 정보를 해독한 후 사용자에게 가상 Wi-Fi네트워크를 생성 제공하고, 이때 일회성 페이지는 자동 소멸한다.

2.1.1 일회성 페이지

DQ-Net은 일회성 페이지를 사용하여 SSID나 비밀번호와 같은 정보를 입력한다. 이 페이지에서 가장 중요한 점은 SSID 이름의 중복확인을 해준다는 점이다. 공유기의 초기 값을 변경하지 않거나, Fig. 3처럼 동일한 또는 가까운 채널에 같은 SSID의 Wi-Fi가 있다면 네트워크가 충돌해 느려지게 된다. 따라서 SSID 이름에 대한 중복확인은 네트워크의 느려지는 문제를 해결할 수 있다.

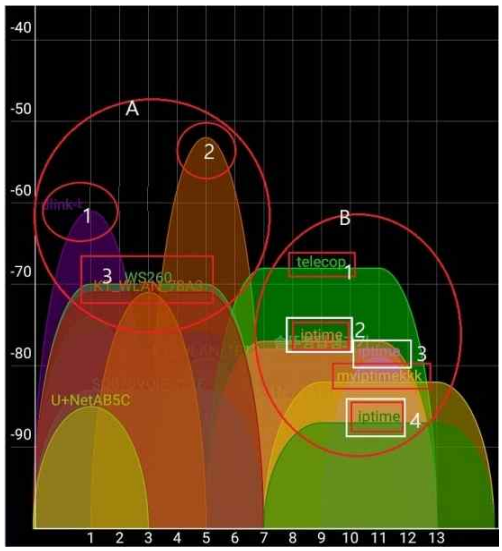


Fig. 3. For Wi-Fi with the same name of SSID on the same channel[4]

2.1.2 OTP (One Time Password)

OTP란 랜덤하게 만들어지는 난수를 일회용 비밀번호로 이용하는 것이다. 기존의 Wi-Fi는 보안성을 유지하기 위해 일정 기간마다 비밀번호를 변경해야 한다. 하지만 대부분의 사람들은 번거로움 때문에 변경하지 않는 경우가 많다. 이런 Wi-Fi를 공공장소에서 이용하면 시간이 지날수록 비밀번호를 아는 사람들의 숫자가 늘어나기 때문에 보안이 취약해진다. 그러므로 본 논문에서는 OTP 개념을 사용한다.

2.1.3 동적 QR 코드(Dynamic QR codes)

QR 코드는 누구나 자유롭게 생성할 수 있기 때문에 악성코드나 유해 웹사이트 주소도 전파가 가능하다. 이러한 악성 QR 코드는 사용자가 육안이

나 어플로 인지할 수 없기 때문에[6] 누군가 나쁜 의도를 가지고 QR 코드를 교체할 경우 큰 피해 상황이 발생할 가능성이 높다. 이것은 정적 QR 코드의 가장 취약한 점이며 많은 문제의 원인이 된다.

2.2 모드

DQ-Net은 장소에 따라 Wi-Fi 접속 방법을 달리하여 보안성을 유지할 수 있도록 하였다. 관리자 모드, 사용자 모드, 게스트 모드 세 가지로 나누었다.

2.2.1 관리자 모드 (Administrator Mode)

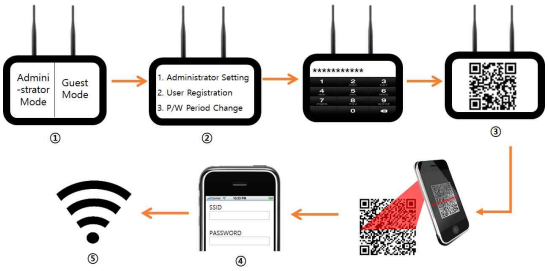


Fig. 4. Administrator Mode of DQ-Net
① Touch the Administrator Mode to the display
② Touch the Administrator Setting → Confirm the OTP number
③ Output dynamic QR codes to the display → Scan the QR codes with the Smart Phone → Link to disposable page
④ Enter the information(SSID, P/W) at the site → RSA Encryption → Convey information to AP via public channel
⑤ Decode the data with AP private key → Create and provide virtual Wi-Fi Networks

관리자 모드는 일반적인 네트워크 연결 과정과 상당히 유사하다. 관리자란, 네트워크를 설치한 이후 최초로 기기를 등록한 사람에게 부여되는 권한이다. 관리자는 OTP 카드를 갖게 되는데, 이 카드는 처음 기기 등록과 비밀번호 바뀌는 주기 변경, 사용자 권한 부여 작업에 사용된다. 처음 AP기기를 켠 후 관리자 모드를 선택한다. 이후에 뜨는 하위 메뉴에서 관리자 등록을 누르고 키패드가 뜨면, 이용자는 OTP 번호를 입력하여 자신이 관리자라는 것을 인증하다. 일회성 등록 사이트로 이동할 수 있는 QR 코드가 디스플레이에 표시되면 이용자는 스마트폰을 통해 QR 코드를 인식하여 일회성 페이지로 접속한다. SSID와 비밀번호 등 필요한 정보들을 입력하고 비밀번호가 자동으로 바뀌는 주기를 설정한다.

2.2.2 사용자 모드 (User Mode)

사용자 모드는 게스트 모드와 달리 지속적으로 네트워크를 이용할 수 있다. 사용자는 관리자와 동일한 네트워크를 사용하기 때문에 관리자로부터 권한을 부여받아야한다. 관리자의 가상 네트워크망을 공유하기 때문에, 별도의 네트워크 생성과정이 필요하지 않다.

Fig. 5. User Mode of DQ-Net
① Touch the Administrator Mode to the display
② Touch the User Setting → Output dynamic QR codes to the display → Scan the QR codes with the Smart Phone
③ Create and provide virtual Wi-Fi Networks

우선 사용자는 AP 기기의 디스플레이에서 관리자 모드를 선택하고, 하위 메뉴에서 사용자 등록을 택한다. 사용자 등록을 누르면 번호를 입력할 수 있는 터치 키패드가 나오는데, 이곳에 관리자의 OTP카드의 번호를 입력한다. 확인이 완료되면 관리자의 네트워크 정보가 담긴 QR 코드가 생성된다. 사용자의 기기로 이 QR 코드를 스캔하고 네트워크 접속을 묻는 알림에 확인을 누르면 네트워크 연결이 완료된다. 그 과정은 Fig.5와 같다.

2.2.3 게스트 모드 (Guest Mode)

게스트 모드는 가상 네트워크를 지속적으로 사용하지 않는 이용자를 위한 모드이다. 따라서 관리자 모드, 사용자 모드와는 달리 연결이 끊길 경우 자동적으로 가상 Wi-Fi 공간이 소멸된다. 다른 모드들과는 달리 게스트 모드는 가정용과 상업용으로 나뉘는데, 용도에 따라 사용 방법 면에서 차이를 보인다.

가정용의 경우 AP의 디스플레이에서 게스트 모드를 터치하면 일회성 사이트로 이동할 수

있는 QR 코드가 생성된다. 게스트는 이 QR 코드를 스캔하여 일회성 사이트에 접속하고 SSID와 비밀번호를 설정한다. 페이지에서 확인을 누르면 가상 네트워크망이 생성되고 자동으로 게스트의 스마트폰에서 네트워크 연결이 된다.

상업용의 경우 AP기기와 연동이 되는 기계를 각 카운터에 두어 버튼을 누르면 QR 코드가 출력된 스티커가 나오게 할 수 있다. 이 부착물을 가상 네트워크를 사용하는 게스트의 영수증에 붙여 전달하고, 그 이후의 과정은 가정용에서의 과정과 일치한다.

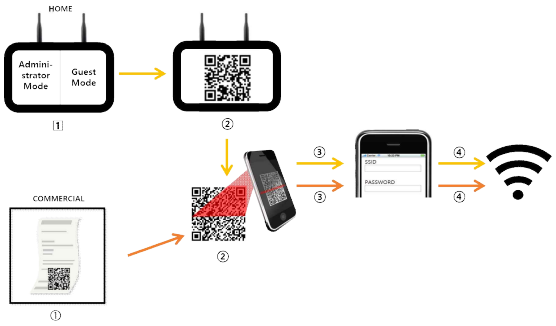


Fig. 6. Guest Mode of DQ-Net (For Home & Commercial Use)
1 Touch the Guest Mode to the display → Output dynamic QR codes to the display
① Attach QR codes sticker to receipt or product
② Scan the QR codes with the Smart Phone → Link to disposable page
③ Enter the information(SSID, P/W) at the site → RSA Encryption → Convey information to AP via public channel
④ Decode the data with AP private key → Create and provide virtual Wi-Fi Networks

2.3 성능평가

Table. 1. Improvement result

	As-Found	Suggestion
Security Performance	LOW	HIGH
Convenience	LOW	HIGH

본 논문에서는 동적 QR 코드를 사용함으로써 정적 QR 코드의 보안적 문제점 해결방안을 제시한다. 또한 사용자의 정보를 입력하는 과정에서 일회용 사이트를 이용하는 것은 애플리케이션의 설치를 필요로 하지 않아 사용이 간편하게 만들어준다. 모드를 분류하여 네트워크를 사용하는 것은 관리를 용이하게 한다. 관리자 모드를 통해 관리자에게만 OTP카드의 이용 및 관리 권한을 부여하는 것은 보안성을 높이는 방법이며, 이용시간이 길지 않은 공공장소 등에서의 게스트 모드 사용은 네트워크 이용을 편리하게 한다.

III. 결론

본 논문에서는 사용자 중심 네트워크를 제안한다. 사용자가 스스로 AP를 설정하여 Wi-Fi 네트워크를 이용한다. 이에 사용되는 Wi-Fi 환경은 보다 구체적이고 보안성이 향상된 방안들을 제시한다. 또한, Wi-Fi를 사용하는 사람들의 목적에 따라 모드를 나눠 보안성을 유지할 수 있는 방안을 고려했다. 모드는 총 세 가지로 나누어졌다.

향후 이 기술이 보편적으로 사용되게 된다면 애플리케이션 설치 없이 보안성이 유지된 상태로 Wi-Fi 네트워크를 이용할 수 있게 될 것이다.

[참고문헌]

[1] Nam-seh Lee, Ju-ho Lee, Choong-Kyo Jeong, "PS-Net : Personalized Secure Wi-Fi Network," *Journal of Korean Institute of Communications and Information Sciences*, 40(3) pp. 497-505, Mar. 2015.

[2] Hyung-Kyu Yang, "A study of Security Weaknesses of QR Codes and Its Countermeasures," *Journal of The Institute of Internet, Broadcasting and Communication*, 12(1), Feb. 2012.

[3] Jae-Kyung Park, "Globl Report - Invasion of Wi-Fi security and current status of security techniques", *Journal of Radio Spectrum & Communications*, June. 2010.

[4] SSID, https://bit.ly/2TERI1D

[5] "무료 와이파이 사용자 노트북 해킹당해...은행 잔고 털려," Daily Secu, Accessed: 2014-04-15.

[6] QR code, https://bit.ly/2r2nAA5, Accessed : 2011-05-24

OpenMP를 활용한 LSH DRBG 병렬 최적 구현

권혁동* 안규황* 서화정*†

*한성대학교 대학원 정보시스템공학과

hdgwon@naver.com tiger9212@gmail.com hwajeong84@gmail.com

LSH DRBG parallel optimization using OpenMP

Hyeok-Dong Kwon* Kyu-Hwang An* Hwa-Jeong seo*†

*Information system engineering, Hansung University of Graduate school.

요 약

결정론적 난수 발생기(Deterministic Random Bit Generator, DRBG)는 미국 표준 기술 연구소(National Institute of Standards and Technology, NIST)에서 권고하는 결정론적 난수 발생 알고리즘으로 입력에 따라 의사난수를 생성하는 알고리즘이다. DRBG의 내부에는 의사난수함수를 사용하는데, 아직까지 표준으로 제정 및 구현된 적 없는 LSH DRBG를 구현하며, 더 나아가 DRBG의 규격 별로 또는 DRBG의 함수 중 유도함수와 내부출력생성함수에서 반복적으로 호출 되는 의사난수함수를 OpenMP를 통해 병렬화 처리를 적용하여 최적화를 하고자 한다.

I. 서론

해시 함수 입력 값의 길이와 무관하게 고정된 길이의 임의 값을 출력하는 함수로 이를 활용한 알고리즘 중 하나가 난수 발생기이다. 난수 발생기는 무작위 값을 제작하지만 입력에 따라 출력 값이 고정된 결정론적 성질을 가지며 내부의 의사난수함수를 교체하는 것으로 출력을 변경할 수 있다.

일반적인 프로그램은 코드의 상단부터 절차적으로 진행된다. 그러나 코드의 중간 결과물이 서로 독립적이라면 각 출력물을 대기하지 않고 병렬적으로 동작하는 구조가 효율적일 수 있다. 이와 관련된 예시가 fig. 1으로, 동일한 함수 func1, func2를 호출한다고 가정하면 일반적인 프로그램은 func1이 종료되어야 func2가 실행될 수 있다. 하지만 병렬화를 적용한 프로그램의 경우에는 func1과 func2가 동시에 동작하기에 전체 동작 시간이 줄어들게 된다.

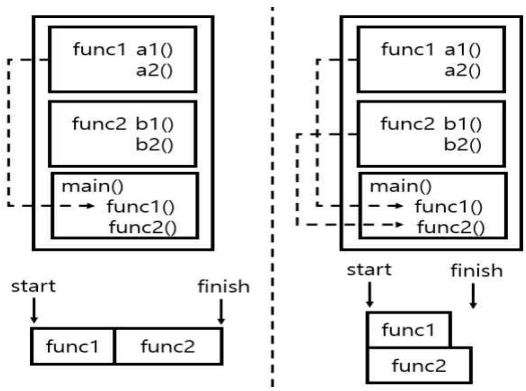


Fig. 1. 좌) 일반적인 프로그램의 구조
우) 병렬화가 적용된 프로그램의 구조

본 논문에서 사용할 결정론적 난수 발생기(deterministic random bit generator, DRBG)는 LSH를 사용하여 총 6개 규격을 지원하기에 각 규격 출력에 병렬화 시도를 한다. 또한 동작 중에는 의사난수함수를 반복적으로 호출하는 과정이 있으므로 내부 동작에도 병렬화를 적용하여 최적화를 구현한다.

II. 관련 연구 동향

2.1. 결정론적 난수 발생기(Deterministic Random Bit Generator)

일반적으로 난수는 예측 불가능한 무작위의 수를 의미하나 실제로는 컴퓨터에서 완전 난수를 생성하기 어렵기 때문에 입력 값에 따라 출력이 정해진 결정론적으로 구성된 난수 발생기를 의미한다.

일반적으로 내부에 적용하는 의사난수함수의 종류에 따라 해시기반 DRBG와 HMAC기반 DRBG로 나뉘지만 둘의 내부 구조는 다르며 해시기반 DRBG가 병렬화 기법을 적용하기 유리하기 때문에 본 논문에서 언급하는 모든 DRBG는 해시기반 DRBG임을 명시한다.

2.2. LSH(Lightweight Secure Hash)[2]

LSH는 국가보안기술연구소(National Security Research Institute, NSR)에서 2014년에 개발한 암호학적 해시 함수로 wide-pipe merkle Damgård 구조가 적용되었다. 출력 규격으로는 224, 256, 384, 512비트를 제공한다. 성능면에서 SHA2, SHA3에 비해 약 2배 이상 성능이 뛰어남을 보였으며 현재 국내 TTA 표준으로 제정되어 있다.

2.3. SSE(Steaming Simd Extensions)[3]

SSE는 Intel사에서 제공하는 병렬화 기법 중 하나로 1999년에 출시된 펜티엄3부터 사용 가능하며, SIMD(Single Instruction Multiple Data)에 속한다. SIMD는 하나의 명령어로 절차지향적으로 수행될 데이터를 한 번에 처리할 수 있기에 전체적인 동작속도를 향상시킨다.

2.4. AVX(Advanced Vector Extension)[4]

AVX는 2008년에 발표된 SIMD 명령어 집합으로 기존 16바이트에서 증가한 32바이트를 한 번에 계산할 수 있다. 특히 AVX는 부동소수점 연산이 뛰어나기 때문에 이와 관련된 모델링, 실험 등의 분야에서 유용하게 활용할 수 있다.

III. 제안 기법

구현한 LSH DRBG에 병렬화를 적용하는 방법에는 데이터 병렬화와 태스크 병렬화 두 가지 방법이 있으며 본 장에서는 각각의 적용 방법을 서술한다.

3.1. 데이터 병렬화(Data Parallelism)

LSH는 총 6개의 규격을 지원하며 프로그램 구조상 후행 규격은 선행 규격의 출력을 대기하게 된다. 따라서 출력을 대기하는 시간이 소요된다. 이러한 구성은 대기 시간이 누적되어 작업 시간을 증가를 초래한다. 하지만 DRBG의 각 출력 값은 서로 독립적이기 때문에 선행 규격의 완료 여부와 상관없이 작업을 진행해도 무방하다. 단, 각 규격별로 작업 시간이 동일하지는 않기에 프로그램이 종료되기 위해서는 모든 규격이 완료되기를 기다려야 한다. 이것을 적용한 모델이 fig. 2로 각 규격별로 병렬적으로 동작하며 모든 규격이 종료되기를 대기하는 시간이 새롭게 포함된다. 새롭게 생긴 대기 시간은 기존의 누적되는 대기 시간에 비해 짧을 것으로 예상되기에 동작 소요 시간을 월등히 절감할 것으로 기대할 수 있다.

```
Algorithm 1
1. read test vectors
2. loop count = number of hash output types
3. operate OpenMP to for loop
4. For i = 1 to loop count
   4.1 DRBG using hash[i] type
   4.2 write DRBG result
   4.3 waiting until other processing finished
5. DRBG finished
```

Fig. 2. 데이터 병렬화가 적용된 LSH DRBG

3.2. 태스크 병렬화(Task Parallelism)

DRBG의 구조 중에서 유도함수와 내부출력생성함수는 내부에서 반복적인 해시 연산을 수행하며 반복 산정 기준은 함수마다 다르다. 유도함수의 반복 횟수 산정에는 규격별로 정해진 상수 값만이 사용되기 때문에 2회 또는 3회로 고정된다. 반면 내부출력생성함수의 반복 산정

기준은 출력 길이에 비례하기 때문에 내부출력 생성함수의 반복 횟수는 출력 길이를 변경하는 것으로 조절이 가능하다. 두 함수 자체는 독립적이며 또한 내부의 해시 연산 작업도 서로 영향을 주지 않기 때문에 병렬화 적용이 가능하며 fig. 3, fig. 4와 같은 모습을 가진다.

```
Algorithm 2
1.1 If hash bits is 224 or 256
    seed bits = 440
1.2 If hash bits is 384 or 512
    seed bits = 888
2. len = ceil (seed bits / hash bits)
3. counter = 0x01
4. input = counter || seed bits || msg from parameter
5. operate OpenMP to for loop
6. For i = 1 to len
   6.1 output[i] = Hash(input)
   6.2 counter += 1
   6.3 refresh input value
   6.4 waiting until other processing finished
7. final output = (output[1] || … || output[n]) mod 2^seed bits
```

Fig. 3. 태스크 병렬화가 적용된 유도함수

```
Algorithm 3
1. len = ceil (output bits from parameter / hash bits)
2.1 If hash bits is 224 or 256
    seed bits = 440
2.2 If hash bits is 384 or 512
    seed bits = 888
3. data = state V from parameter
4. operate OpenMP to for loop
5. For i = 1 to len
   5.1 output[i] = Hash(data)
   5.2 data = (data + 1) mod 2^seed bits
   5.3 waiting until other processing finished
6. final output = (output[1] || … || output[n]) mod 2^output bits
```

Fig. 4. 태스크 병렬화가 적용된 내부출력생성함수

IV. 성능평가

본 장에서는 데이터 병렬화와 태스크 병렬화에 따른 성능 변화를 비교할 것이며 DRBG 구현 및 작업 환경은 다음 table. 1과 같다.

Table. 1. 작업 환경

OS	Windows 10 Pro
Processor	Intel Core i7-8550U CPU 1.8GHz
Compiler	MinGW
IDE	Eclipse Photon (4.8.0)
Language	C

테스트에 사용된 데이터와 DRBG 구현 작업물은 깃허브[1]에서 열람이 가능하다.

4.1 데이터 병렬화 성능 평가

각 규격별 동작에 대해 병렬화를 적용하여 6개 규격이 동시에 출력을 진행한다. 6개 규격 출력이 완료된 시점이 1회이며 규격별로 실험 환경은 테스트벡터 60종을 1000회에 반복해서 출력하게 설정하며 예측내성 지원모델로 동작한다. 대조군은 병렬화를 적용하지 않고 동일하게 동작하며 실험 결과는 table. 2와 같다.

Table. 2. 데이터 병렬화 비교

	평균수행시간(ms)	cpb
대조군	73	491
병렬화	27	181

대조군의 수행 시간을 CPB(Clock cycle Per Bytes) 단위로 환산하면 약 491cpb가 나온다. 반면 병렬화를 적용한 경우에는 약 181cpb의 효율을 보여 약 2.71배의 성능 향상을 보인다. 대조군은 출력 대기 시간이 누적되지만 병렬화 모델은 그렇지 않기에 그만큼 빠른 동작을 보장한다.

4.2 태스크 병렬화 성능 평가

DRBG의 내부 구조상 병렬화가 적용 가능한 두 부분에 병렬화를 적용하여 해시 연산이 동시에 이뤄지게 한다. 4.1절과 동일한 환경에서 진행하며 마찬가지로 모든 규격이 완료된 시점을 1회로 한다. 실험 결과는 table. 3과 같다.

Table. 3. 태스크 병렬화 비교

	평균수행시간(ms)	cpb
대조군	73	491
병렬화	149	1001

대조군은 데이터 병렬화의 대조군과 유사한 결과로 약 491cpb를 보여준다. 반면 병렬화 적

1) https://github.com/korlethean/drbg_with_mp

용 모델은 1001cpb가 도출되며 약 2.04배 가량 부족한 성능을 보인다.

이는 병렬화 과정의 오버헤드로 성능이 저하되는 것으로 판단되었다. 병렬화 과정 중에 오버헤드의 발생은 필연적이기 때문에 병렬화 효율을 높이기 위해서는 반복의 횟수를 증가시켜야 한다. DRBG 구조상 출력 길이 조절로 내부 출력생성함수의 반복 횟수를 늘릴 수 있으며 이에 따라 유도함수 내부는 통상적인 동작으로 수정하고 출력 길이를 점진적으로 늘려가며 실험한 결과는 fig. 5와 같다.

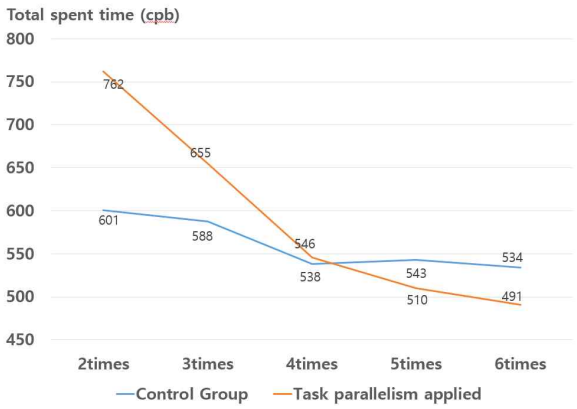


Fig. 5. 출력 길이별 비교 그래프

출력 길이가 기존대비 2, 3배인 경우에는 대조군의 효율이 뛰어나지만 4배의 경우엔 근소한 차이만을 보이며 5배 이후부터 병렬화 모델이 뛰어남을 보인다. 또한 병렬화 모델은 출력 길이가 증가함에도 동작 효율의 변화가 크게 변하지 않음을 확인할 수 있다.

V. 결론

본 논문에서는 DRBG의 구조상에서 병렬화 모델을 적용하여 최적화 구현을 시도하였고 그 성능이 통상적인 DRBG 모델보다 뛰어남을 확인하였다.

데이터 병렬화를 적용할 경우 여러 규격의 DRBG를 동시에 가동할 수 있기에 다수 규격 출력에 유리함을 보인다. 반면 태스크 병렬화의 경우에는 표준 규격 상에서는 효과적이지 못하나 일정 출력 길이 이상에서는 병렬화 적용이 뛰어남을 보였다. 이는 표준과는 다르기 때문에 실제로 사용하기는 어렵지만 이론상으로 DRBG에 태스크 병렬화 적용이 가능함을 보였다.

각종 병렬화 기법은 다양한 영역에서 활용된다. 이때 단지 병렬화 기법을 적용하는 것만으로 성능 향상을 기대하게 되나 본 논문의 실험 결과와 같이 오버헤드로 인해 성능이 저하되는 일이 발생할 수 있다. 따라서 병렬화 기법을 맹목적으로 적용하여 성능 향상을 꾀하려 하지 않으며 통상적인 동작과 저울질 끝에 정확한 적용을 시도하여야만 성능 향상을 이끌어 낼 수 있다.

[참고문헌]

[1] National Institute of Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", Available: https://www.nist.gov/publications/recommendation-random-number-generation-using-deterministic-random-bit-generators-2

[2] Telecommunications Technology Association, "Hash function Full structure and compression function of LSH", Available: http://commitee.tta.or.kr/data/standard_view.jsp?nowPage=2&pk_num=TTAK.KO-12.0276&commit_code=TC5.

[3] Srinivas K. Raman, Vladimir Pentkovski, Jagannath Keshava, "Implementing Streaming SIMD Extensions on the Pentium III processor", Available: https://www.computer.org/csdl/mags/mi/2000/04/m4047.pdf

[4] Chris Lomont, "Introduction to Intel® Advanced Vector Extensions", Available: https://software.intel.com/sites/default/files/m/d/4/1/d/8/Intro_to_Intel_AVX.pdf

파일 고유 식별 정보 기반 통합 유사도 시스템 설계

이동주*, 강성묵** ,안건우***, 김동욱****, 유기환*****,이현******
김기홍*******, 곽경주********

*순천향대학교 , **강남대학교 , ***대구대학교 , ****창원과학고등학교,
*****한국디지털미디어고등학교 , ******선린인터넷고등학교,
*******세인트시큐리티, ********금융보안원

Integrated Code Similarity Analysis System based on File Identification Information

Dong-Ju Lee*, Seong-muk Gang**, Kun-woo Ahn***, Dong-Uk Kim****, Ki-Hwan Yu*****, Hyun Yi******, Ki-Hong Kim*******, Kyung-Ju Kwak********

*Soonchunhyang University,**Kangnam University, ***Daegu university,

****Changwon Science Highschool. *****Korea Digital Media highschool.

******Sunrin Internet High School, *******SAINTSECURITY Inc.

********Financial Security Institute

요 약

오늘날 국내·외 기업들은 시스템으로부터 유입되는 악성코드에 대한 사이버 위협에 직면하고 있다. 악성코드를 활용한 사이버 침해 사고는 개인 혹은 팀으로 활동하고 있는 것으로 추정된다. 이에 따라 보안 인텔리전스를 구성하고 있는 기관들은 악성코드의 다양한 속성 및 특징을 분석하여 사이버 침해 사고를 그룹별로 분류하고 있다. 본 논문에서는 PE 파일의 고유 식별 정보를 기반한 통합 유사도 분석 시스템을 개발하고, 이를 활용하여 악성코드의 제작 그룹 및 관련된 정보를 정확하게 분류 제공함으로서 분석의 실용성을 향상시키고, 추후 연구 방향에 대하여 제시하며 끝을 맺는다.

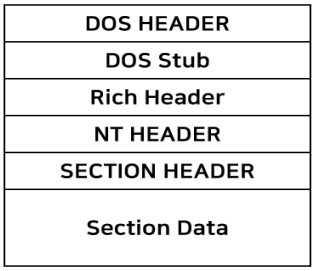
I. 서론

4차 산업혁명을 맞이하며 인터넷이 실생활의 필수적인 요소로 활용됨에 따라 개인 정보나 금융 정보와 같은 민감한 정보들이 인터넷을 통해 전달되며 이러한 정보를 다루는 기관일수록 각종 사이버 침해 사고 대상이 되어 악성코드의 주요 공격 대상이 된다. 최근 사이버 침해는 개인이 아닌 집단으로 수행되는 것으로 추정되며 이에 따라 지능화되고 고도화된 변종이 지속적으로 생성되고 있다. 이러한 특정 집단에 의해 생성된 변종 악성코드를 각 기관에서 초기에 대응하기 위해서는 신뢰성이 보장된 유사도 분석 도구를 활용하여 대처할 수 있어야 한다. 본 논문의 통합 유사도 시스템을 활용하여 악성코드의 변종을 정확하게 판별하며 모집단 정보를 도출하는 시스템을 제시한다. 통합 유사도 시스템에서는 PE(Portable Executable) 파일에서 추출한 문자열, 컴파일(Compile) 정보, 베이직 블록(BasicBlock), PE 헤더 정보 등을 활용하여 유사도 측정을 수행한다. 또한, 정확한 지표를 제시하기 위해 각 유사도 측정 기법에 신뢰성 가중치를 적용하여 악성코드의 변종 여부를 정확하게 분류한다. 본 논문의 구성은 2장에서는 PE 실행 파일에 포함된 고유 식별 정보에 대한 배경지식에 대해 설명한다. 3장에서는 통합 유사도 시스템의 전체적인 구성을 기술한다. 4장에서는 시스템을 바탕으로 평가 및 실험한 결과를 기술한다. 마지막으로 5장에서는 결론과 향후 계획에 대하여 설명하고 마치도록 한다.

II. 파일 고유 식별 정보

2.1 PE 헤더 정보

PE 헤더란 실행 가능한 바이너리(Binary) 파일인 윈도우 운영체제의 실행 파일 (DLL, EXE 등)이 윈도우 로더(Loader)에 의해 구동될 때 필요한 정보가 속해있는 데이터 구조체이며 여러 가지 리소스를 할당하거나 제작자의 코드 문자열 등에 대한 정보를 갖는다. PE 헤더의 구조도는 아래 그림 1과 같다.



[그림 1] PE Header Structure

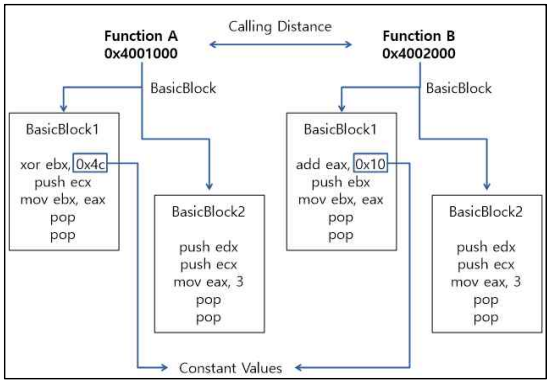
PE 헤더에서 제공되는 정보를 기반으로 유사도를 측정하기 위해서는 분류가 가능한 고유 식별 정보를 통해 악성코드 제작자 및 제작 그룹을 판단하는 중요 요소[1]로 활용할 수 있다. 아래 표 1은 PE 헤더 정보 중 본 유사도 분석 시스템에서 활용된 요소들의 목록이며 활용 방안은 3장에서 다루도록 한다.

	Identification
1	Rich Header
2	Debug Data (GUID, AGE, PDB Path)
3	Section Entropy
4	Imp(Import) Hash
5	Code Sign
6	File Hash
7	SSDEEP Hash
8	.Text Strings

[표 1] PE Header list for Similarity Analysis

2.2 PE 파일 식별 정보

PE 파일 유사도 분석은 식별 대상 파일을 비교하기 위해 본 유사도 분석 시스템에서는 아래 그림 2와 같이 여러 방면으로 식별 요소를 선정하였다.



[그림 2] Binary identification element

그림 2의 베이직 블록은 코드의 한 부분으로 이를 블록 단위로 의미를 정형화하면 악성코드의 암호화, 복호화 방식에 대한 루틴 혹은 악성코드 제작자의 고유 코드 작성 패턴 등에 대한 정보를 수집할 수 있다. 또한, 함수 내에 포함된 제작자가 정의한 상숫값 혹은 문자열은 벡터값 ,복호화 키, C&C(Command & Control) 주소, 네트워크 주소 등으로 이러한 값은 주식별 요소의 대상이 된다.

III. 유사도 분석 기법 소개

앞서 2장에서 설명한 식별 가능한 요소를 기반으로 유사도 시스템에 적용된 기법을 소개한다. 정형화된 베이직 블록을 이용한 방식으로는 N-gram 비교 방법(3.2절), 함수 대푯값 비교 방법(3.3절)이 있으며 패턴 정보를 기반한 방식은 상숫값 비교 방법(3.4절) 마지막으로 SSDEEP Fuzzy Hashing 비교 방법(3.5절), PE 헤더 비교 방법(3.6절)을 소개하고 3장을 마친다.

3.1 유사도 알고리즘 비교 사항

본 시스템에 상기의 유사도 알고리즘을 적용하기 위해 비슷한 유형의 25개 샘플을 기반으로 효율성 및 성능 연구를 진행하였다. Fuzzy Hash 기법을 수행하는 SSDEEP 알고리즘은 일정 크기 단위로 각 단위 블록에 대한 해시 값을 생성하여 비교를 수행하기에 유사도 측정률이 타 알고리즘보다 상대적으로 높지만 베이직 블록 단위로 측정을 수행할 시 비교 단위의 양이 적어지므로 측정률이 낮다.

반대로 Levenshtein, N-gram, Jaccard 같은 문자열 알고리즘은 문자열의 배열을 기반으로 측정을 수행하여 파일 자체에 대한 측정률은 낮지만 베이직 블록 단위는 상대적으로 높았으며 그 중 N-gram 유사도 알고리즘의 효율성이 가장 우수하였다.

3.2 베이직 블록 N-gram 비교

N-gram은 문자열 유사도 측정을 N개의 블록으로 나눈 결과를 이용하는 알고리즘[2]을 말한다. 정형화된 베이직 블록의 opcode를 추출하여 이를 하나의 문자열로 선정하여 길이를 비교할 시 길이가 긴 문자열일수록 유사도가 높게 측정될 가능성이 크기에 문자열의 길이와 비교 점수를 나눈 결과를 유사도 값으로 선정한다.

3.3 함수 대푯값 비교 방법

함수 대푯값은 베이직 블록의 opcode를 hex로 변환하여 이를 모두 더한 뒤 해시화 한 결과 값을 통해 유사도 매칭 작업을 수행한다. 악성코드 제작자의 코드 재사용에 대한 식별이 가능할 뿐만 아니라, 블록 내 opcode의 일부 순서가 변경이 되어도 대응이 가능해진다.

3.4 상수 값 비교 방법

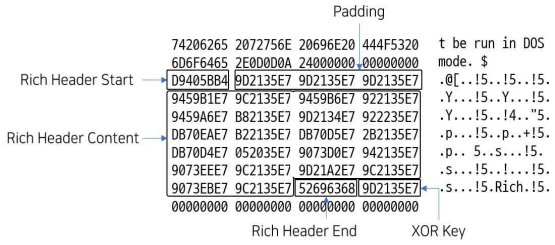
윈도우 실행 파일에서 사용된 상수 목록을 추출하여 리스트에 삽입한다. 리스트의 각 요소는 악성코드 제작에 활용된 복호화 키, 벡터 값, 뮤텍스 이름 등 다양한 요소로 활용되는 상수가 포함되어 있으며 이를 기반으로 유사도 분석을 수행한다.

3.5 SSDEEP Fuzzy Hashing 비교 방법

SSDEEP은 문자열 혹은 바이너리 코드를 일정한 크기로 블록화를 하며 파일 혹은 긴 문자열에 대한 유사도를 측정할 수 있다.

3.6 PE 헤더 비교 방법

2장 표 1의 식별 정보 사항을 보면 Rich Header는 실행 파일을 제작 시 사용된 컴파일러(Compiler) 정보가 포함되어 있다.

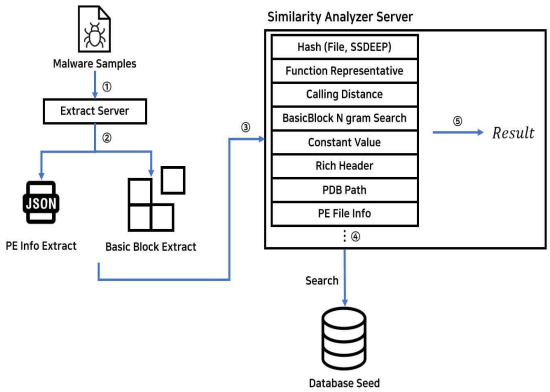


[그림 3] Rich Header Structure

Rich Header의 구조는 그림 3과 같으며, Header Start, Padding, Header Content, Header End, XOR Key로 구성되어있다. Content에 대한 내용을 XOR Key로 복호화하게 될 시 위에서 언급한 제작자의 제작 환경에 대한 컴파일러 정보, 오브젝트 정보, 링커 정보 등을 추출하여 이를 통해 악성코드 유사도 분석에 활용할 수 있다.

PDB 파일은 프로그램의 디버깅 및 프로젝트의 상태 정보인 프로젝트 소스코드 내 변수들의 이름과 위치, 소스 파일 정보가 내포되어 있다. PE 헤더에 포함된 IMAGE_DEBUG_TYPE_CODEVIEW 내에 PDB 경로, 빌드 횟수를 나타내는 age, 프로젝트 생성 시 할당되는 시그니처인 GUID가 있어 악성코드 제작자의 제작 환경을 기반으로 유사도 분석에 활용[3]할 수 있다.

IV. 통합 유사도 분석 시스템 구현



[그림 4] Similarity Analysis System Diagram

본 유사도 분석 시스템은 그림 4와 같이 과정 ①을 통해 샘플의 PE 여부를 확인하고 과정 ②을 통해 PE 헤더 정보와 함수 정보 추출기를 활용하여 베이직 블록 정보를 추출한다. 과정 ③은 추출된 정보를 유사도

비교를 수행하기 위해 유사도 분석 서버로 전달한다. 이후 과정 ④에서 추출한 정보와 기존 패턴 정보가 저장되어 있는 Seed 데이터베이스와 비교한다. 3장에서 설명된 각 유사도 기법을 통해 유사도 결과를 도출하며 최종적으로 과정 ⑤에서 각 유사도 기법에 대한 신뢰성을 기반으로 가중치[4]를 적용하여 최적의 유사도 결과를 도출해낸다. 다음은 유사도 기법별 신뢰성에 대한 가중치 부여 방식에 대한 공식이다.

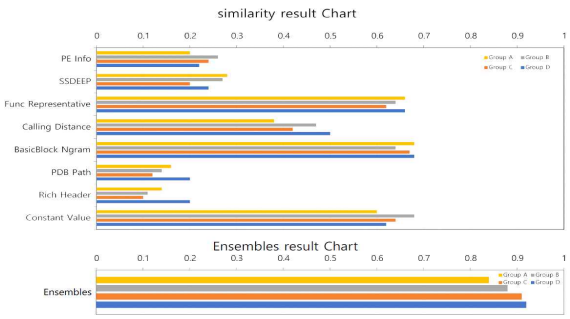
$$S_j = \sum_{i=1}^{n} G_{(i,j)} W_i + R_j$$

$$R: \begin{cases} 0.05\,(result=1) \\ 0\,(result=0) \end{cases}$$

n : 유사도 분석 총 개수
m : 데이터베이스에 레이블링된 그룹의 개수
i : 유사도 분석 단계
j : $\{j$는 자연수, $1 \le j \le m\}$ 악성 그룹 인덱스
$G_{(i,j)}$: i번째 유사도 분석 단계에서 G_j의 점유율
S_j : 샘플에 대한 j번째 악성 그룹 유사도
W_j : 유사도별 가중치

(1)

V. 통합 유사도 시스템 분석 결과



[그림 5] Similarity Analysis Result

그림 5의 그래프는 악성코드 제작 그룹별로 유사도 시스템의 데이터베이스와 비교하여 얻은 통합 유사도 수치 및 앞서 설명한 각 유사도 기법별 수치를 나타낸다.

일반적으로 PE 헤더에 기반한 정보는 비슷한 유형의 샘플에 대해서는 유사도 수치가 높지만, 변종 샘플에 대해서는 유사도 수치가 낮을 수 있다. PE 파일 식별 정보(BasicBlock, 상숫값 등)에 기반한 유사도 결과는 특정 그룹에 의해 제작된 변종 악성코드의 유사함까지 측정하기에 유사도가 상대적으로 높게 나타났다. 때문에, 유사도 시스템의 가중치를 적용한 공식 1에서는 PE 헤더에 기반한 정보를 일련의 보조 지표로 활용한다. 또한, PE 파일 정보에 기반한 유사도 분석은 기법별 신뢰성에 대한 가중치를 적용하여 유사도 수치 결과를 도출해낸다. 이러한 수치를 확인한 결과 하나의 유사도 기법을 활용한 것보다 본 시스템의 통합 유사도 기법을 적용한 것이 뛰어난 성능을 보여주는 것으로 확인 되었다.

VI. 결론

본 논문에서는 악성코드의 유사도를 분석하는 기법을 통합하여 최적의 유사도 수치를 도출하는 시스템에 대해 설명하였다. 결과적으로는 하나의 유사도 알고리즘을 통해 도출한 결과보다는 신뢰성이 적용된 통합 유사도 알고리즘을 이용한 결과물이 유사도 수치의 정확성이 높은 것으로 판단되며, 추후에는 클러스터링을 통한 모집단 형성 및 PE 파일 이외에 다양한 유형의 악성코드(문서형 악성코드 등)에 유사도 분석을 적용할 예정이다.

[ACKNOWLEDGMENT]

본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best) 지원을 받아 수행되었음.

[참고문헌]

[1] Choi Yang Seo, Kim Ik-kyun, Oh Jin-tae, and Ryu Jae-chul. (2008) PE Header Characteristics Analysis Technique for Malware Detection.

[2] Kwon Hee-joon, Kim Sun-woo. (2016) Malicious code classification system using Multi N-program.

[3] George D. Webster, Bojan Kolosnjaji, Christian von Pentz, Julian Kirsch, Zachary D. Hanif, Apostolis Zarras and Claudia Eckert. (2015) Finding the Needle: A Study of the PE32 Rich Header and Respective Malware Triage.

[4] Kim Young-Soo. (2017) Ensemble Model using Multiple Profiles for Analytical Classification of Threat Intelligence

취약점 악용 예측에 활용되는 Reference Feature의 적합성 분석 연구

양희동*, 이만희*

*한남대학교 컴퓨터공학과

A Study on Appropriateness Analysis of Reference Feature Utilized Predicting Vulnerability Exploit

Hee-Dong Yang*, Man-Hee Lee*

*Department of Computer Engineering, Hannam University.

요 약

정보통신기술과 네트워크의 비약적인 발전에 따라 보안취약점과 이를 이용한 악용 사례 또한 증가하고 있다. 취약점으로 인한 악용을 예방하기 위해 보안 관리자는 취약점을 점검 및 식별하고 패치를 수행해야 하지만 모든 취약점에 대해 예방하는 것은 보안 관리자 입장에서도 불가능에 가까운 일이다. 이에 따라, 최근에는 악용될 것 같은 취약점에 우선적으로 대응하기 위해 머신러닝 또는 딥 러닝을 활용하는 연구가 진행되고 있다. 하지만 이러한 연구들에서 대부분 Feature로서 사용하는 Reference는 정답 정보를 포함하고 있을 가능성이 존재한다. 그러므로 본 연구에서는 보다 정확한 예측 모델을 만들기 위해 이전 연구들에서 활용한 Reference의 적합성을 분석하였다.

I. 서론

최근 정보통신기술과 네트워크의 비약적인 발전으로 인해 등장하는 보안 취약점 또한 매년 증가하고 있다. Flexera 社의 2018년 Vulnerability Review report에 의하면 2017년에 발견된 취약점의 수는 19,954개로 해당 수치는 2016년에 비해 약 14% 증가한 수치라고 밝혔다[1]. KENNA Security 社의 Prioritization to Prediction 리포트에서는 취약점을 활용한 악용 사례의 절반 이상이 CVE(Common Vulnerability Exposure) 게시일로부터 2주 내에 이루어졌으며 CVE가 공개 된 달에 가장 많은 공격이 이루어졌다고 밝혔다[2][3].

취약점은 하드웨어와 소프트웨어 모두에 존재할 수 있고 여부에 따라 악용으로 이어질 수 있기 때문에 중요한 시스템일수록 취약점 존재 유무를 점검하고 패치를 진행하는 작업이 매우 중요하다. 하지만 보안 관리자 입장에서 모든 시스템의 취약점을 점검하고 식별 및 대응한다는 것은 사실상 불가능에 가깝다. 대처 방안으로 악용이 될 것 같은 취약점을 우선적으로 패치하는 방안이 있지만 어떠한 취약점이 우선적으로 악용될지 예측하는 것 또한 매우 어려운 작업이다.

취약점에 대한 악용 예측을 위한 방안으로 FIRST에서 CVE에 대한 위험도를 평가하는 시스템으로 만든 CVSS(Common Vulnerability Scoring System)를 활용할 수 있지만 모든 CVE에 대해 CVSS 점수가 산정되어 있지 않기 때문에 CVSS 점수만으로 취약점에 대한 악용을 예측하는 것은 부족하다고 판단한다[4].

이에 따라, 최근에는 머신러닝 또는 딥 러닝을 활용하여 취약점 악용 예측에 대한 다양한 연구가 시도되고 있다[2][5][6][7]. 머신러닝과 딥러닝은 정답을 통해 학습하는 지도학습과 정답 없이 학습을 진행하는 비지도 학습으로 나뉜다. 지도학습은 학습 데이터에 대한 정답을 제공하고 모델을 학습해 나가는 방식으로 학습

된 모델이 학습한적 없는 데이터에 대해 정확하게 정답을 맞추는 것을 목표로 한다. 비지도 학습은 정답이 없는 데이터만으로 학습을 하여 데이터가 어떻게 구성되었는지를 알아내어 숨겨진 특징이나 구조를 발견하는 것을 목표로 한다.

취약점의 악용 여부를 예측하기 위한 다양한 연구의 대부분이 지도학습을 기반으로 NVD의 정보와 Exploit-DB 정보를 활용하였다[6][7]. NVD는 취약점에 대한 정보를 모아놓은 대표적인 데이터베이스이며 SCAP (Security Content Automation Protocol)을 사용한 미국 정부 표준 취약점 관리 저장소로 CVE에 대한 Description, Reference, Vendor, CVSS, CWE(Common Weakness Enumeration), Published Date 등 학습 데이터의 Feature로서 사용될 수 있는 다양한 정보를 보유하고 있다[8]. 정답 레이블로서 활용될 수 있는 실제 악용 데이터를 보유하고 있던 OSVDB(Open Source Vulnerability Database)와 Symantec WINE(Worldwide Intelligence Network Environment)의 서비스 종료로 인하여 많은 연구에서 Exploit에 대한 PoC(Proof of Concept) 코드가 공개되어 있는 Exploit-DB의 데이터를 정답 레이블로서 활용하였다[9][10].

대부분의 악용 예측 연구에서는 Exploit-DB의 데이터가 PoC코드를 보유하고 있기 때문에 Exploit-DB 데이터를 정답 레이블로서 사용했지만 Exploit-DB외에도 NVD의 Reference에 공개 되어 있는 다양한 주소에서 PoC를 찾아볼 수 있는 경우가 존재한다. 이에 따라, 우리는 악용 예측을 위해 구성되는 Feature에 활용되는 Reference의 적합성 분석을 통해 보다 정확한 모델을 만드는데 기여하고자 한다.

II. 관련연구

1.1 Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits[5]

해당 연구는 머신러닝을 활용하여 취약점의 악용 여부에 대한 가능성과 얼마나 빠른 기간 내에 악용될 것인지를 예측하기 위한 연구를 수행하였다. 실제 악용 데이터를 보유하고 있는 OSVDB와 MITRE 社의 취약점 정보로부터 Feature를 추출하여 유용한 정보가 포함되어 있는 특징인 날짜, 스코어 등을 정수형으로 추출하고, 나머지 정보는 bag-of-words 기법을 사용하여 바이너리 정보로 추출하였다. 학습하기 위한 알고리즘으로는 SVM(Support vector machine)을 사용하고 데이터 셋의 구성으로는 Exploit, Unexploit 데이터에서 각각 약 4천개의 샘플을 무작위로 산출하여 데이터 셋을 10개로 분할하고 10개를 테스트 셋, 나머지 9개를 트레이닝 셋으로 사용하는 Ten-Fold 교차검증 방식을 사용하였다. 결과로는, 89.8%의 예측 정확도를 도출하고 악용될 기간에 대한 예측에는 기간대별로 75~80%의 정확도를 도출하였다.

1.2 Predicting Exploit Likelihood for Cyber Vulnerabilities with Machine Learning[6]

해당 연구에서도 위에 소개된 연구와 같이 취약점의 악용 여부에 대해 예측을 시도한 연구로 Naive Bayes, Liblinear, LibSVM linear, LibSVM RBF, Random Forest 총 5가지 머신러닝 알고리즘을 사용하여 악용 예측을 시도하였다.

해당 연구에서는 OSVDB의 서비스 종료로 인하여 정답레이블로 Exploit-DB 사이트의 데이터를 활용하였으며 취약점 정보로는 NVD의 데이터를 사용하였다. 수집한 취약점 데이터를 바로 Feature로 사용하지 않고 Exploit에 대한 가능성을 나타내는 Probability를 조건부 확률을 통해 도출하였고 Feature를 구성함으로써 78~82%의 Accuracy를 도출하였다.

1.3 Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-source Data[7]

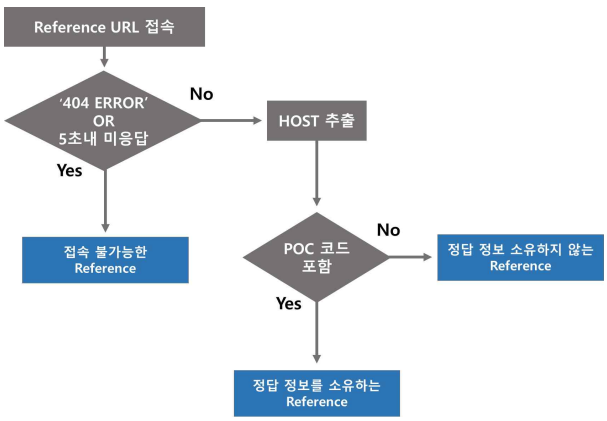
해당 연구에서는 취약점 악용 여부 예측에 대한 선행 연구들이 사용한 방법론에 대해 예측력을 부풀리는 결점을 소개하였다. 이러한 결점들이 예측 모델의 성능에 어떻게 영향을 미치는지 보여주기 위해 핵심 부분을 복제하고 접근법을 비교한다.

머신러닝을 활용하여 악용을 예측하는 다양

한 연구에서 고려되지 않은 클래스 불균형, 훈련 및 테스트 데이터 무작위 분할과 시간적 분할 방법의 차이, 사전에 악용되지 않은 취약점 활용 등의 사항들을 고려하였을 때 나타나는 정확도의 차이를 실험을 통해 보여준다. 또한, NVD의 데이터에 소셜 미디어로부터 수집한 데이터를 추가로 활용해도 악용 예측 정확도에 큰 차이가 발생하지 않음을 실험을 통해 증명하였다.

III. NVD Reference의 Feature 적합성 분석

취약점의 악용 여부를 예측하기 위해 진행된 다양한 연구들은 학습 데이터의 Feature로서 NVD의 데이터를 주로 사용한다[5][6][7]. 그 중 NVD의 Reference는 취약점과 관련된 정보가 있는 외부 사이트를 나타낸다. 하지만 Reference를 통해 접속한 외부 사이트에서 PoC 코드가 포함되어 있다면 정답 정보를 포함하고 있는 것으로 해석할 수 있기 때문에 Exploit-DB와 같이 PoC코드가 공개되어 있는 링크가 사용되어서는 안 된다. 하지만 Reference에 PoC코드가 포함되어 있는지에 대한 여부를 분석한 연구는 진행된 바가 없기 때문에 Reference에 대한 분석이 요구된다. 이에 따라, 우리는 [그림 1]과 같은 Flow를 통해 NVD에 존재하는 282,800개에 대한 Reference에 대해 PoC존재 여부를 파악하는 실험을 진행하였다.



[그림 1] Reference 적합성 분석 Flow

Reference의 Feature 적합성 여부를 분석하기 위해 2018년 7월 12일을 기준으로 등록된 104,970개의 CVE와 Reference 데이터를 수집하였다. 수집한 Reference 중 정답 레이블로 사용되는 Exploit-DB 와 자체적으로 판단했을 때 정답 정보를 포함하고 있는 Reference로 판단되는 http://www.osvdb.com, http://milw0rm.com 주소는 제외시키고 URL의 중복제거 과정을 거쳐 282,800개의 데이터를 분석대상으로 지정하였다.

파이썬 언어를 기반으로 스크립트를 제작하였으며 [그림 1] 과 같이 Reference URL을 하나씩 접속하여 PoC코드의 존재 유무를 파악하였다. 우선 접속을 하였을 때 '404 Page Error'가 발생하거나 5초 내에 응답이 없을 경우에는 접속이 불가능한 Reference로 분류하고 접속이 가능한 경우에는 페이지 내에 존재하는 Text를 모두 읽은 후 'POC', 'Proof of Concept', 'PoC', 'Exploit/POC', 'Attack Script', 'proof Of Concept' 등의 단어의 포함 여부를 확인하여 PoC코드의 존재 여부를 판단한다. 해당 패턴에 탐지되지 않는 Reference의 경우에는 정확한 분류를 위해 PoC를 코드를 갖고 있을 때 나타나는 패턴을 분석하여 분류를 진행하였다. 추가적으로, Reference의 Host 주소 별로 포함하고 있는 정답 정보의 수를 파악하기 위해 접속 가능한 Reference의 주소 전체를 편집하여 Host 주소만 추출하고 중복 제거 작업을 진행함으로써 전체 8,185개의 host주소를 추출하였다.

순서	Reference Host	Reference Count			
		정답 포함	정답 미포함	접속 불가	PoC 포함 비율
1	securityfocus.com	14562	33479	12391	24.1%
2	securityreason.com	5152	844	36	85.4%
3	packetstormsecurity.com	2329	2	0	99.9%
4	packetstormsecurity.org	1149	18	0	98.5%
5	seclists.org	362	1965	13	15.5%
6	github.com	352	7996	154	4.1%
7	mozilla.org	306	3555	12	7.9%
8	bugzilla.mozilla.org	305	2434	5	11.1%
9	marc.info	272	5569	76	4.6%
10	securitytracker.com	262	13742	30	1.9%
합 계		26,217	215,290	41,293	9.3%

[표 1] TOP 10 Reference

실험 결과 우리는 위 [표 1]과 같은 결과를 도출하였다. [표 1]은 전체 282,800개의 Reference 중 정답정보를 포함하고 있는 상위 10개의 Reference 정보를 나타낸다. 상위 10개 뿐 만이 아닌 모든 Reference가 보유하고 있는 정답 정보는 26,217개로 악용 예측의 정확한 분류를 위해서는 이러한 Reference들을 제거한 뒤 Feature로서 활용하여야 한다.

IV. 결론

본 논문에서는 취약점 악용 예측을 위해 진행된 연구들에서 사용한 NVD Reference 정보가 Feature로서의 적합성을 분석하는 연구를 수행하기 위해 각각 Reference에 PoC코드의 존재 유무를 파악하고 분류하였다. 모든 Reference 사이트에서 제공되는 취약점 정보가 정형화 되어 있지 않아 발견하지 못한 정답 정보를 포함하는 사이트가 존재할 수도 있지만, 최소 282,800개의 전체 Reference 중 26,217개의 PoC가 존재한다는 것을 실험을 통해 파악하였다. 전체 Reference 중에서 9.3% 비율로 다소적은 비율이지만 정확한 분류 모델을 만들기 위해서는 반드시 제거 되어야 한다고 판단한다. 이에 따라, 우리는 향후 연구로 적합하다고 판단되는 Reference만을 머신러닝 또는 딥 러닝을 통한 학습에 Feature로 사용하였을 때 나타나는 정확도의 차이를 분석함으로써 우리가 조사한 결과에 대한 검증 연구를 진행하고자 한다.

[참고문헌]

[1] Flexera, Vulnerability Review 2018, https://resources.flexera.com/web/pdf/Research-SVM-Vulnerability-Review-2018.pdf

[2] KEENA Security, Prioritization To Prediction Report, https://www.kennasecurity.com/prioritization-to-prediction-report/images/Prioritization_to_Prediction.pdf

[3] MITRE, Common Vulnerabilities and Exposures, https://cve.mitre.org/about/index.html

[4] FIRST, Common Vulnerability Scoring System, https://www.first.org/cvss/

[5] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits, The 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July, 2010

[6] Michel Edkrantz and A. Said , Predicting Exploit Likelihood for Cyber Vulnerabilities with Machine Learning, Unpublished Master's Thesis, Chalmers University of Technology Department of Computer Science and Engineering, Gothenburg, Sweden, 2015

[7] B. L. Bullough, A. K. Yanchenko, C. L. Smith, and J. R. Zipkin, Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-source Data, The 3rd ACM on International Workshop on Security And Privacy Analytics, 2017

[8] NVD, "General Information", https://nvd.nist.gov/general

[9] KOUNS and Jake, Open Source Vulnerability Database, The Open Source Business Resource, April, 2008

[10] Tudor Dumitras and Darren Shou, Toward a standard benchmark for computer security research: the worldwide intelligence network environment (WINE), April, 2011

Window Stations and Desktops를 활용한 보안 프로그램 활용성에 대한 고찰

여상준*

고동의*

*대구대학교 정보보호영재교육원

A consideration of utilization of security program via Window Stations and Desktops

Sang-Jun Yeo*

Dong-Eui Ko*

*Information Protection Gifted Education Center, Daegu University.

요 약

만약 공용 컴퓨터를 운영하는 관리자가 보안 대책을 허술하게 관리 한다면 어떻게 될까? 만약 이전 사용자로 의해 바이러스에 감염된 컴퓨터를 사용한 일반인은 보안 위협에 노출될 우려가 생길 것이다. 그래서 이 논문에서는 공용 컴퓨터에서도 일반인이 간편하게 다룰 수 있는 보안 대책으로 Window Stations and Desktops를 활용한 보안에 주목했다. 그래서 이 논문에서는 Window Stations and Desktops의 활용성과 문제점에 대해 알아본다.

I. 서론

본 논문에서는 공공기관에서 일반인도 간편하게 실천할 수 있는 보안 대책으로 Window Stations and Desktops를 활용한 보안 프로그램에 대해 알아가고자 한다. 공공기관 컴퓨터 관리자는 사용자에 의해 공공 컴퓨터가 컴퓨터 바이러스에 감염되었을 경우를 위해 1)스냅샷(Snapshot)이나 가상머신(Virtual machine)을 통해 컴퓨터 바이러스에 감염 안 된 상태로 유지 할 수 있다. 그러나 만약 관리자가 이러한 환경을 구축하지 않은 공용 컴퓨터일 때 일반 사용자는 취약한 보안 환경에 노출되어 누군가 악의적으로 설치한 2)Hooking 프로그램과 같은 컴퓨터 바이러스 의해 개인정보가 노출될 우려가 있다. 그래서 이러한 공공 컴퓨터에서 기존 시스템의 환경을 유지하면서도 보안 위협으로부터 지킬 방법을 모색하여 Window Stations and Desktops를 활용한 보안 대책을 연구하게 되었다.

1) 빠르고 효율적인 백업 기술로 가상 머신, 볼륨 관리자, 데이터베이스 등 다양한 분야에 널리 이용되고 있다.

2) 사용자의 입력 값을 가로채는 기술이다.

II. 관련 연구

2.1 Window Stations and Desktops 배경지식

MSDN에 있는 내용을 따르면 Windows OS는 크게 3)UI, 4)GDI, Kernel 총 3가지의 구성요소를 지원한다. Kernel에서 생성된 Kernel object는 Windows OS에서 보호할 수 있으나 UI의 User object와 GDI는 보호하지 않는다. 그래서 Windows OS에서 추가 보안을 제공할 수 있도록 Kernel의 보안 object인 Window Stations and Desktops를 사용하여 User object를 관리하도록 한다.

3) User Interface의 약자

4) Graphic Device Interface의 약자

Window Stations and Desktops로 생성된 새로운 바탕화면은 Kernel에서 생성된 Kernel object이다. 즉, 이전에 있던 바탕화면에 사용자의 키보드 입력 값을 전달하지 않고 새로 만든 바탕화면에 사용자의 키보드 입력 값을 전달하는 것이 가능하다. 그래서 이전에 있던 바탕화면에 5)Keylogger와 같은 악의적인 프로그램이 실행되고 있더라도 악성 프로그램에 사용자의 키보드 입력 값이 전달되지 않는다.

2.2 Window Stations and Desktops 활용방법

이 문단에서는 Window Stations and Desktops를 활용할 수 있는 3가지 방법을 프로그램으로 만들어본 결과에 대해 다루어 본다. Window Stations and Desktops를 구현하기 위한 기본적인 코드는 MSDN과 Code Project 사이트를 참조하면 쉽게 만들 수 있다. 단지 서로 다른 점이라면 가상 화면에 원하는 프로그램을 실행하는 방법이 서로 다를 뿐이다.

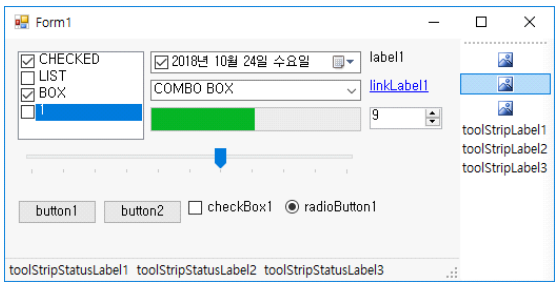
2.2.1 Windows Forms 기반을 활용하기(6)MTA)

새로운 화면에 원하는 프로그램을 실행하기 위해 [코드 1]과 같이 Thread를 활용하였다.

```
IntPtr 0 = IntPtr.Zero,
    hOldDesktop =
    GetThreadDesktop(GetCurrentThreadId()),
    hNewDesktop =
    CreateDesktop("Name", 0, 0, 0, 0x1FF, 0);

SwitchDesktop(hNewDesktop);

var thread = new Thread(() => {
    SetThreadDesktop(hNewDesktop);
    Application.Run(new Form1());
});
thread.Start();
thread.Join();  //프로그램 종료 대기

SwitchDesktop(hOldDesktop);
CloseDesktop(hNewDesktop);
```

[코드 1] MTA 환경에서 실행되는 실행코드

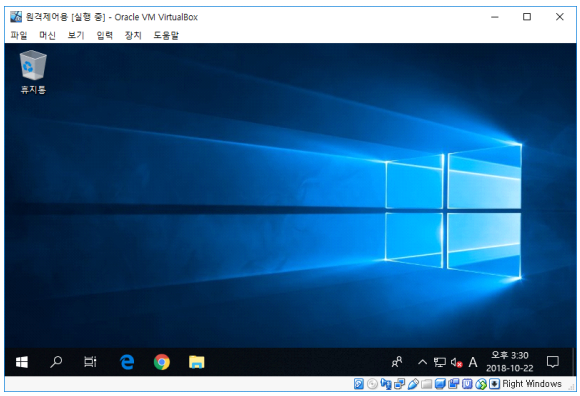
C#은 기본적으로 MTA 환경에서 동작한다. 그래서 [코드 1]에서 정상적으로 Form을 구현하기 위해 [그림 1]과 같이 MTA 환경에서도 동작 가능한 COM 객체만으로 구성한 Form을 구성해 보았다.



[그림 1] MTA에서도 동작하는 COM 객체로 구성되었다.

[코드 1]과 같은 코드로 새로 만든 바탕화면에 [그림 1]의 Form이 정상적으로 실행된 것을 확인할 수 있다. 하지만 WPF 및 UWP 프로그램 또는 7)STA만 지원하는 COM 객체를 사용한다면 다른 상황이 발생한다. 그래서 다음과 같이 두 가지의 활용 방법을 예로 시험해 보았다.

2.2.2 원격제어를 활용하기(STA)



[그림 2] 가상머신으로 구현한 RDP Server

원격제어를 위해 가상 머신으로 RDP Server를 구현하고, Microsoft에서 제공하는 8)RDP 컨트롤러를 이용하여 RDP Client를 구현하였다. 프로그램은 2.2.1처럼 기본적으로 MTA 환경에서 실행되도록 코딩하였다.

[그림 3] 실행 시 STA로 설정하지 않아 발생한 오류

5) keystroke logging의 약자로 사용자의 키보드 입력을 기록하는 Hooking 프로그램의 일종이다.
6) Multi Thread Apartment
7) Single Thread Apartment
8) Remote Desktop Protocol

그러나 RDP Client가 STA만 지원하는 객체이기에 [그림 3]과 같이 스레드 오류와 함께 정상 동작하지 않는다. 이처럼 STA 동작 환경에서만 초기화할 수 있는 웹 브라우저, 드래그 앤드 드롭, 클립보드 관련 등……. 이와 같은 객체를 사용할 때는 [코드 1]에서 스레드 상태를 STA로 설정한 [코드 2]와 같은 형태로 코드를 작성한다.

```
....
var thread = new Thread(() => {
    ....
});
thread.SetApartmentState(ApartmentState.STA);
thread.Start();
thread.Join();
....
```

[코드 2] STA 환경에서 실행되는 실행코드

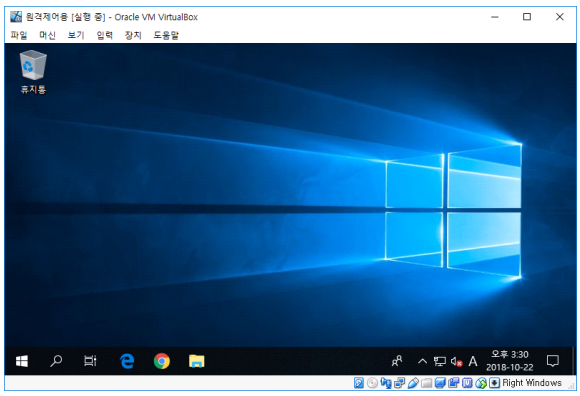
[그림 4] 새로 만든 바탕화면의 메모리 오류

하지만 RDP가 연결을 시도하는 순간 새로 만든 바탕화면에 많은 양의 스택이 쌓이게 되어 메모리 부족 현상이 발생한다. 그래서 이처럼 용량이 큰 프로그램이 실행될 경우 새로운 바탕화면을 구성할 때 최소한의 메모리 용량으로 바탕화면을 생성하고 [코드3]과 같이 [9] CreateProcess 함수를 통해 새로 만든 바탕화면 위에 별도의 프로그램을 실행시켜야 한다.

```
....
var si = new STARTUPINFO();
    si.cb = Marshal.SizeOf(si);
    si.lpDesktop = "바탕화면 이름" ;

var pi = new PROCESS_INFORMATION();
    pi.hThread = NewDesktop;

IntPtr 0 = IntPtr.Zero;

CreateProcess(
    null, "경로" , 0, 0, true, 0x20, 0,
    null, ref si, ref pi);
....
```

[코드 3] 사용된 CreateProcess 코드 구조

9) Process.Start() 메서드와 유사하며 원하는 바탕화면의 실행 정보를 추가로 설정하여 새로 만든 바탕화면에 프로그램을 실행할 수 있다.



[그림 5] 정상 동작하는 RDP Client 프로그램

개별적인 프로세스로 실행한 결과 문제없이 잘 동작하는 것을 확인할 수 있었다.

2.2.3 WPF를 활용하기

WPF에서도 2.2.2와 같이 STA 환경으로 실행해 본다. 새로운 바탕화면에 프로그램이 띄워지지만, 렌더링이 되지 않아 흰색 화면으로 보이며 각각의 구성요소는 성능이 크게 떨어지는 문제점이 발생한다. 반면에 CreateProcess를 통해 새로운 바탕화면에 개별적인 프로세스로 구현하면 잘 동작한다.

[그림 6] WPF 기반 기본 프로그램 실행 문제점

그러나 Windows 8 이후 사진, 동영상, 음악과 같이 WPF를 기반으로 하는 응용프로그램이 기본 프로그램으로 등록돼 있다. 그래서 파일을 열 때 [코드 3]과 같이 실행하지 못하여 이러한 기본 프로그램이 기존 바탕화면에 실행되는 문제점이 있다.

2.2.4 보안 실험 결과

키로거와 모니터링 프로그램을 사용하여 보안 실험을 해본 결과 사용자의 개인정보를 보호할 수 있다는 것을 알 수 있었다. 하지만 일부 보안 프로그램 간의 충돌이 있어 비정상적으로 바탕화면이 구성되는 문제점이 있다. 또한 네트워크나 파일 등……. 은 보호하지 않기에 추가적인 보안이 없다면 사용자의 작업 환경 보안이 취약해진다.

III.본론

3.1 활용 가능성

본 논문에서 Window Stations and Desktops에 대한 활용 가능성에 대해 알아보기 위해 여러 방법에 대해 시험해 보았다. 그 결과 WPF에서는 일부 문제점이 있었지만, Windows Forms 기반은 안정적인 모습을 보여주었다. 그래서 Windows Forms 기반으로 Window Stations and Desktops를 활용한다면, 공공 컴퓨터에서 작업하더라도 안정적인 동작을 제공 할 수 있다는 점에서 활용성과 발전 가능성이 있을 것으로 기대된다. 하지만 이러한 발전 가능성을 앞에 두고 다음과 같은 활용성에 대한 문제점을 해결해야 활용 범위가 좋아질 것이다.

3.2 활용성에 대한 문제점

3.2.1 WPF에 대한 문제점

WPF 기반의 기본 프로그램을 새로 만든 바탕화면에서 실행할 경우 기존에 있던 바탕화면에서 실행된다. 심지어 Windows 8 이상부터는 기본적으로 WPF 기반의 기본 프로그램이 제공된다. 이러한 문제점은 최신 버전의 Windows에서 안정적으로 실행하지 못해 향후 상용화까지의 활용성을 가로막는 가장 큰 요인이 될 것이다.

3.2.2 Anti-virus 프로그램의 충돌

일부 Anti-virus 프로그램이 새로운 바탕화면을 만들 때 수상한 프로그램으로 오진하여 프로그램의 동작을 차단한다. 이를 해결하기 위해 보안 회사 간의 협력을 통해 예외 처리로 해두거나 비정상적으로 동작 될 때 정상 화면으로 되돌리는 기능을 만들어야 할 것이다.

3.2.3 보안 취약점

Window Stations and Desktops는 process의 보안과 관련되어 있지만 실행되는 프로그램이 악성 코드인지 아닌지 판별할 수 없다. 즉, 외부 또는 내부에서 추가적인 바이러스 감염에 대한 보안 대응이 없다. 그러므로 별도의 Anti-virus 프로그램이 없다면 보안 취약점으로부터 사용자의 작업 환경을 보호하기에는 한계점이 있다.

IV.결론

본 논문에서는 Window Stations and Desktops를 활용한 보안 프로그램 활용성에 대한 고찰을 위해 여러 개발 상황에 대해 시험을 해 보았다. Window Stations and Desktops를 활용하면 스냅샷(Snapshot)과 가상머신(Virtual machine)에 비해 기존에 있던 사용자의 정보를 지우지 않고도 사용자를 바이러스로부터 보호할 수 있다. 그리고 별도의 복구 시간과 재부팅 없이 실행과 동시에 기존에 작업하던 상태를 유지해둔 채로 원하는 작업을 안전하게 수행할 수 있다. 또한 WPF에 비해 안정적인 동작을 보여준 Windows Forms를 기반으로 Window Stations and Desktops 환경에 맞추어 설계한다면 활용하기 좋을 것이다. 그렇기에 Window Stations and Desktops를 보안 프로그램에도 고려해봐야 한다고 생각한다.

V.향후 연구과제

Window 8 이상부터 WPF 기반 기본 응용 프로그램이 사용된다. 하지만 WPF 기반의 기본 응용 프로그램을 실행할 경우 새로 만들어진 바탕화면이 아닌 기존 바탕화면에 표시된다. 그래서 WPF의 정상적인 동작에 관한 연구를 계속해 나갈 것이다.

[참고문헌]

[1] Window Stations and Desktops

https://docs.microsoft.com/en-us/windows/desktop/winstation/about-window-stations-and-desktops

[2] STA와 MTA에 대한 참고 문헌

https://www.slideshare.net/topcredu/c-sta-mta-cwpf

[3] CreateProcess 함수

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createprocessa

[4] 코드 참고 사이트

https://www.codeproject.com/Articles/677695/Csharp-Create-Secure-Desktop-Anti-Keylogger

MITM 공격을 이용한 IoT Hub 환경의 취약점 연구

김지섭*, 김동영*, 김학진*, 방수정*, 조진호*, 최강현*, 임홍렬*, 박의성*, 지한별*, 이종호**, 이기택**

*KITRI 차세대 보안리더 양성 프로그램 (Best of the Best)

**고려대학교 정보보호 대학원

Title

Ji-Sub Kim*, Dong-Young Kim*, Hak-Jin Kim*, Su-Jung Bang*, Jin-Ho Jo*, Kang-Hyun Choi*, Hong-Ryeol Lim*, Ui-Sung Park*, Han-Byeol Ji*, Jong-Ho Lee**, Ki-Taek Lee**

*Best of the Best, Korea Information Technology Research Institute(KITRI)

**Graduate School of Information Security, Korea University

요 약

IoT Hub는 IoT 기기들을 연결하는 중추적인 역할을 수행함으로써 사용자에게 편의성을 제공한다. 최근 IoT Hub의 취약점이 국내·외에서 지속적으로 발표되고 있으며 이러한 취약점들은 사용자의 안전과 자산에 심각한 문제를 초래할 수 있다. 따라서 본 논문에서는 IoT Hub의 심각한 위험 중 하나인 MITM (Man-in-the-Middle) 공격을 이용하여 IoT Hub 환경의 취약점을 분석하여 침해가 발생했을 때의 위험성을 증명하고, 이에 대한 해결방안을 제시한다.

I. 서론

4차 산업혁명 시대를 맞아 사물 인터넷(Internet of Things:이하 IoT)에 대한 관심이 높아지고, 이를 구현할 수 있는 제품들이 다양하게 선보이고 있다. 이중 IoT Hub는 기존의 통신 게이트웨이에 핵심 기능을 접목 시켜 IoT 기기들을 연결하는 중추적인 역할을 수행한다. IoT Hub의 도입에 따라 사용자의 편의성은 확대되었지만 현재 IoT Hub 서비스의 보안성은 미흡하다. 본 논문에서는 현재 IoT Hub에서 심각한 위험이 되고 있는 MITM 현황을 분석하여, 발생 가능한 위협요소에 대하여 살펴보고 이에 대한 대응방법을 기술하였다.

II. 관련 연구

2.1 IoT Hub 보안 관련 연구

IoT Hub에 대한 취약점에 대한 연구는 DEFCON, ReConm HITCON, Codegate와 같은 국내·외 컨퍼런스에서 지속적으로 발표되고 있다. 예를 들어 Dennis Giese는 DEFCON, ReCon, HITCONE 등의 컨퍼런스에서 임의의 클라우드 서버를 만들어서 MITM이 공격이 가능하다는 것을 발표하였다. 또한 컨퍼런스 이외에도 Fernandes는 삼성 SmartThings를 대상으로 IoT Hub에 설치되는 여러 IoT 기기에 대한 공격 시나리오를 작성하고 이에 대한 Exploit Platform을 구현하여 제보하였다. 이처럼 IoT Hub를 공격할 수 있는 방법은 다수 알려져 있다. 본 논문에서는 알려진 IoT Hub 공격 방법 중 MITM 공격을 사용하여 IoT Hub에 설치되는 여러 IoT 기기에 대한 명령이 변조가 가능했을 때의 위험성을 증명한다.

2.2 IoT Hub 서비스 구조

IoT Hub 서비스는 사용자가 사용하는 IoT 기기들을 한 번에 원격으로 관리할 수 있도록 하는 서비스이다. 그림 1은 IoT Hub의 서비스 구성이다.

사용자의 스마트폰, 태블릿과 같은 스마트 기기를 통해 기기를 제어하는 명령을 서버를 거쳐 각 AP에 연결된 IoT Hub로 전송하고 이 명령은 IoT Hub를 통해 Z-Wave, Zigbee등과 같은 근거리 저전력 무선 프로토콜을 이용하여 통신하며 서버로부터 오는 명령을 실행하는 구조로 되어 있다.

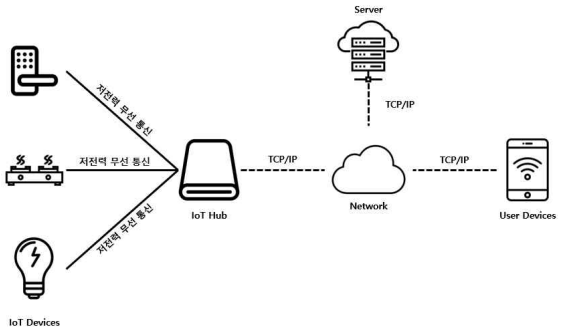


그림 1 IoT Hub Service Concept

2.3 IoT Hub와 서버의 통신 과정 분석

본 절에서는 IoT Hub와 서버와 통신하는 과정에 오가는 패킷을 분석한다. 통신하는 과정에서 TLS/SSL을 적용하지 않은 일반적인 TCP나 UDP 통신이라면 쉽게 MITM 공격이 가능하다. 이를 막기 위해 인증서에 Pinning을 적용하는 경우가 있지만 이러한 경우에도 기기에 대한 쉘 획득 후 해당 펌웨어에서 사용하고 있는 인증서를 교체하여 우회할 수 있다. 이 때 캡처한 패킷을 분석하여 통신 과정의 취약점에 대해 분석할 수 있다.

```
{ size: 40,
  protocolVersion: 17,
  headerType: 17,
  headerLen: 37,
  msgType: 2,
  msgPattern: 2,
  methodType: 525,
  transactionId: 1539692277836,
  authToken: '00000000-3ba2-3a87-0000-000000000001',
  [redacted]
  resultCode: 0,
  node: 6,
  body: <Buffer 25 01 ff> }
```

그림 2 분석한 패킷의 데이터

그림3의 패킷 수집 결과를 통해 서버와 IoT Hub 사이의 패킷은 자체적으로 사용되는 프로토콜을 통해 통신을 하는 것을 확인할 수 있다. 여기서 msgType은 해당 패킷이 요청에 대한 패킷인지 응답에 대한 패킷인지에 대한 옵션이다. 또한 서버는 IoT 기기들을 제어하기 위해 node 영역과 body 영역을 사용한다. node영역에는 제어할 node에 대한 정보가 들어 있고, data 영역에는 제어할 기기의 대한 명령을 정의하고 그 명령에 대한 값을 담아 보낸다. 그림 4는 IoT Hub 내에서 IoT 기기 내에 명령 보낼 수 있는 목록들을 분석한 결과이다.

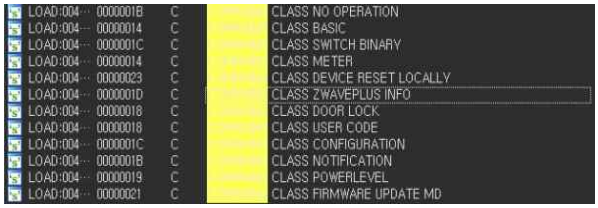


그림 3 IoT Hub내에서 보낼 수 있는 명령

III. 실험

본 장에서는 국내·외에서 쓰이고 있는 IoT Hub 제품 5개를 대상으로 진행한 MITM 공격을 통한 취약점 분석에 대해 기술한다.

3.1 MITM 시나리오

본 논문에서는 서버와 IoT Hub가 통신하는 단계에서 MITM 공격을 수행한다. 그림 2는 본 논문에서 수행하는 실험에 대한 흐름이다.

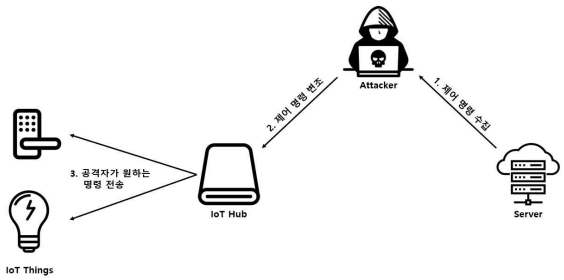


그림 4 MITM 공격 시나리오

우선 공격자의 목적은 MITM 공격을 통해 IoT Hub가 관리하는 IoT 기기들을 조작하는 것이다. 이를 위해 공격자는 서버와 IoT Hub 사이에서 전송되는 기기 제어 명령을 가로채어 공격자가 원하는 스마트 기기의 제어 명령으로 변조해서 IoT Hub로 전송한다. 변조된 명령을 받은 IoT Hub는 서버가 전송한 것으로 착각하고 스마트 기기를 공격자가

원하는 임의의 행위로 조작한다.

3.2 IoT Hub와 서버의 통신과정 도청

IoT Hub와 서버와의 통신하는 패킷을 중간에서 도청하기 위해서 실제 서버와 같은 방식으로 동작하는 Fake 서버를 제작하고, IoT 기기가 접속하는 서버를 속이기 위해 DNS Spoofing을 통하여 서버의 IP를 공격자 서버의 IP로 속여서 접속하게 한다. 만약 펌웨어 DNS에 IP가 Static으로 되어 있을 경우 라우팅 테이블을 조작하거나 해당 기기에 대한 쉘을 획득하였다면 공격자는 iptables등을 이용하여 IoT Hub와 서버의 통신 과정을 도청할 수 있다.

3.3 패킷 유효성 검증 우회

본 절에서는 3.2에서 제시한 방법으로 패킷을 도청하여 2.3에서 분석한 패킷 구조를 통해 IoT Hub에서는 패킷 데이터의 유효성 검사를 transactionId 값으로 처리하고 있음을 확인했다. 이때의 transactionId 값은 서버의 Timestamp 값이다. 그림5는 공격자가 원하는 대로 조작하여 보내는 패킷이다.

```
{ protocolVersion: 17,
  headerType: 17,
  headerLen: 37,
  msgType: 1,
  msgPattern: 3,
  methodType: 525,
  [redacted]
  transactionId: 1540278356261,
  encodingType: 1,
  node: 8,
  body: <Buffer 25 01 00> }
```

그림 5 공격자가 의도한대로 변조된 패킷

IoT Hub에게 공격자가 원하는 IoT 기기 제어에 대한 요청을 보내기위해 msgType 옵션을 1로 설정을 하고, transactionId 값은 서버의 NTP 서버와 동일한 시간대로 설정 한 후 원하는 제어 명령을 body에 추가해서 보내면 그에 알맞은 응답 패킷을 보내 패킷에 대한 유효성 검증을 우회하고 성공적으로 MITM 공격이 가능함을 확인했다. 그림6은 Fake 서버를 만들어서 패킷을 변조하여 보냈을 때 IoT Hub가 공격자에게 보내주는 응답 패킷에 대한 데이터이다.

```
{ packetSize: 40,
  protocolVersion: 17,
  headerType: 17,
  headerLen: 37,
  msgType: 2,
  msgPattern: 2,
  methodType: 525,
  transactionId: 1540278356261,
  [redacted]
  encodingType: 1,
  resultCode: 0,
  node: 8,
  body: <Buffer 25 01 00> }
```

그림 6 변조된 패킷에 대한 응답

msgType 옵션이 2라는 것을 봐서 기기제어에 대한 응답이라는 것을 확인하였고, body에 현재 기기에 대한 상태를 보내기 때문에 공격자가 의도한대로 현재 기기가 동작했다는 것을 확인할 수 있었다.

3.4 MITM 공격 실험 결과

본 실험에서는 MITM 공격을 사용하여 국내·외에서 쓰이고 있는 IoT Hub 5개의 기기에 대하여 취약점 분석을 진행하였다. 표 1은 본 논문에서 진행한 MITM 공격에 대한 결과를 정리한 표이다.

구분	제품 A	제품 B	제품 C	제품 D	제품 E
일반적인 MITM 공격	X	O	X	X	X
SSL 인증서 Pinning 여부	O	X	O	O	O
기기에 대한 Root 쉘 획득 후 MITM 공격	O	O	O	O	O

표 1 제조사 별 IoT Hub MITM 공격 결과

Fake 서버를 구축하여 통신하는 중간에서 MITM을 시도하였을 때 대부분이 SSL 인증서가 Pinning이 되어 있어 IoT Hub가 연결을 끊어버렸기 때문에 실패했다. 하지만 펌웨어 내의 루트 권한의 쉘을 획득해서 인증서를 공격자가 임의로 바꿔주면 우회가 가능하여 MITM 공격이 가능함을 확인했다.

IV. 대응 방안

본 장에서는 본 논문에서 제시한 MITM 공격을 막기 위한 대응방안을 하드웨어 보안과 통신구간 보호 관점에서 나누어 제시한다.

4.1 조작방지 솔루션 도입

일반적으로 Root 쉘에 접근하기 위해서는 펌웨어 내에 직접적인 접근을 해야 가능하며, 디바이스 내부 접근하여 비인가 조작을 보호하려면 조작 방지 솔루션을 도입하고, 비인가 조작 탐지 시 주요 키 및 민감 데이터의 초기화를 수행하도록 해야 하며, 초기화가 어려울 경우, 제품이 비활성화 되도록 구현해야한다.

4.2 통신구간 보호

본 논문에서 점검한 IoT Hub 제품은 5개 모두 SSL/TLS 적용을 했다. 하지만 SSL 적용을 한 경우에도 공격자가 SSL Strip 등의 기법을 통해 충분히 MITM이 가능했으므로 IoT Hub와 서버 간 통신 시 중요한 값에 대해 암호화 또는 마스킹 처리된 값을 사용해야 한다. 또한 서버와 IoT Hub 모두 SSL 인증서를 검증하여 신뢰할 수 있는 경우에만 통신을 할 수 있도록 구현하여야 한다.

V. 결론

본 논문에서는 IoT Hub와 서버가 통신을 하는 과정에서 어떤 보안 수단이 있는지 확인하고, 그에 대한 우회 방법에 대해 논하였다. 해커가 만들 수 있는 시나리오를 작성하고 실제 국내·외에서 쓰이고 있는 IoT Hub 5개의 기기에 실제 테스트를 수행함으로써 대부분의 IoT Hub가 취약점을 가지고 있다는 점, 그리고 그에 대한 대응방안을 설계함으로써 IoT Hub의 SSL 인증서 Pinning 여부가 얼마나 중요한지 제시하였다는 데에 의의가 있다. IoT Hub 서비스 제공사들은 사내 모의 해킹을 통해 이러한 내용들을 점검해 보고, 추후에 유사한 공격을 방지하고 안전하게 서비스를 제공할 수 있도록 해야 한다.

[참고문헌]

[1] LG Blog, "[메가트렌드로 보는 미래] #2. 스마트 홈의 중추 역할, 홈 허브(Home Hub)가 진화하고 있다, 2017, [Online; accessed 14-Oct-2018].

[2] Cisco, "Vulnerability Spotlight: Multiple Vulnerabilities in Samsung Samsung SmartThings Hub 2018, [Online; accessed 16-Oct-2018].

[3] Symantec, "와일드카드 및 SAN 다용도 SSL 인증서의 이해", 2018

[4] Dennis Geise, "Having fun with IoT: Reverse Engineering and Hacking of Xiaomi IoT Devices", DEFCON 26, 2018

전송장비로 활용 되고 있는 백본의 Router의 기능을 이용하여 업무망 PC 상단의 백본에서 해당 IP로 연결 되지 않도록 (Fig 10)과 같이 설정 하였다. Access List Rule을 이용하여 전송되어지는 불필요 트래픽을 목적지 주소별로 차단하여 적용하였다. 설정방법은 (Fig 11)와 같다.

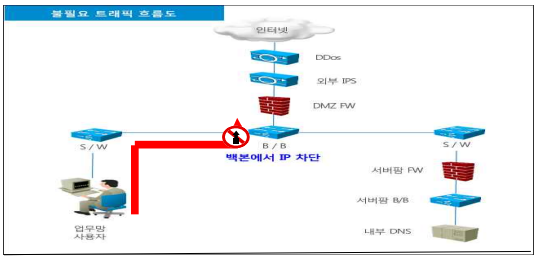


Fig. 9. Unnecessary traffic flow by IP

```
================네트워크 스위치 Access list 적용 =======================

지속적으로 외부와 연결하려는 패킷을 Access List 정책으로 차단

  ip access-list 140
  statistics per-entry
  10 deny(deny 차단) ip(IP를 확인함) any(모든 출발지 IP) 1.224.180.0/24(목적지 대역)
  20 deny ip any 1.224.181.0/24
  30 deny ip any 175.126.163.0/24
  40 deny ip any 222.239.118.0/24
  50 deny ip any 104.75.42.31/32
  60 deny ip any 104.76.88.54/32
  70 deny ip any 211.36.85.142/32
  80 deny ip any 204.79.197.200/32
  90 deny ip any 13.107.4.50/32
  100 deny ip any 13.107.21.200/32
  110 permit ip any any(맨마지막에는 반드시 허용정책을 넣어줘야 함)
  ==================================================================
```

Fig. 10. Application of network switch access list

```
Access List 적용 방법
interface Vlan6
  description ## UP-L7 ##
  no shutdown
  ip access-group 140 in (해당 Vlan의 들어오는 패킷을 제한함)
  ip access-group 140 out 해당 Vlan의 나가는 패킷을 제한함)
  no ip redirects
  ip address xx.xx.xx.xx/29
  hsrp version 2
  hsrp 6
    priority 120
    ip xx.xx.xx.xx
```

Fig. 11. Network switch Access list method

V. 불필요 트래픽 제거 실증 결과

5.1 실증 네트워크 구성

불필요 트래픽 제거를 하기 위하여 실증 네트워크를 선정하여 상기의 정책방안을 적용하여 실제적인 불필요 트래픽의 차단정도를 트래픽 감소율로 파악하였다.

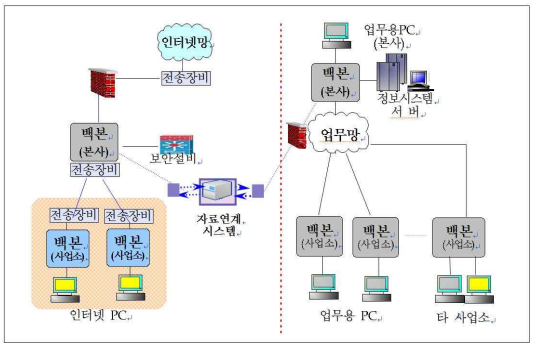


Fig. 12. Test network simplicity for demonstrating unnecessary traffic removal

실증의 대상이 되는 네트워크는(Fig. 12)와 같이 본 연구를 위해 실제 사이트에서 운영중인 업무망 전용 침입차단시스템(firewall)을 선정하여 통계데이터로 활용하였다. 침입차단시스템 차단로그는 2018년 5월 1일부터 2018년 5월 30일 까지 1개월 간 약 2,500명이 사용하는 업무용 PC에서 외부로 접근 요청이 차단된 침입차단시스템 차단 데이터를 기준으로 하였다. 실증 방법은 첫째 업무전산망 PC에 설정된 NetBIOS설정 해제, 둘째 방화벽(firewall)의 라이프타임(세션정보관리) 설정값 변경, 셋째 업무전산망 PC의 hosts파일을 이용한 불필요 URL 트래픽 조치, 넷째 업무전산망 네트워크(백본)에서 불필요 트래픽 유발되는 IP차단 방법을 4개월에 걸쳐 단계적으로 적용 했을 때의 트래픽 감소율로 비교 분석하였다.

5.2 실증 네트워크에 대한 적용 결과

실증 네트워크에서 불필요 트래픽 분석 제거를 위해 앞서 제시한 방법들을 실제 적용하여 그 효과를 도출하였다.

첫째, 업무망 PC에 설정된 NetBIOS설정 해제 적용 후 결과는 랜섬웨어 SMB(Server Message Black)확산의 취약 프로토콜인 137port를 본사 사용자 대상으로 311대중 프린터 및 서버를 제외한 278대를 NetBIOS 제거하는 스크립트를 NAC를 통해 자동 배포하여 문제가 발생되지 않아 전사 배포

를 2018. 6. 19에 하였다. 그 결과는 1억9천 6백만 건에서 1억5천9백만 건으로 18.68% 감소 효과가 있었다. 통계를 간편하게 하기 위해 월단위로 건수를 취합하였다.

둘째, 방화벽의 라이프타임 변경 정책 적용 후 결과는 세션 종료 위한 FIN 또는 Reset을 정해진 시간 내에 수신 받지 못하여 세션정보가 없어서 방화벽에서 차단되기 때문에 방화벽 라이프 타임을 1차, 2차에 걸쳐 시간을 300초로 조정하고 그 결과 1억5천9백만 건에서 1억 4천6백만 건으로 전월대비 7.87% 감소 효과를 거두었다.

셋째, 불필요 URL Localhost 화일 설정 변경 정책 적용 후 결과는 불필요 트래픽 제거 방법중 가장 좋은 효과를 거두었으며 앞에서 제시한 각종 운영체제, Adobe, 한글과컴퓨터, 백신 등등 업데이트 URL을 hosts화일로 불필요 트래픽 발생요인을 제거함으로서 1억 4천6백만 건에서 6천 9백만 건으로 전월대비 52.64% 제거율의 결과를 얻을 수 있었다.

넷째, URL이 아닌 IP로 직접 발생되는 불필요 트래픽 제거 정책 불필요 URL을 제거 하고도 지속적으로 방화벽 차단 건수가 발생하여 차단로그를 분석해보니 IP 10개 직접차단 되었고 백본에서 Routing기능을 이용한 Access List에서 차단정책으로 제거함으로써 6천 9백만 건에서 3천 2백 만건으로 전월대비 53.68% 제거율의 결과를 얻을 수 있었다.

5.3 실증 네트워크에 대한 트래픽 제거 검증

실증 네트워크에서 불필요 트래픽 제거에 대한 4가지 방법을 통하여 업무망에서 인터넷망으로 발생하는 트래픽 제거율이 83.57%로 불필요 트래픽이 대폭 감소하였다. 특히 사용자 PC에서 불필요 트래픽을 유발하는 URL을 hosts파일 설정에의한 PC단 localhost로 리다이렉션 시킨 방법과 백본에서 특정 IP를 차단시킨 방법이 획기적인 감소효과가 있었다.

Table 4. Results after removing unnecessary traffic

구분	방화벽 차단로그 (건)	제거방법	감소율 (%)	적용일자
5월1일~31일	196,200,318	제거 前	제거 前	제거 前
6월1일~30일	159,533,717	NetBIOS설정해제	18.68	2018.6.19
7월1일~31일	146,965,433	방화벽 라이프타임 조정	7.87	2018.6.20
8월1일~31일	69,595,434	hosts 파일적용	52.64	2018.8.01
9월1일~30일	32,233,978	백본 Access list적용	53.68	2018.8.31

Ⅲ장에서 연구 분석한 자료중 윈도우 업데이트, Internet Explorer 업데이트, Adobe 제품군 업데이트, 알툴즈 제품군 업데이트, Ahnlab 제품군 업데이트 설정 해제 등은 개인 업무용 PC에서 일일이 설정변경 하기에는 어려운 상황이다. 연구결과에서 조사된 불필요 트래픽을 유발하는 878개 URL을 이용하면 망분리된 모든 기업에서는 이를 참조하여 불필요 트래픽을 제거하는데 많은 도움이 될 것이다. 또한 선행연구 사례 2.2에서 매일 탐지되는 트래픽의 분포가 $\mu \pm 2\sigma$ 기준으로 관리상한과 관리하한 내에 데이터가 포함됨을 확인하여 발생하는 보안로그 데이터가 큰 변화를 보이지 않고 있음을 알 수 있다고 하였고, 업무와 무관하게 발생하는 트래픽에 대한 발생원인을 제거하여 임계치를 낮추면 관제방법에 더욱 유용성이 있다고 하였다. 본 연구논문은 불필요 트래픽의 임계치 상하한 값을 (그림 43)과 같이 84%정도 낮추는 효과가 있었다.

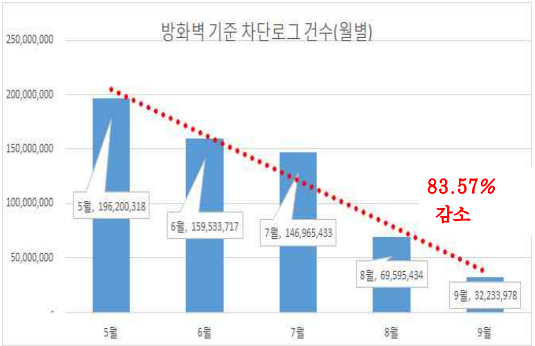


Fig. 13. Reduced amount verification result after eliminating unnecessary traffic

또한 방화벽의 차단로그가 상당히 줄었고, 이로 인한 보안관제 경보 감소로 관제인원의 분석 업무에 소요되는 시간을 효과적인 관제 업무에 투자함으로써 관제의 생산성이 향상 될 것이라고 생각한다.

VI. 결론

6월부터 9월까지 4개월간 연구결과 불필요 트래픽을 유발하는 URL이 878개에 달하였으며 그중 운영체제 관련 URL이 가장 많았고 그 종류로는 윈도우 업데이트, 아마존 클라우드, 스마트스크린, MS원격 데이터관리 등이 있다. 다음으로는 백신관련 URL이다. 백신은 윈도우에서 제공하는 백신과 상용 백신으로 구분할 수 있으며 그 다음으로는 MS오피스 관련 URL이고 구글 등 검색엔진 업데이트 관련 URL 기타로는 알툴즈 업데이트, 플래시 업데이트, PDF 업데이트, SSL인증서 업데이트, 오토캐드 업데이트, 외부 NTP서버, IP확인요청 등 이다. 또한 업무망에서 불필요 트래픽을 유발하는 IP로는 1.224.180.0/24,1.224.181.0/24, 175.126.163.0/24,222.239.118.0/24, 104.75.42.31, 104.76.88.54, 211.36.85.142, 204.79.197.200, 13.107.4.50, 13.107.21.200으로 이 또한 유해차단시스템에서 IP를 확인해 본 결과 대부분이 MS관련 Update IP였다. 앞에서 제시한 망분리 기관 및 기업체 내에서 망분리 전·후에 도입된 내부 정보시스템 자산들이 인터넷 연결환경(OS 업데이트, 보안패치 등)에 최적화 되어 내부 업무망에서 인터넷 연결을 지속적으로 시도한다.

이런 여러 형태의 인터넷 연결을 시도하는 불필요 네트워크 트래픽을 발생하는 원인을 분석해보고, 제거하는 기법을 네가지 실험적 방법을 제시하였다. 첫째 업무전산망 PC에 설정된 NetBIOS설정 해제, 둘째 방화벽(firewall)의 라이프타임(세션정보관리) 설정값 변경, 셋째 업무전산망 PC의 hosts파일을 이용한 불필요 트래픽 조치, 넷째 업무전산망 네트워크(백본)에서 불필요 트래픽 유발되는 IP차단 방법을 제시함으로써 현재 국가기관 망 분리 구축 가이드에서는 인터넷 PC의 최소 지급등 인터넷PC의 보안강화에 대한 기준만을 제시 하였으나, 이 논문에서는 내부 불필요 트래픽 제거 표준보안대책을 제시하여 망분리 기관의 표준모델을 정립에 도움을 주고자 한다..

향후에는 제시한 방법론을 지속적으로 운용하고 업무망에 도입되는 정보시스템들에 대하여 선 제거를 통하여 불필요 트래픽을 제거하고, 제거하지 못한 26%에 대하여 지속적으로 제거가 연구 된다면 망분리된 업무망 환경에서 Zero Day 취약점 발견도 어렵지 않을 것이라 생각된다.

References

〔1〕 "White Paper on the National Information Protection of 2018" Korea Internet &Security Agency 2018. 5

〔2〕 Bhun Chun-seok "A Study on the Network Separation and Related aws" Graduate School of Information and Communications, Konkuk University 2016

〔3〕 Kim, Yeon-Sang "Design and Implementation of Computer Center Security Policy Based on Network Separation" Daejeon University 2015. 2.

〔4〕 Han Chang-woo, "A Study on the Efficiency of Security Management in Network Separation Environments" Korea University 2014. 12.

〔5〕 Park, Hong-jae "Efficient Information Technology Architecture Management System (EAMS) A Study on Security Measures "Korea University June 6, 2012

〔6〕 "National Cyber Security Center, National Security Technology Research Institute 2015. 6

〔7〕 AhnLab, AhnLab TrusZone standard proposal

〔8〕 Utilization "Necessity and method of network separation" Ahnlab

〔9〕 Analyze and block Internet traffic in Intrrnet Border Zone (IBZ) Sogang University 2001. 7. 10

〔10〕 http://asec.ahnlab.com/937〔SEC Threat Research &Response blog〕

Binary Rewriting을 사용한 난독화된 신종 악성코드 식별 시스템

이영전, 한명묵*
가천대학교 컴퓨터공학과

Identification System of Obfuscated New Malware Using Binary Rewriting

Young-Jeon Lee, Myung-Mook Han*
Computer Engineering, Gachon University.

요 약

악성코드는 컴퓨터에 잠재적인 위험을 초래하는 코드를 말한다. 최근 몇 년간 악성코드는 급격하게 발달되었으며 컴퓨터 보안에 심각한 위협을 초래했다. 따라서 악성코드 식별은 컴퓨터 보안영역에서 중요한 과제로 떠올랐다. 하지만 기존의 악성코드 식별 모델들은 난독화된 악성코드를 식별하는데 어려움을 나타냈다. 특히 악성 코드의 고도화된 난독화와 신종 악성코드의 폭발적인 증가는 기존의 식별 식별시스템으로 식별할 수 없는 문제를 발생시켰다. 본 논문에서는 이러한 난독화된 신종 악성코드를 분석하기 위해 기계학습을 활용한 Binary Rewriting특징추출 기반 식별 시스템을 제안한다. 난독화된 바이너리 실행파일을 분석하는 기법인 Binary Rewriting을 통해 악성코드의 특성들을 추출하고 악성코드 분류를 위한 RF, NN 등의 머신러닝 기법을 적용시킨다면 보다 향상된 식별 시스템이 구축될 것으로 기대된다.

I. 서론

악성코드는 사용자 개인정보 탈취, 공격자 명령에 따른 다른 컴퓨터 공격 그리고 금전적인 피해 등 개인, 국가, 사회에 큰 피해를 남긴다. 문제는 이러한 악성코드 발전에 따른 탐지 및 피해방지가 어렵다는 것이다. 특히 악성코드 탐지 기술은 급변하는 악성코드의 속도를 따라가지 못하는 문제점이 있다.

기존의 악성코드 분류 시스템들은 크게 두 종류의 접근방식을 취했다. 압축 해제된 실행파일의 고정정적버전에서 가져온 특징 추출방식과[1] 실행 파일의 동적인 특징추출방식이다[2].

하지만 이러한 방식의 시스템의 경우 악성코드공격에 취약점을 나타낸다. 위의 시스템이 공개된 경우 공격자는 탐지를 피하기 위해 추출된 특징들만 난독화하면 되기 때문이다[3].

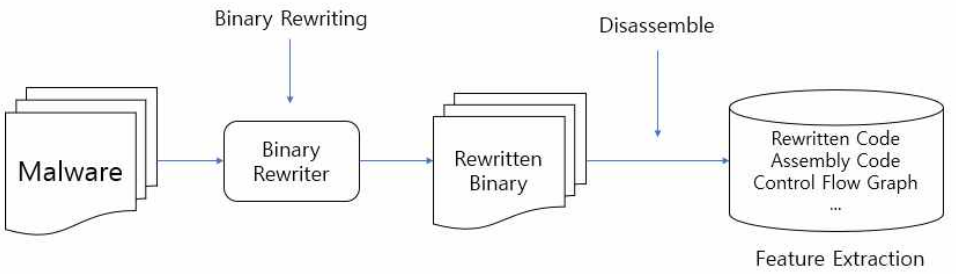
또한 이러한 시스템 방식은 새로운 유형의 악성코드에 대응할 수 없는 문제점이 있다. 학습데이터에 포함 되지 않은 새로운 유형의 변종 악성코드에서는 위의 특징들이 적용되지 않을 수 있기 때문이다.

본 논문에서는 이러한 난독화된 신종 악성코드를 탐지하기 위해 Binary Rewriting을 통한 난독화된 신종 악성코드 식별 시스템을 제안한다. 난독화된 악성코드를 식별 할 수 있도록 Binary Rewriting을 수행하며, 해독된 악성코드의 디스어셈블을 통한 특징들을 추출한다. 그리고 추출된 특징들을 기반으로 기계학습을 통한 악성코드 식별 시스템을 구축할 것이다.

II. 관련연구

2.1 Static Feature Extraction

*Corresponding Author: mmhan@gachon.ac.kr
이 논문은 2018년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07050864)



[그림 1] Binary Rewriting Process

Static Feature Extraction은 정적분석방법을 통한 악성코드의 특성을 추출하는 방법이다. 정적분석방법은 악성코드 탐지 및 식별 시스템에서 가장 널리 사용되는 방법 중 하나로 악성실행 파일에 역공학(Reverse Engineering)을 적용하여 특징을 추출하는 방식이다. 관련 연구로는 Portable Executable(PE), byte-sequence n-gram, string feature를 추출한 방법과[4] 프로그램 실행파일에 기록된 text strings를 추출하는 방법[5] 그리고 function-based feature를 추출한 방법[1]등이 있다.

하지만 정적분석방법에는 취약점이 존재한다. 우선 정적분석방법은 전문가에 의해서 악성코드의 특징으로 분류되는 Signature를 필요로 하며 이것은 시간이 많이 소요된다는 점이다. 또한 악성코드가 난독화 된다면 이러한 특징 추출을 피해갈 수 있다는 점이다[9].

따라서 악성코드 식별을 위한 분석에는 정적분석방법 뿐만 아니라 동적분석방법이 함께 수행되어야 한다[6].

2.2 Dynamic(run-time) Feature Extraction

동적분석방법은 악성코드를 가상환경 또는 샌드박스에서 실행시켜 프로그램 동작과정에서 발생하는 다양한 특징들을 추출하는 방법이다.

관련연구로는 Tian[7]이 제안한 가상 환경에서 악성코드 실행파일의 API call Function을 추출하고 패턴인식알고리즘과 통계기법을 적용한 방법과 악성코드 실행 중 시스템 호출을 포착하여 최대 공통 sub-graph를 탐지하는 방법 등이 있다[8].

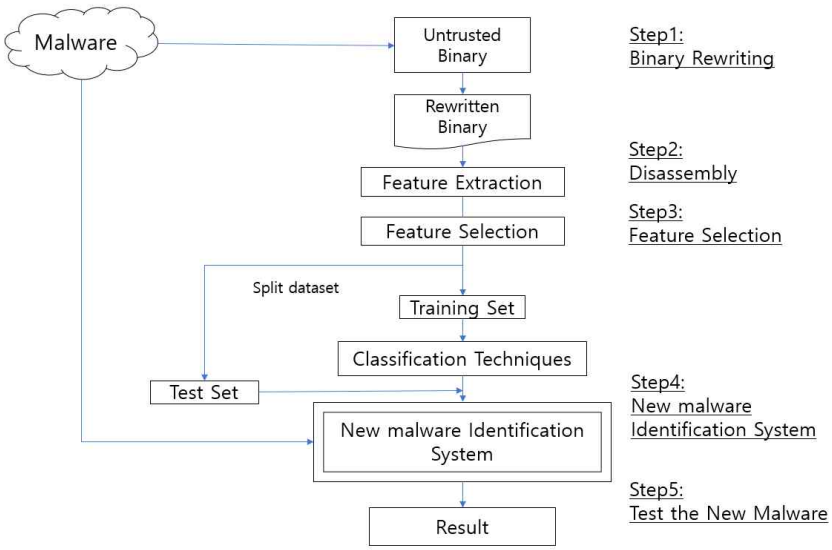
하지만 동적분석방법 또한 마찬가지로 취약점을 가지고 있다. 우선 동적분석은 정적분석보다 많은 시간이 소요된다. 분석을 통한 특징 추출을 위해서 악성코드 실행파일이 특정시간 동안 반드시 실행되어야하기 때문이다. 또한 분석 작업이 플랫폼을 감염시키지 않도록 실행파일들이 통제된 환경에서 기록되어야 한다. 이러한 통제된 환경은 악성코드가 유포되는 실제 환경과는 다르기 때문에 실제로 악성코드가 어떻게 동작하는지 파악하는데 어려움을 겪을 수 있다. 무엇보다 악성코드는 특정한 명령이나 상황에서 동작할 수 있기 때문에 한계가 있다[9].

2.3 Binary Rewriting

본 논문에서는 이러한 기존의 악성코드 분석 및 식별과정에서 발생하는 문제점을 해결하기 위한 방안으로 Binary Rewriting을 통한 난독화된 신종 악성코드 식별 시스템을 제안한다. 제안된 시스템을 통해 정적분석방법에서의 특징생성 어려움 문제와 난독화문제 그리고 동적분석방법에서 발생하는 가상환경에서의 악성코드분석에 대한 어려움을 해결하고자 한다.

[그림 1]는 Binary Rewriting을 통한 악성 코드 실행파일 변환 프로세스이다. Binary Rewriting란 바이너리 실행파일을 정의된 규칙에 의거하여 변환시키는 방법이다. Binary Rewriting은 바이너리 코드 재작성시 소스코드가 필요하지 않다는 장점이 있다. 즉 바이너리 상에서 모든 작업을 수행한다. 또한 코드 추가, 제거, 수정이 가능하며 바이너리 안에 불필요한 코드 삽입이 가능하게 하나, 반대로 불필요한 바이너리 코드들을 제거 할 수도 있다[10]. 난독화된 악성코드를 식별하기 위해 사용되는 방법 중 하나이다.

III. 악성코드 식별 시스템



[그림 2] Identification System of Obfuscated New Malware

제안하는 시스템의 구조는 [그림 2]와 같다.

3.1 Binary Rewriting 단계

우선 악성 코드의 실행파일로부터 얻은 바이너리를 Rewriting하는 과정을 거친다. 공격자는 어셈블리코드 사이에 탐지되지 않는 코드들을 기록해둔다. Rewriting과정을 통해 이러한 난독화 된 코드들의 특징들을 추출할 수 있다.

3.2 Disassemble 단계

다음으로 Disassemble을 거쳐 특징들을 추출하는 단계이다. Rewritten Binary를 디스어셈블 한다면 악성코드를 판별하는 특징들을 추출할 수 있을 것이다. 그리고 추출된 특징들을 학습시켜 난독화된 악성 코드 식별 시스템을 구축할 수 있게 된다.

3.3 Feature Selection 단계

추출한 특징들을 선택하는 단계이다. 어떤 특징들이 난독화된 악성코드들을 식별할 수 있는 특징인지 선별하는 단계이다.

3.4 Identification System 구축 단계

수집한 데이터를 Training Set과 Test Set으로 분류한 뒤 기계학습 기법인 RF(Random Forest), NN(Neural Network)를 활용하여 학습시킨다. 학습을 통해 난독화된 신종 악성코드를 식별할 수 있는 모델을 만들 수 있다. 구축된 모델은 Verification을 거쳐서 향상된 모델로 거듭날 수 있도록 한다.

3.5 New Malware Test 단계

한편 악성코드 식별 시스템은 난독화된 신종 악성코드를 정확히 식별해야한다. 구축된 모델에 신종 난독화된 악성코드를 Test함으로써 식별 시스템의 성능을 평가할 수 있다.

IV. 결론

본 논문에서는 난독화된 신종 악성코드를 식별하기 위해 Binary Rewriting을 통한 특징추출을 기반으로 악성코드를 식별 시스템을 제안했다.

기존의 악성코드 판별 시스템의 경우 신종 악성코드를 판별하는 데 어려움을 겪었다. 특히 난독화된 바이너리 코드 해독의 어려움을 해결하고자 수행한 동적분석의 경우 프로그램 파일을 일일이 가상환경에서 실행시켜야 하는 수고에도 불구하고 제대로 된 특징 추출의 어려움이 있었다. 이러한 기존의 문제점을 Binary Rewriting을 통한 특징추출로 해결할 수 있을 것으로 기대된다. 그리고 무엇보다도 난독화 되어있는 신종 악성코드를 식별해 낼 수 있을 것으로 기대된다. 향후 연구로는 Linear Sweep와 Recursive Disassemble을 통한 악성코드의 특징 추출을 통한 기계학습 실험이다. 연구의 최

종 목표는 악성코드의 저자를 식별하는 시스템 구성이다[13].

[참고문헌]

[1] Tian R, Batten L, and Versteeg S, "Function length as a tool for malware classification," In: Proceedings of the 3rd international conference on malicious and unwanted software, pp. 69 - 76, 2008.

[2] Kolbitsch C, Comparetti PM, and Kruegel C, "Effective, and efficient malware detection at the end host." In: Proceedings of the 18th conference on USENIX security symposium, Berkeley, CA, USA: USENIX Association, pp. 351 - 66, 2009.

[3] Moser A, Kruegel C, and Kirda E, "Limits of static analysis for malware detection," In: Twenty-third annual computer security applications conference, pp. 421 - 30, 2007.

[4] Schultz MG, Eskin E, Zadok E and Stolfo SJ, "Data mining methods for detection of new malicious executable" In: Proceedings of the 2001 IEEE symposium on security and privacy. pp. 38 - 49, 2001.

[5] A. Shabtai, R. Moskovitch, Y, and Elovici, C, "Detection of malicious code by applying machine learning classifiers on static features," a state-of-the-art survey Elsevier Advanced Technology Publications, pp. 16 - 29, 2009.

[6] E. Menahem, A. Shabtai, and L. Rokach, Y, "Improving malware detection by applying multi-inducer ensemble," Computational Statistics and Data Analysis, pp. 1483-1494, 2009.

[7] Tian R, Islam R, Batten L, and Versteeg S. "Differentiating malware from cleanware using behavioural analysis." In: Proceedings of the 5th international conference on malicious and unwanted software, pp. 23 - 30 , 2010.

[8] Park Y, Reeves D, and Mulukutla V, "Fast malware classification by automated behavioral graph matching," In: Proceedings of the sixth annual workshop on cyber security and information intelligence research, pp. 45:1 - 45:4, 2010.

[9] R.Islam, R.Tian, and Lynn M.Batten, S. "Classification Of Malware Based on integrated static and dynamic Features," Journal of Network and Computer Applications, pp. 646-656, 2012.

[10] R.Wartell, V.Mohan, and K W.Hamlen, "Securing Untrusted Code Via Compiler-Agnostic Binary Rewriting," the 28th Annual Computer Security Applications conference, pp. 299-308, 2012.

실습형 정보보안 교육 환경 구축 모델에 대한 연구

지한별* 김보경* 윤원석*
*라온화이트햇

A Study on Implementing Model of Practical Information Security Education Environment

Han Byeol Ji* Bo Kyung Kim* Won Seok Yoon*
*RAON Whitehat

요 약

최근 정보보안 분야에 대한 사회적 관심이 높아지면서 대학 등 교육기관에서는 정보보안 전문인력 양성에 많은 노력을 들이고 있다. 하지만, 국내 정보보안 교육 현장의 현실은 전문교육자의 부족과 실습환경의 미비로 인해 이론 중심의 교육이 이루어지고 있어, 산업 현장에서 기대하는 전문기술과 실무 능력을 갖춘 인력을 배출하기는 어려운 상황이다. 본 논문에서는 국내 정보보안 실습 환경 현황을 살펴보고 교육자와 학습자의 관점에서 효율적인 실습 환경 구축을 위한 필요 요건을 도출하고자 한다. 그리고, 도출된 필요 요건을 반영하여 실습형 정보보안 교육 단계를 기초교육-실습-평가의 세 단계로 설계하고, 시간적 공간적 제약 없이 효율적인 실습이 가능한 웹기반의 정보보안 실습형 교육 환경 모델을 구성하였다. 본 논문을 통해 새로이 설계된 실습형 정보보안 교육 환경 모델은 학습자의 실습 환경을 획기적으로 개선함으로써 기업현장에서 필요한 실무 능력을 갖춘 정보보안 인력을 양성하는데 기여할 것으로 기대한다.

I. 서론

최근 산업계 전반의 다양한 정보보안사고 발생과 4차 산업혁명의 기반요소로서 정보보안 분야에 대한 관심의 증가에 따라 다수의 교육기관들이 정보보안학과 및 유관학과를 개설하고 전문인력을 배출하기 위해 노력하고 있다. 대학의 전문인력 육성과 관련하여 전효정[1] 등은 정보보호 분야 직무별 필요지식 및 기술분석에 대한 연구를 통해 전문인력에게 요구되는 기술분야를 분류하였고, 이효직[2] 등은 정보보호 전공자의 영역별 전공필수와 선택과목에 대한 분류를 제시하였다. 선행 연구들은 공통적으로 정보보안 전문인력의 요건으로 기술적 능력을 꼽고 있으나, 기술능력의 효과적인 육성을 위해 필수적인 실습 환경의 구축에 대해서는 아직 구체적인 연구성과가 없다. 이로 인해 대학 정보보안학과 및 대학원 등 교육기관을 통해 배출되는 전문인력의 실무능력은 기업, 금융기관 등 수요자의 필요에 못미치고 있어 산업계의 낮은 만족도와 재교육에 따른 추가적 비용 소요 등 다수의 문제점이 나타나고 있다. 본 연구에서는 정보보안 교육 현황과 문제점을 살펴보고, 전문인력의 기술적 능력을 효과적으로 배양할 수 있는 실습형 정보보안 교육 환경 구축 모델을 제시함으로써, 기업 실무에 즉시 적용 가능한 전문기술인력 양성을 지원하고, 궁극적으로 교육기관과 기업 간의 직무 불일치를 해소할 수 있는 방안에 대하여 논하고자 한다.

II. 정보보안 실습 교육 현황과 문제점

II장에서는 효과적인 정보보안 실습 교육 방향을 도출하기 위해 국내 정보보안 교육의 현황과 문제점을 살펴본다.

2.1 기업의 필요에 미흡한 전문인력 배출

이문구[3] 등의 연구에 따르면 정보보안 직무에 종사하는 산업체 담당자들을 대상으로 한 설문 결과, 산업체에서 필요로 하는 정보보안 전문 인력은 원만한 대인능력을 기반으로 하는

문서작성능력과 네트워크 및 통신보안 기술을 기본으로 갖추어야 한다는 답변이 가장 많았으며, 서버보안 시스템 구축 능력 및 취약점 분석 능력이 필요하다는 답변이 다음으로 많았다. 하지만 대학 등 교육기관은 대부분 실무에서 필요로 하는 각 기술들에 대한 개념적인 내용만 다룰 뿐 실습을 통한 실질적인 학습 환경을 구축하여 제공하지는 못하고 있다.

2.2 정보보안 분야 전문교육자 확보의 어려움

현재 정보보안 전문 교육자 육성 및 지원제도가 미흡한 상황에서 업무와 강의를 병행하는 교육자가 많은 추세이다. 지식정보보안 전문 인력 양성 과정의 경우, 강사진 전원이 한국인터넷진흥원의 연구원으로 구성되어 있으며, 행정안전부의 개인정보보안 교육은 한국인터넷진흥원, 정보보안 전문 업체, 대학 등의 전문가를 강사로 활용하고 있다. 일부 공공기관의 경우에는 민간 정보보안 교육기관에 정보보안 교육을 위탁하여 운영하고 있다. 이 경우, 교육을 실제 업무를 수행하고 있는 인력이 진행하므로 전문성을 확보할 수 있다는 것이 장점이나, 업무와 교육을 함께 진행해야 하는 상황에서는 교재 개발 및 강의 준비, 과제 검토 등의 측면에서의 문제가 있다.

또한, 정보보안 교육은 다양한 세부 분야로 구성되며 분야별로 각각의 전문성을 요구함에도 불구하고 전문교육자 부족으로 인해 교육자가 자신의 전문 분야가 아닌 분야에 대해서 교육을 하는 경우도 있다. 교육자가 직접 교재를 제작하고 강의를 진행하는 현실을 고려할 때, 한 분야의 정보보안 전문가라 해도 자신의 전문 분야 외 교육까지 진행할 경우, 교육 내용이 부실해질 가능성이 높아진다.

2.3 교육기관별 자체적 실습 환경 확보 미흡

기업에서 요구하는 인력양성 및 직무상 수요를 만족하기 위해서는 모의해킹, 취약점 진단 등 전 분야에서 실습형 교육이 병행되어야 한다. 하지만, 현재 이루어지고 있는 대부분의 교육에서는 실습용 전문 프로그램이 없어 교육자가 준비한 실습형 데모 영상이나 유투브 등에 등록되어있는 해킹 시연 영상 등을 통하여 간접적인 경험만 수행하는 경우가 많으며, 본격적인 실습을 위한 프로그램 구축에는 과도한 비용과 전문기술력이 요구되어 자체적인 실습 환경을 확보하지 못한 경우가 대부분이다. 이로 인해 학습자는 일상적인 실습을 수행하기 어려운 현실이다.

2.4 비관련학과 출신자를 위한 전문교육 프로그램 부족

한국인터넷진흥원[4]의 조사 결과에 따르면, [표 2]의 국내 정보보안 업계에 종사하는 인력 현황과 같이 비관련학과가 68,477명으로 가장 많고, 정보보안 관련학과 39,897명, 정보보안 학과/전공 15,369명으로 조사된 것을 확인할 수 있다.

구분	정보보안학과/전공	정보보안 관련학과(통신,전산,정보통신 학과 등)	비관련 학과	전체
전체	15,369	39,897	68,477	123,743
공공부분	496	2,117	1,384	3,997
보안업체	10,279	16,208	13,441	39,928
비보안업체	4,594	21,572	53,652	79,818

[표 2] 국내 정보보안 인력 현황(2016년) (단위: 명)

이처럼 정보보안 비관련학과 출신 인력들이 정보보안 업계에 활발히 활동하고 있으므로, 이들에게 효율적으로 실무를 익히고 기술력을 배양하기 위한 실습형 정보보안 교육 환경의 제공도 절실하다.

2.5 시사점

정보보안 교육 환경에 대한 분석 결과는 다음과 같이 요약할 수 있다.

첫째, 기업의 필요에 맞는 실무능력을 갖춘 정보보안 전문인력을 육성하기 위해서는 전문기술을 활용한 체계적인 실습형 교육 환경의 제공이 필수적이다.

둘째, 정보보안 영역별 전문성을 고려하여 실습형 정보보안 교육 환경도 모의해킹, 취약점 진단 등 기술 영역별로 제공되어야 한다.

셋째, 교육자의 개별적 역량과 노력에 대한

의존도를 낮추면서 동시에 교육자가 필요에 따라 활용할 수 있는 형태의 실습형 교육 환경 구축이 필요하다.

넷째, 실습형 정보보안 교육 환경은 관련학과 전공자 뿐만 아니라 비관련학과 출신 학습자의 기술 수준까지 고려하여 초급, 중급, 고급 등 등급별로 제공되어야 한다.

다섯째, 실습형 정보보안 교육 환경은 교육의 효율을 높이기 위해 학습자가 시간적 공간적 제약을 최대한 받지 않도록 제공하여야 한다.

III. 실습형 정보보안 교육 환경 구축 모델

III장에서는 II장에서 살펴본 실습 환경의 문제를 해결하기 위한 필요 요건을 정의하고, 필요 요건을 반영한 실습형 정보보안 교육 환경 구축 모델을 제시한다.

3.1 실습형 정보보안 교육 환경 필요 요건

실습형 정보보안 교육 환경 필요 요건을 교육자, 학습자 입장으로 나눠 정리하였다.

3.1.1 교육자 입장

II장에서 살펴본 것과 같이 교육자는 도구의 부재로 인한 실습형 교육 운영상 어려움이 존재한다. 이를 해결하기 위하여 기존의 개념 위주 교육이 아닌 실습형 교육을 위한 새로운 교육장이 마련되어야 한다. 실습형 교육장에는 개념 학습을 위한 기초교육 환경, 실습형 학습을 위한 실습 환경, 실습 결과를 평가하기 위한 평가 환경이 제공되어야 한다.

3.1.1.1 기초교육 환경

실습형 교육이 진행되기 위해서는 실습의 기본 개념을 학습할 수 있는 기초교육을 위한 교육자료와 강의내용이 필요하다. 기존의 개념교육과 달리 실습 주제에 초점을 맞춘 기초학습이 이루어지도록 학습내용과 강의가 제공되어야 한다.

3.1.1.2 실습 환경

기초교육 후 이해도를 확인하고 기술을 실제로 활용하는 방법을 교육하기 위한 실습 환경이 필요하다. 실습 환경은 학습자가 원하는 시점에 원하는 방식으로 실습문제와 함께 문제해결을 위한 도구가 같이 제공되어야 한다.

3.1.1.3 평가 환경

실습이 진행된 후 교육자가 학습자를 평가하기 위해서는 적절한 평가 환경이 필요하다. 실습 환경과 유사한 평가 환경을 구상하여 개념에 대한 이해력 및 응용력을 확인하기 위한 객관적 평가 도구가 제공되어야 한다.

3.1.2 학습자 입장

II장에서 살펴본 것과 같이 학습자에게는 기존의 개념 위주의 수동형 학습 방식이 아닌 개념 학습 이후 실습까지 이어지는 형식의 능동형 학습이 필요하다.

3.1.2.1 접근성과 편리성

해킹·보안의 다양한 분야에서 효율적인 학습을 위해서는 도구에 대한 접근성과 편리성이 제공되어야 한다. 개념학습과 실습이 한 곳에서 통합적으로 제공되도록 웹기반의 실습 환경이 필요하다.

3.1.2.2 단계별/수준별 학습

학습자의 학습수준에 따른 단계별/수준별 학습 실습 환경이 필요하다.

3.1.2.3 실무와 관련된 학습

실무 능력을 갖추기 위한 실습내용이 제공되어야 하며 기업 현장의 이슈를 중심으로 지속적으로 실습환경이 업데이트되어야 한다.

3.2 실습형 정보보안 교육 환경의 구성

3.1절에서 살펴보았던 교육자와 학습자의 입장을 반영하여 웹기반의 실습형 정보보안 교육 환경을 구성한다.

[그림 1] 실습형 정보보안 교육 환경 구성

실습형 정보보안 교육 환경은 교육시스템과 관리시스템으로 구성한다. 교육시스템은 학습자와 교육자를 각각을 위한 웹 페이지로 구성한다. 이를 통해 학습자는 강의정보 확인, 강의수강, 학습활동, 수강현황 등의 기능을 사용할 수 있도록 하고, 교육자는 강의 정보관리, 강의실 관리, 성적 평가, 교육 관리 등의 기능을 사용할 수 있도록 한다. 관리시스템에서는 실습형 정보보안교육 환경의 사용자, 교육과정, 교육일정, 강의운영 등을 관리할 수 있도록 한다.

3.2.1 기초교육 환경

실습교육을 위한 사전 교육 단계로, 정보보안에 대한 기초 지식을 학습하는 환경으로 구성된다. 웹 기반으로 교육 자료를 실시간으로 확인 할 수 있도록 구성하여 실습과 연계성을 가질 수 있도록 한다.

3.2.2 실습 환경

기초교육 후 개념에 대한 내용을 실무적인 관점에서 실습을 통해 체화할 수 있도록 환경을 구성한다. 학습자의 흥미와 학습욕구를 증대시키기 위해 실습을 단계별로 구성하여 미션을 해결하는 방식으로 구성하고, 타 학습자들과 자신의 학습수준을 비교할 수 있는 스코어보드 등을 활용한다.

3.2.3 평가 환경

기초교육을 통한 개념학습과 실습을 통한 실무적인 관점에서의 학습 이후에는 학업성취도를 평가할 수 있는 환경이 필요하다. 최신 이슈와 기술 트렌드를 반영하여 문제를 출제하고, 이를 통해 학습자의 현재 수준을 파악하여 보충학습 및 심화학습에 활용할 수 있도록 한다.

IV. 결론

본 논문에서는 현 정보보안 교육 환경의 문제점을 분석하여 교육자와 학습자의 관점에서 필요 요건을 도출하고, 이를 활용하여 웹기반의 기초교육-실습-평가 단계를 포함하는 실습형 정보보안 교육 환경 구축 모델을 제시하였다.

본 논문을 통해 제시된 구축 모델을 활용하면 교육기관에서는 기존의 개념 위주 교육의 문제를 해결하고, 전문 교육자를 확보하지 않더라도 실습형 교육 환경을 운영할 수 있을 것이다. 또한 정보보안 분야로 진출하고자 하는 비전공자에게도 웹기반의 실습 환경은 쉽게 접근하여 활용할 수 있으므로 매우 유용한 교육 도구가 될 것이다.

향후 연구에서는 본 논문에서 제안한 웹기반의 실습형 정보보안 교육 환경 구축 모델을 실제 교육환경에 맞게 구축할 방안을 제시해 보고자 한다.

[참고문헌]

[1] 정보보호 분양 직무별 필요 지식 및 기술 분석, 전효정, 한국경영정보학회, 2008.8, p.253-267

[2] 보안전문인력 양성을 위한 직업분류체계별 정보보호 핵심지식 설계, 이효직, 한국전자거래학회지, 2015.8, p113-125

[3] 정보보안 전문 인력 양성을 위한 교육과정 개발, 이문구, 전자공학회논문지, 2017.1, p.49

[4] 2016 정보보호 인력수급 실태조사, 과학기술정통부, 한국인터넷진흥원, 2017.8, pp.17

AIX 운영체제에서의 엔트로피 측정 및 분석

송행권*, 김현기*, 이옥연*
*국민대학교 금융정보보안학과

Entropy measurement and analysis on the AIX operating system

Haeng-Gwon Song*, Hyun-Ki Kim*, Ok-Yeon Yi*
*Division of Financial information security, Kookmin University.

요 약

암호학적으로 난수는 암호시스템의 안전성을 책임지는 중요한 요소이다. 난수에 대한 대표적인 통계적 테스트로 SP 800-90B, SP 800-22 등이 있다. 이 중 SP 800-90B는 잡음원의 안전성을 측정하는 방법으로써 난수 생성 시 사용되는 잡음원의 엔트로피 테스트를 수행하는 표준이다. 본 논문에서는 AIX 운영체제에서 난수를 생성하기 위해 수집되는 잡음원을 SP 800-90B에서 언급하고 있는 통계적 방법을 이용하여 엔트로피 테스트를 수행하여 AIX 운영체제에서 암호학적 난수를 생성할 때 사용 할 수 있는 잡음원을 제안하며 제안한 잡음원의 평가를 수행한다.

I. 서론

미국 시장조사기관에 가트너(Gartner)에서 발표한 2018년 10대 전략기술을 살펴보면 Intelligent digital mesh라는 개념을 정의하면서 사람, 장치, 컨텐츠 및 서비스를 통합하여 모든 사물이 연결된다는 개념을 정의하고 있다. 현시대는 IoT 환경을 넘어서 우리가 사용하고 있는 모든 디바이스들이 촘촘하게 연결되어 정보를 교환할 것이라고 정의하고 있다.

다양한 형태의 디바이스들이 인터넷과 결합되면서 개인의 프라이버시를 침해 할 수 있는 개인정보 혹은 비밀 정보들이 인터넷상에서 전송되고 있다. 이러한 데이터들의 보안 위협을 막기 위해 현대 암호를 적용하여 데이터의 암호화를 수행한다. 현대 암호는 Kerckhoff의 원칙에 따라 키가 안전성을 결정한다. 키는 잡음원을 이용하여 난수발생기에서 발생한 랜덤 값을 사용하여 생성하게 된다. 만약 키 생성 시 낮은 엔트로피의 잡음원을 사용한다면 비밀정보인 키의 값을 예측 할 수 있음으로, 키의 안전도는 급격하게 낮아질 것이다.

본 논문에서는 AIX 환경에서 암호학적 난수를 사용하는 오픈소스(ex. OpenSSL, Ethereum)에서 안전한 난수를 생성 할 수 있게 하도록 운영체제에서 생성되는 잡음원의 엔트로피 분석을 연구하였다.

본 논문의 구성은 2장에서 엔트로피가 무엇인지 그리고 엔트로피를 평가할 수 있는 통계적 방법을 정의하고 있는 SP 800-90B를 소개한다. 3장에서는 AIX OS에서 잡음원으로 이용할 수 있는 함수를 소개하고 4장에서는 그 잡음원을 평가를 수행한 결과를 명시한 후 5장에 결론을 제시한다.

II. 엔트로피

암호학적으로 비밀키 값을 생성하는 것은 중요한 일이다. 비밀키는 사용자 외에는 아무도 예측 할 수 없어야 하기 때문에 높은 엔트로피

를 갖는 잡음원을 이용하여 난수를 생성해야 한다.

2.1 엔트로피란 무엇인가?

암호학에서 정의하는 엔트로피는 클로드 섀넌(Claude Shannon)의 정의에 따라 데이터가 가지는 정보량을 수치적으로 나타낸 것으로 무질서도(Disorder) 또는 난수성(Randomness)을 나타내며 난수에 가까울수록 엔트로피가 높다. 엔트로피 측정은 Renyi Entropy, Min Entropy, Shannon Entropy 등이 있다. 본 논문에서는 SP 800-90B에서 사용하고 있는 가장 보수적인 엔트로피 측정 방법인 Min Entropy를 사용하여 엔트로피를 측정한다.

2.2 SP 800-90B

SP 800-90B는 잡음원에서 출력된 표본 수열에 대한 통계적 검증으로 IID Track과 Non IID Track으로 나뉜다. IID Track은 IID(Independent and Identically Distributed)특성을 확인하는 검정을 먼저 실시 한 후 IID 데이터로 간주 된 표본 수열의 엔트로피 측정을 수행한다. 그러나 IID 특성을 통과하지 못하면 표본 수열은 Non IID Track으로 테스트를 수행한다.

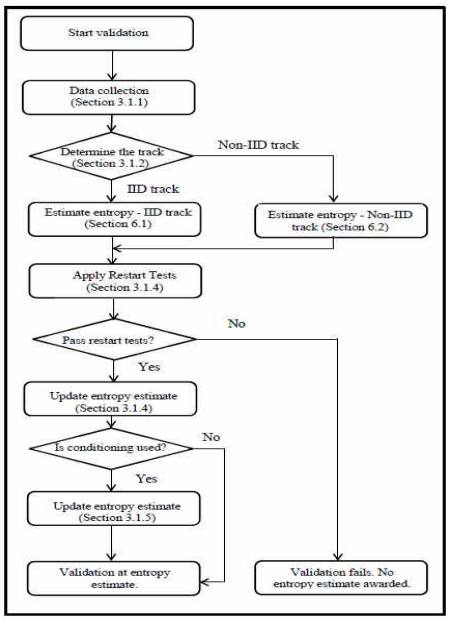


그림 1 Entropy estimation strategy

IID Track은 Permutation 테스트와, Chi-squre test 두가지의 타입으로 나뉜다. Permutiation 테스트는 알 수 없는 분포를 가진 샘플에 대한 통계적 분석을 수행하는 테스트로 세부적으로 11가지의 테스트가 있다. Chi-squre 테스트는 근사 분포를 가진 샘플에 대한 통계적 분석을 수행하는 테스트로 5가지의 테스트가 존재한다. Non IID Track은 IID Track을 통과 하지 못한 엔트로피를 평가하는 평가 방법으로 다섯가지의 테스트 방법이 존재한다

IID Test	Permutation Test	1. Excursion 2. Number of Directional runs 3. Length of directional runs 4. number of increases and decreases 5. number of runs based on the median 6. length of runs based on median 7. average collision 8. Maximum collision 9. Periodicity 10. Covariance 11. Compression
	Chi Squre Test	12. independence for non-binary data 13. goodness of fit for non binary data 14. testing independence for binary data 15. goodness of fit for binary data 16. length of the Longest Repeated Substring

Non-IID Test		1. Frequency 2. Collsion 3. Compression 4. Partial Collection 5. Markov

표 2 IID vs Non IID Test 분류표

SP 800-90B 최소 엔트로피 추정 과정은 그림 1과 같다. 엔트로피 소스의 최소 엔트로피 추정을 위해 1,000,000개 이상의 출력 표본 수열을 입력하도록 권장하고 있으며, 이 때 표본 수열은 고정된 길이의 비트열을 입력받는다.

본 논문에서는 NIST에서 제공하고 있는 최종 버전은 엔트로피 측정 툴을 이용하여 잡음원의 엔트로피를 측정한다.

III. AIX에서 제공하는 잡음원

암호기능을 사용하는 오픈소스에서는 운영체제의 잡음원을 이용하여 암호키 혹은 비밀 정보를 생성하는 데 사용한다. 운영환경에 따라 달리 수집되는 잡음원은 결정론적 난수발생기를 통해 생성하는 난수의 안전성에 매우 중요하다.

난수발생기에 사용되는 잡음원은 보통 운영체제에서 가변하는 시스템 데이터들을 사용한다. 2.2절에서 설명한 SP 800-90B에서 제시하는 통계적 기법을 이용하여 잡음원의 엔트로피 테스트를 수행 할 수 있다.

운영체제에서 제공되는 잡음원의 수집방법은 운영환경에 따라 상이하다. 본 논문에서는 AIX 운영체제에서 잡음원으로 사용 할 수 있는 함수 및 명령어를 호출하여 잡음원을 수집하였다.

본 절에서 수집한 잡음원은 크게 두 가지로 구분된다. 첫 번째로 AIX 운영체제에서 제공하는 난수발생기의 출력을 잡음원으로 수집하였다. AIX에서 수집 할 수 있는 잡음원은 OS에서 제공하는 커널 디바이스인 /dev/urandom 과 /dev/random 호출하여 그 출력을 사용하였다. 두 번째로 OS에서 제공하는 커널 디바이스 출력 외 추가 잡음원을 제공하는 함수 및 명령어를 이용하여 잡음원을 수집하였다.

수집정보	함수명 혹은 명령어(#)
가상메모리 통계	# vmstat 1 1
시스템 부팅시간	# uptime
시스템의 현재시간(초)	gettimeofday_sec
시스템의 현재시간(밀리초)	gettimeofday_usec

표 3 AIX운영체제 잡음원 제공 함수 및 정보

IV.실험 및 분석

본 실험은 AIX 6.1버전이 동작하는 시스템에서 표 2의 정보를 수집하여 잡음원의 엔트로피 분석을 수행하였다.

운영체제	AIX 6.1
프로세서	PowerPC_POWER5 64bit
메모리	3744 MB
컴파일러	gcc 4.8.4

표 4 실험환경 명세

본 논문에서는 SP 800-90B의 분석 도구를 이용하기 위하여 각각의 수집 정보들을 1,000,000개의 샘플을 수집하여 통계적 분석을 수행한다. 또한, 표본의 크기는 SP 800-90B의 6.4절에 언급된 Reducing the symbol space 에 언급된 대로 8비트 단위의 샘플의 바이너리 파일로 출력하여 테스트를 수행하였다.

수집정보	엔트로피 평가
가상메모리 통계	0.8756541
시스템 부팅시간	0.000582943
시스템의 현재시간(초)	0.00200779

시스템의 현재시간 (밀리초)	3.11414
/dev/random	3.93867
/dev/urandom	3.98367

표 5 AIX운영체제 잡음원 제공 함수 및 정보

V.결과

현대 암호를 사용하는데 있어서 키의 안전성은 매우 중요하다. 하지만 오픈소스에서 난수를 생성하기 위해 사용되는 잡음원은 암호에서 요구하는 만큼 충분하지 못하다. 그러므로 암호학적으로 안전한 난수를 생성하기 위해서는 오픈소스에서 난수 생성을 위해 많이 사용되고 있는 /dev/(u)random 이외에도 난수가 예측되지 않도록 하기 위하여 추가적인 잡음원을 이용하여 난수를 생성해야 한다.

본 논문에서는 OS에서 제공하는 함수와 명령어들을 이용하여 잡음원 분석을 수행하였다. 하지만 난수를 제공해주는 디바이스 이외의 함수 및 명령어에서 수집한 잡음원들은 엔트로피가 매우 낮게 나왔다. OS에서 생성하는 잡음원은 한계가 존재하며 만약 사용하고자 한다면 후처리 과정 후 잡음원 사용하는 것을 권고한다.

향후 AIX제공하는 시스템 함수, 명령어들을 분석하여 더 높은 엔트로피를 갖는 잡음원에 대한 분석을 수행하여 더 높은 엔트로피를 갖는 잡음원을 수집하는 것이 필요하다. 더불어 잡음원의 사후처리와 관련한 연구를 수행하여 결정론적 난수발생기에 사용되는 잡음원의 엔트로피의 안전성을 높이는 것이 필요하다.

[사사 정보]

본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

[참고문헌]

[1] NIST, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90Ar1, June, 2005

[2] NIST, Recommendation for Random Bit Generator (RBG) Constructions, Special Publication 800-90C, April, 2016

[3] NIST, Recommendation for the Entropy Sources Used for Random Bit Generation, Special Publication 800-90B, January, 2018

[4] H. Park, M. Bae, Y. Yeom, and J. S. Kang, "A study on the predictor of Non-IID track in SP 800-90B," in Proc. KICS Int. Conf. Commun., pp. 115-116, Jeju, Korea, Jun. 2016.

[5] NIST, SP 800-90B_EntropyAssesment, https://github.com/usnistgov/SP800-90B_EntropyAssessment , accessed Oct. 2018

VR 상에서의 안전한 PIN 입력 방법 제안

우재민 · 김현준 · 김도영 · 서화정*
*한성대학교 IT융합공학부

Proposal of Safe PIN Input Method on VR

Jae-Min Woo · Hyunjun Kim · Doyoung Kim · Hwajeong Seo*
*Department of IT, Hansung University, Seoul, 02876 Korea

요 약

가상현실 속에서 실제와 같은 서비스를 제공하는 기술인 VR (Virtural Reality)는 실제와 유사한 공간적 체험을 할 수 있도록 Head Mounted Display (HMD) 기기를 이용하여 영상과 음성의 출력 그리고 사용자 입력을 제공한다. 현재 VR을 활용한 많은 서비스들이 개인화되어 있는 만큼 사용자에 대한 안전한 인증이 매우 중요하다. 하지만 VR 기기 상에서 제공하는 Personal Identification Number (PIN)은 해커가 Shoulder Surfing Attack (SSA)을 이용하여 사용자의 입력 패턴을 분석하게 될 경우 비밀번호가 그대로 노출되는 문제점을 가지고 있다. 본 논문에서는 사용자의 편의성은 그대로 유지하면서 해커가 입력 패턴을 분석하더라도 사용자의 비밀 번호를 안전하게 보호할 수 있는 방법에 대해 제안한다. 이는 립모션을 활용한 두 가지 입력 방법과 여러 센서와 아두이노를 활용하여 구현된 스마트 글러브를 이용하여 입력하는 방법으로 총 세 가지의 새로운 형태의 안전한 PIN 입력 방법을 포함한다.

I. 서론

최근 스마트폰 시장이 포화 상태에 도달함에 따라 IT기업들은 새로운 기술을 발굴하는게 큰 화두가 되었다. 이에 IT기업들이 주목하는 기술이 가상현실이다. 가상현실이란 컴퓨터 등을 사용한 인공적인 기술로 만들어낸 실제와 유사하지만 실제가 아닌 어떤 특정한 환경이나 기술 그 자체를 의미한다 [1]. 사용자가 Head Mounted Display(HMD) 기기를 사용한다면 HMD 기기를 통해 가상 현실 속에서 음성과 영상을 제공받아 여러 컨텐츠 서비스를 이용할 때 더욱 몰입감 있게 체험할 수 있다. VR은 여러 분야에서 활용이 가능한데 크게 게임, 교육, 의료, 영상 등 분야에 활용이 되고 있다.

VR이 여러 분야에 활용이 되면서 사용자에게 많은 서비스가 제공이 되고 개인화가 이루어 지면서 이제는 VR 상에서 사용자에 대한 인증이 매우 중요해졌다. 현재 VR 상에서 제공하는 Personal Identification Number(PIN)를 사용하게 되면 VR의 특징상 HMD 기기를 착용을 하면 주변 환경을 인식하지 못하는 취약점으로 인해 해커는 사용자의 어깨너머로 훔쳐보는 공격인 Shoulder Surfing Attack (SSA)를 수행하여 사용자의 PIN 입력 패턴을 분석하게 되면 사용자의 PIN은 고스란히 노출이 된다. 따라서 본 논문에서는 립모션과 스마트 글러브를 활용하여 해커의 shoulder surfing attack을 통한 입력 패턴 분석을 방지하면서 사용자의 편의성을 보장하는 새로운 형태의 PIN 입력 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 VR과 립모션의 관련 연구 동향에 대해 살펴본다. 3장에서는 안전한 PIN 입력을 위해 립모션과 스마트 글러브를 활용한 제안 기법에 대해 설명을 진행하며 4장에서는 본 논문의 결론을 맺도록 한다.

II. 관련 연구 동향

본 장에서는 제안하고자 PIN 입력 방법을 설명하기 전에 VR, 립모션에 대해 소개하고 간단하게 관련 연구 동향에 대해서 알아보도록 한다.

2.1 Virtual Reality(VR)

최근 VR을 체험할 수 있는 HMD 기기의 성능이 향상되면서 VR에 대한 관심이 증가하고 여러 분야에서 VR을 활용한 다양한 연구가 진행이 되고 있다. 교육 분야에서는 VR을 사용하여 다양한 교육용 시뮬레이션이 제안되고 있다. 소방안전과 같이 사실적인 체험이 필요하지만, 현장에서 경험할 수 없는 한계를 가지고 있으며 사고가 발생할 수 있는 위험성이 있기 때문에 VR을 사용하여 가상의 사고 현장을 구현하고 사용자는 구현된 가상의 사고 현장에서 오감을 이용하여 능동적인 학습자가 될 수 있다 [2].

의료 분야에서도 VR을 활용하여 환자들의 치료를 위한 다양한 연구가 진행되고 있다. 특정 장소나 상황을 싫어하는 증상인 공황장애는 보통 노출훈련이라는 치료 방식을 적용한다. 하지만 노출훈련은 비용, 인력, 시간이 많이 소모되기 때문에 공황장애 환자 및 공황장애 환자일 가능성이 있는 사람에게 앱을 통해 불안지수를 측정하고 노출훈련의 문제점을 해결한 VR 기술을 적용한 노출훈련 시스템을 통하여 환자들의 공황장애를 치료하는 방안이 제안되었다 [3].

이처럼 현재 VR은 교육, 의료 분야에서 활발하게 연구되고 있으며 해당 분야가 아니더라도 구현된 오브젝트와 상호작용이 가능하다는 장점은 다양한 분야에서 적용이 될 수 있기 때문에 가상현실에 대한 연구는 더욱 활발해질 것이다. 하지만 가상현실에서의 보안에 대한 연구는 아직 다른 분야에 비해 미흡하다. 현재 가상현실 기기와 게임 시스템의 정보보증을 위해 보안 위협을 식별하였고 이를 보호하는 방법이 제안되었지만 [7], 가상현실 기술이 보편화되기 위해서는 보안에 대한 연구는 앞으로 더욱 더 활발하게 진행되어야 할 것이다.

2.2 립모션

NUI 장치의 일종인 립모션은 사용자의 손동작을 3D 정보로 얻어 컴퓨터의 입력으로 활용할 수 있는 대표적인 제품이다. 초 당 200번씩 사용자의 동작을 인식할 수 있고 손동작의 변화를 0.01mm까지 구별하기 때문에 립모션은 VR 기술과 함께 다양한 분야에 활용되고 있다 [4].

Leap Motion을 활용한 연구는 의료 분야에서도 진행되었다. 수술교육은 장소와 시간의 제약이 심하고 인체해부 실습용 시체를 사용하더라도 실제와 다르기 때문에 어려움이 많다. 또 가상 수술 방법이 존재하지만 장비가 고가이거나 구하는 것이 쉽지 않다는 단점이 존재한다. 이러한 문제점의 해결하기 하고 수술 교육에서 좋은 효과를 얻기 위해 VR과 립모션을 이용한 수술 시스템이 제안되었다 [5].

III. 제안 기법

본 장에서는 해커가 HMD를 착용하고 있는 사용자를 지켜봐도 사용자의 비밀번호를 알아낼 수 없으며 사용자의 편의성을 유지하는 PIN 입력 방법들에 대하여 소개하고자 한다. 또한 제안하는 기법에 대한 구현영상[1]을 Youtube에 업로드하였으며, 구현한 코드[2]는 Github에 올려 Open Source화 하였다.

3.1 립모션을 사용한 PIN 입력 방법

립모션을 사용한 첫 번째 방법으로는 키패드형 PIN 입력 방법이다. 해당 방법에서는 기존의 직사각형 모양의 키패드에서 벗어나 새로운 원 모양의 가상 키패드 입력 방법을 제안하고자 한다. 제안하는 키패드에서 숫자가 주어지는

1)https://www.youtube.com/playlist?list=PLZoOwq4TgNPbW3F7ZJxBYe2HR76bwWXzz
2)https://github.com/amdjd/VR-Pin-Input

것은 기존 키패드와 동일하게 0부터 9까지의 숫자가 주어지지만 기존의 배치와는 다르게 10개의 숫자 버튼이 원 모양으로 구성이 된다. 초기에 배치되는 위치도 랜덤으로 배치되고 구현된 모습은 fig. 1의 오른쪽과 같다. 처음부터 랜덤한 위치에 숫자가 배치되기 때문에 사용자의 입력 패턴을 관찰하더라도 해커의 shoulder surfing attack는 성공할 수 없다.

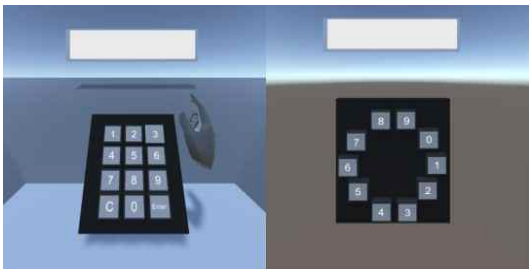


Fig. 1. Left) Traditional keypad Right) keypad to suggest

만약 립모션을 통해 사용자가 입력에 성공하면 입력 이벤트가 발생하여 키패드 위에 위치한 패널에 '*' 모양으로 표시된다. 또한 키패드의 위치는 사용자가 클릭한 지점을 중심으로 다시 재배치되며 숫자들도 랜덤하게 회전하여 배치가 된다. 입력이 발생하여도 0부터 9까지 증가하는 패턴은 변하지 않기 때문에 사용자의 편의성을 유지하면서 사용이 가능하다.

두 번째 제안하는 방법으로는 현실의 물체인 자물쇠를 구현하여 사용자와 상호작용하는 방법이다. 해당 연구에서는 Unity 3D를 통하여 직접 자물쇠 모양의 오브젝트 fig. 2와 같이 구현하였다. 구현된 자물쇠 오브젝트에서 초기 자물쇠에 중앙선에 위치하는 숫자들은 랜덤한 숫자들이 위치하게 된다.

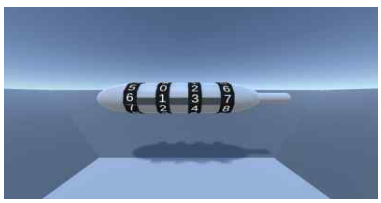


Fig. 2. Implementation of lock object

사용자는 립모션을 사용하여 실제로 자물쇠를 돌리는 것처럼 손가락으로 숫자 버튼을 위로 돌리면 아래로 돌리는 방법으로 중앙선에 위치하는 숫자들이 증가시키고 감소시키는 형태로 사용이 가능하다. 사용자가 돌려서 설정한 자물쇠의 비밀번호가 인증을 위해 등록한 비밀번호와 일치하지 않을 경우 자물쇠는 잠금 상태가 유지되며 비밀번호가 일치할 경우에는 자물쇠의 잠금이 풀리게 되어 인증에 성공한 것을 알 수있다.

3.1 스마트 글러브를 사용한 PIN 입력 방법

본 논문에서는 자체적으로 제작한 스마트 글러브를 사용하였다. 스마트 글러브는 관성측정장치(Inertial Measurement Unit)와 가변저항방법의 구부림 센서를 사용하여 손 제스처 데이터를 전송하는 장갑형태의 입력장치이다. 구현된 스마트 글러브의 모습은 fig. 3과 같다.

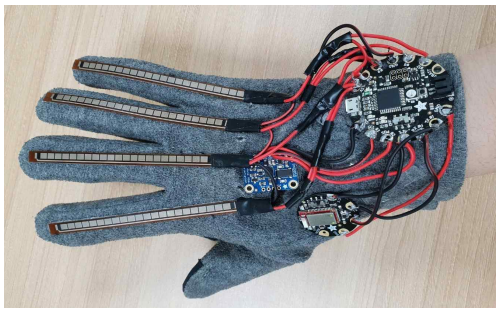


Fig. 3. Smart gloves implemented using multiple sensors

다음으로 스마트 글러브를 사용하여 안전하게 PIN을 입력하는 방법에 대해 설명하도록 한다. 사용자는 초기 화면에서 0에서 9까지의 값을 가지고 이 값들이 랜덤하게 위치하고 있는 4개의 원모양의 키패드를 볼 수 있다. 해당 화면에서 11시에서 1시 방향에 위치한 투명한 부채꼴 모양 안에 위치한 숫자들은 사용자가 선택한 비밀번호가 된다.

스마트 글러브의 사용 방법은 다음과 같다. 먼저 사용자는 구부림 센서를 통하여 어느 키패드를 움직일 것인지 선택이 가능하다. 다음으로 손가락을 구부린 상태에서 IMU 센서를 사

용하여 부채꼴 모양에 위치한 숫자를 변경할 수 있다. 사용자가 볼 수 있는 화면의 초기 모습은 fig. 4의 right와 같으며 손을 시계 방향이나 반시계 방향으로 회전시켜 사용이 가능하다.

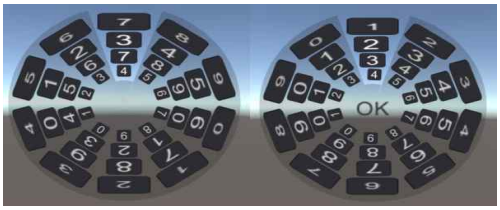


Fig. 4. Left) Initial Run Screen Right) When user authentication is completed

IV. 결론

기존의 PIN 입력 방법들은 해커가 shoulder surfing attack를 통해 사용자의 패턴을 파악하면 비밀번호가 쉽게 노출된다는 취약점을 가지고 있었다. 언급한 취약점을 보완하기 위해 본 논문에서는 립모션과 Unity 3D를 활용하여 기존의 직사각형 모양에서 벗어난 새로운 형태의 가상 키패드를 최초로 구현하였고 보안성이 있는 자물쇠 오브젝트를 구현하여 사용자와 직접 상호작용이 가능하도록 구현하였다. 마지막으로는 여러 센서와 아두이노를 활용한 스마트 글러브를 구현하여 총 세 가지의 VR 상에서의 안전한 PIN 입력 방법에 대하여 제안하였다. 본 논문을 시작으로 VR 상에서의 보안에 대한 연구가 활발히 진행되기를 기대한다.

[참고문헌]

[1] Eun-jee Song, "A Study on Training System for Fire Prevention based on Virtual Reality," *Journal of Digital Contents Society*, 17(3), pp. 189-195, June 2016

[2] Yoo-kyung Chung, "Development of VR Fire-extinguishing Esperience Education Contents Using UX Design Methodology," *Journal of the Korea contents association*, 17(3), pp. 222-230, March 2017

[3] Yang-min Lee, Jo-an Park, Sang-ho Lee, Seon-je Kim and Jae-kee Lee, "Development of Anxiety Measuring App and VR System for Panic Disorder Exposure Training," *KIISE Transactions on Computing Practices*, 24(5), pp. 227-233, May 2018

[4] Yeong-hun Kong and Won-chang Lee, "Motion Control System for a Robotic Manipulator Using Leap Motion," *Journal of Korean Institute of Information Technology*, 14(12), pp. 1-6, Dec 2016

[5] Byung-hoon Kang, Ji-sook Kim and Han-woong Kim, "Study for Operation Teaching Machine Using 3D Virtual Reality System," *Journal of Digital Contents Society*, 17(4), pp. 287-293, Aug 2016

VR 상에서의 안전한 PIN 입력 방법 제안

우재민 · 김현준 · 김도영 · 서화정*
*한성대학교 IT융합공학부

Proposal of Safe PIN Input Method on VR

Jae-Min Woo · Hyunjun Kim · Doyoung Kim · Hwajeong Seo*
*Department of IT, Hansung University, Seoul, 02876 Korea

요 약

가상현실 속에서 실제와 같은 서비스를 제공하는 기술인 VR (Virtural Reality)는 실제와 유사한 공간적 체험을 할 수 있도록 Head Mounted Display (HMD) 기기를 이용하여 영상과 음성의 출력 그리고 사용자 입력을 제공한다. 현재 VR을 활용한 많은 서비스들이 개인화되어 있는 만큼 사용자에 대한 안전한 인증이 매우 중요하다. 하지만 VR 기기 상에서 제공하는 Personal Identification Number (PIN)은 해커가 Shoulder Surfing Attack (SSA)을 이용하여 사용자의 입력 패턴을 분석하게 될 경우 비밀번호가 그대로 노출되는 문제점을 가지고 있다. 본 논문에서는 사용자의 편의성은 그대로 유지하면서 해커가 입력 패턴을 분석하더라도 사용자의 비밀 번호를 안전하게 보호할 수 있는 방법에 대해 제안한다. 이는 립모션을 활용한 두 가지 입력 방법과 여러 센서와 아두이노를 활용하여 구현된 스마트 글러브를 이용하여 입력하는 방법으로 총 세 가지의 새로운 형태의 안전한 PIN 입력 방법을 포함한다.

I. 서론

최근 스마트폰 시장이 포화 상태에 도달함에 따라 IT기업들은 새로운 기술을 발굴하는게 큰 화두가 되었다. 이에 IT기업들이 주목하는 기술이 가상현실이다. 가상현실이란 컴퓨터 등을 사용한 인공적인 기술로 만들어낸 실제와 유사하지만 실제가 아닌 어떤 특정한 환경이나 기술 그 자체를 의미한다 [1]. 사용자가 Head Mounted Display(HMD) 기기를 사용한다면 HMD 기기를 통해 가상 현실 속에서 음성과 영상을 제공받아 여러 컨텐츠 서비스를 이용할 때 더욱 몰입감 있게 체험할 수 있다. VR은 여러 분야에서 활용이 가능한데 크게 게임, 교육, 의료, 영상 등 분야에 활용이 되고 있다.

VR이 여러 분야에 활용이 되면서 사용자에게 많은 서비스가 제공이 되고 개인화가 이루어 지면서 이제는 VR 상에서 사용자에 대한 인증이 매우 중요해졌다. 현재 VR 상에서 제공하는 Personal Identification Number(PIN)를 사용하게 되면 VR의 특징상 HMD 기기를 착용을 하면 주변 환경을 인식하지 못하는 취약점으로 인해 해커는 사용자의 어깨너머로 훔쳐보는 공격인 Shoulder Surfing Attack (SSA)를 수행하여 사용자의 PIN 입력 패턴을 분석하게 되면 사용자의 PIN은 고스란히 노출이 된다. 따라서 본 논문에서는 립모션과 스마트 글러브를 활용하여 해커의 shoulder surfing attack을 통한 입력 패턴 분석을 방지하면서 사용자의 편의성을 보장하는 새로운 형태의 PIN 입력 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 VR과 립모션의 관련 연구 동향에 대해 살펴본다. 3장에서는 안전한 PIN 입력을 위해 립모션과 스마트 글러브를 활용한 제안 기법에 대해 설명을 진행하며 4장에서는 본 논문의 결론을 맺도록 한다.

II. 관련 연구 동향

본 장에서는 제안하고자 PIN 입력 방법을 설명하기 전에 VR, 립모션에 대해 소개하고 간단하게 관련 연구 동향에 대해서 알아보도록 한다.

2.1 Virtual Reality(VR)

최근 VR을 체험할 수 있는 HMD 기기의 성능이 향상되면서 VR에 대한 관심이 증가하고 여러 분야에서 VR을 활용한 다양한 연구가 진행이 되고 있다. 교육 분야에서는 VR을 사용하여 다양한 교육용 시뮬레이션이 제안되고 있다. 소방안전과 같이 사실적인 체험이 필요하지만, 현장에서 경험할 수 없는 한계를 가지고 있으며 사고가 발생할 수 있는 위험성이 있기 때문에 VR을 사용하여 가상의 사고 현장을 구현하고 사용자는 구현된 가상의 사고 현장에서 오감을 이용하여 능동적인 학습자가 될 수 있다 [2].

의료 분야에서도 VR을 활용하여 환자들의 치료를 위한 다양한 연구가 진행되고 있다. 특정 장소나 상황을 싫어하는 증상인 공황장애는 보통 노출훈련이라는 치료 방식을 적용한다. 하지만 노출훈련은 비용, 인력, 시간이 많이 소모되기 때문에 공황장애 환자 및 공황장애 환자일 가능성이 있는 사람에게 앱을 통해 불안지수를 측정하고 노출훈련의 문제점을 해결한 VR 기술을 적용한 노출훈련 시스템을 통하여 환자들의 공황장애를 치료하는 방안이 제안되었다 [3].

이처럼 현재 VR은 교육, 의료 분야에서 활발하게 연구되고 있으며 해당 분야가 아니더라도 구현된 오브젝트와 상호작용이 가능하다는 장점은 다양한 분야에서 적용이 될 수 있기 때문에 가상현실에 대한 연구는 더욱 활발해질 것이다. 하지만 가상현실에서의 보안에 대한 연구는 아직 다른 분야에 비해 미흡하다. 현재 가상현실 기기와 게임 시스템의 정보보증을 위해 보안 위협을 식별하였고 이를 보호하는 방법이 제안되었지만 [7], 가상현실 기술이 보편화되기 위해서는 보안에 대한 연구는 앞으로 더욱 더 활발하게 진행되어야 할 것이다.

2.2 립모션

NUI 장치의 일종인 립모션은 사용자의 손동작을 3D 정보로 얻어 컴퓨터의 입력으로 활용할 수 있는 대표적인 제품이다. 초 당 200번씩 사용자의 동작을 인식할 수 있고 손동작의 변화를 0.01mm까지 구별하기 때문에 립모션은 VR 기술과 함께 다양한 분야에 활용되고 있다 [4].

Leap Motion을 활용한 연구는 의료 분야에서도 진행되었다. 수술교육은 장소와 시간의 제약이 심하고 인체해부 실습용 시체를 사용하더라도 실제와 다르기 때문에 어려움이 많다. 또 가상 수술 방법이 존재하지만 장비가 고가이거나 구하는 것이 쉽지 않다는 단점이 존재한다. 이러한 문제점의 해결하기 하고 수술 교육에서 좋은 효과를 얻기 위해 VR과 립모션을 이용한 수술 시스템이 제안되었다 [5].

III. 제안 기법

본 장에서는 해커가 HMD를 착용하고 있는 사용자를 지켜봐도 사용자의 비밀번호를 알아낼 수 없으며 사용자의 편의성을 유지하는 PIN 입력 방법들에 대하여 소개하고자 한다. 또한 제안하는 기법에 대한 구현영상[1]을 Youtube에 업로드하였으며, 구현한 코드[2]는 Github에 올려 Open Source화 하였다.

3.1 립모션을 사용한 PIN 입력 방법

립모션을 사용한 첫 번째 방법으로는 키패드형 PIN 입력 방법이다. 해당 방법에서는 기존의 직사각형 모양의 키패드에서 벗어나 새로운 원 모양의 가상 키패드 입력 방법을 제안하고자 한다. 제안하는 키패드에서 숫자가 주어지는

1)https://www.youtube.com/playlist?list=PLZoOwq4TgNPbW3F7ZJxBYe2HR76bwWXzz
2)https://github.com/amdjd/VR-Pin-Input

것은 기존 키패드와 동일하게 0부터 9까지의 숫자가 주어지지만 기존의 배치와는 다르게 10개의 숫자 버튼이 원 모양으로 구성이 된다. 초기에 배치되는 위치도 랜덤으로 배치되고 구현된 모습은 fig. 1의 오른쪽과 같다. 처음부터 랜덤한 위치에 숫자가 배치되기 때문에 사용자의 입력 패턴을 관찰하더라도 해커의 shoulder surfing attack는 성공할 수 없다.

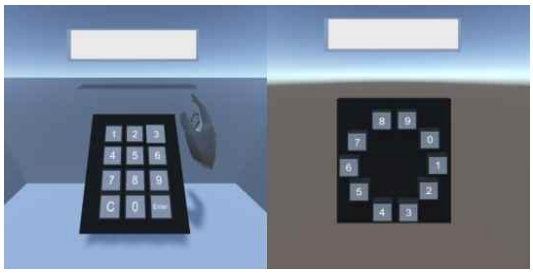


Fig. 1. Left) Traditional keypad Right) keypad to suggest

만약 립모션을 통해 사용자가 입력에 성공하면 입력 이벤트가 발생하여 키패드 위에 위치한 패널에 '*' 모양으로 표시된다. 또한 키패드의 위치는 사용자가 클릭한 지점을 중심으로 다시 재배치되며 숫자들도 랜덤하게 회전하여 배치가 된다. 입력이 발생하여도 0부터 9까지 증가하는 패턴은 변하지 않기 때문에 사용자의 편의성을 유지하면서 사용이 가능하다.

두 번째 제안하는 방법으로는 현실의 물체인 자물쇠를 구현하여 사용자와 상호작용하는 방법이다. 해당 연구에서는 Unity 3D를 통하여 직접 자물쇠 모양의 오브젝트 fig. 2와 같이 구현하였다. 구현된 자물쇠 오브젝트에서 초기 자물쇠에 중앙선에 위치하는 숫자들은 랜덤한 숫자들이 위치하게 된다.

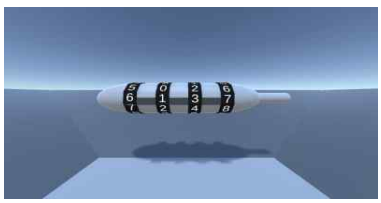


Fig. 2. Implementation of lock object

사용자는 립모션을 사용하여 실제로 자물쇠를 돌리는 것처럼 손가락으로 숫자 버튼을 위로 돌리면 아래로 돌리는 방법으로 중앙선에 위치하는 숫자들이 증가시키고 감소시키는 형태로 사용이 가능하다. 사용자가 돌려서 설정한 자물쇠의 비밀번호가 인증을 위해 등록한 비밀번호와 일치하지 않을 경우 자물쇠는 잠금 상태가 유지되며 비밀번호가 일치할 경우에는 자물쇠의 잠금이 풀리게 되어 인증에 성공한 것을 알 수있다.

3.1 스마트 글러브를 사용한 PIN 입력 방법

본 논문에서는 자체적으로 제작한 스마트 글러브를 사용하였다. 스마트 글러브는 관성측정장치(Inertial Measurement Unit)와 가변저항방법의 구부림 센서를 사용하여 손 제스처 데이터를 전송하는 장갑형태의 입력장치이다. 구현된 스마트 글러브의 모습은 fig. 3과 같다.

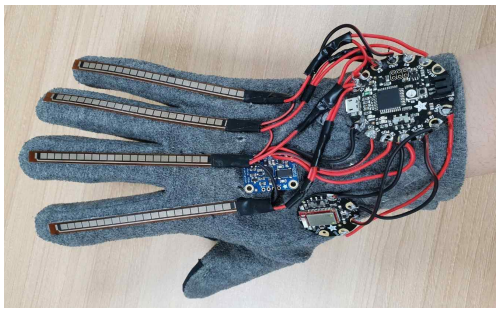


Fig. 3. Smart gloves implemented using multiple sensors

다음으로 스마트 글러브를 사용하여 안전하게 PIN을 입력하는 방법에 대해 설명하도록 한다. 사용자는 초기 화면에서 0에서 9까지의 값을 가지고 이 값들이 랜덤하게 위치하고 있는 4개의 원모양의 키패드를 볼 수 있다. 해당 화면에서 11시에서 1시 방향에 위치한 투명한 부채꼴 모양 안에 위치한 숫자들은 사용자가 선택한 비밀번호가 된다.

스마트 글러브의 사용 방법은 다음과 같다. 먼저 사용자는 구부림 센서를 통하여 어느 키패드를 움직일 것인지 선택이 가능하다. 다음으로 손가락을 구부린 상태에서 IMU 센서를 사

용하여 부채꼴 모양에 위치한 숫자를 변경할 수 있다. 사용자가 볼 수 있는 화면의 초기 모습은 fig. 4의 right와 같으며 손을 시계 방향이나 반시계 방향으로 회전시켜 사용이 가능하다.

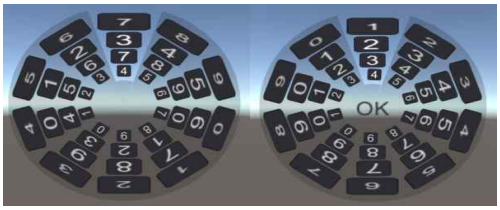


Fig. 4. Left) Initial Run Screen Right) When user authentication is completed

IV. 결론

기존의 PIN 입력 방법들은 해커가 shoulder surfing attack를 통해 사용자의 패턴을 파악하면 비밀번호가 쉽게 노출된다는 취약점을 가지고 있었다. 언급한 취약점을 보완하기 위해 본 논문에서는 립모션과 Unity 3D를 활용하여 기존의 직사각형 모양에서 벗어난 새로운 형태의 가상 키패드를 최초로 구현하였고 보안성이 있는 자물쇠 오브젝트를 구현하여 사용자와 직접 상호작용이 가능하도록 구현하였다. 마지막으로는 여러 센서와 아두이노를 활용한 스마트 글러브를 구현하여 총 세 가지의 VR 상에서의 안전한 PIN 입력 방법에 대하여 제안하였다. 본 논문을 시작으로 VR 상에서의 보안에 대한 연구가 활발히 진행되기를 기대한다.

[참고문헌]

[1] Eun-jee Song, "A Study on Training System for Fire Prevention based on Virtual Reality," *Journal of Digital Contents Society*, 17(3), pp. 189-195, June 2016

[2] Yoo-kyung Chung, "Development of VR Fire-extinguishing Esperience Education Contents Using UX Design Methodology," *Journal of the Korea contents association*, 17(3), pp. 222-230, March 2017

[3] Yang-min Lee, Jo-an Park, Sang-ho Lee, Seon-je Kim and Jae-kee Lee, "Development of Anxiety Measuring App and VR System for Panic Disorder Exposure Training," *KIISE Transactions on Computing Practices*, 24(5), pp. 227-233, May 2018

[4] Yeong-hun Kong and Won-chang Lee, "Motion Control System for a Robotic Manipulator Using Leap Motion," *Journal of Korean Institute of Information Technology*, 14(12), pp. 1-6, Dec 2016

[5] Byung-hoon Kang, Ji-sook Kim and Han-woong Kim, "Study for Operation Teaching Machine Using 3D Virtual Reality System," *Journal of Digital Contents Society*, 17(4), pp. 287-293, Aug 2016

클라우드 환경에서의 스마트카드를 활용한 개체 인증 개선 방안

조정석*, 곽진**

*아주대학교 컴퓨터공학과 정보보호응용및보증연구실

**아주대학교 사이버보안학과

Improvement of Entity Authentication Using Smartcard in Cloud Environment

Jeong-Seok Jo*, Jin Kwak**

*ISAA Lab., Department of Computer Engineering, Ajou University.

**Department of Cyber Security, Ajou University.

요 약

클라우드 서비스는 다양한 서비스 제공이 목적이며, 기존의 네트워크와 달리 대량의 자원이 클라우드에 존재하게 된다. 이에 따라 하나의 위협으로 인해 전체의 클라우드 자원에 영향을 끼칠 수 있으므로, 높은 보안성이 요구된다. 이를 위해 클라우드 서버에서의 인증 방안 중 스마트카드를 활용한 다중 요소 인증 방식이 많이 사용되고 있으며, Rohan등은 스마트카드를 활용한 인증을 제시하였다. 하지만 스마트카드를 대상으로 한 공격이 다수 발생하고 있으며, 감염된 스마트카드를 활용하여 제 3자가 권한이 없는 클라우드 자원에 접근하는 문제들이 발생하고 있다.

이에 따라 본 논문에서는 클라우드 환경에서의 스마트카드를 활용한 인증의 개선 방안을 제시하고자 한다.

I. 서론

클라우드 서비스가 등장하면서, 클라우드 환경에서 인증을 위한 다양한 방안들이 등장하였다. Rohan등은 Sarang V. Hartwar의 Multi-factor 인증을 활용하여[1] 클라우드 환경에서의 Multi-factor 방식 스마트카드 인증 프로토콜을 제시하였다[2]. 그러나 Rohan등이 제시한 프로토콜에서 사용자의 PC에 맬웨어를 감염시키고 백도어를 통해 공격자가 원하는 서버에 접근하는 취약점이 존재한다. 이에 따라, 해당 취약점을 분석하고 이에 대한 대응 방안을 제시하고자 한다.

본 논문에서는 2장에서 sykipot 맬웨어 분석 및 Rohan등이 제시한 스마트카드를 활용한 인증 프로토콜과 해당 프로토콜의 취약점을 분석하였으며, 3장에서 해당 취약점의 대응하는 개선된 인증 프로토콜을 제시하고, 4장에서 Rohan등이 제시한 프로토콜과 개선된 프로토콜을 비교 분석하고, 5장에서 결론을 맺는다.

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1E1A1A01075110).

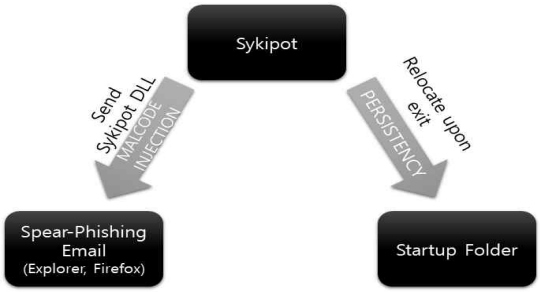
II. 관련연구

2.1 sykipot 맬웨어

sykipot 맬웨어는 스마트카드를 대상으로 동작하는 맬웨어이다. 해당 웨어는 sykipot EXE, sykipot DLL로 구성되어있다. sykipot EXE는 Sykipot DLL이 암호화되지 않은 상태로 내장된 실행파일이며, 동작방식은 사용자가 스피어 피싱 전자 메일에서 악성 링크 혹은 첨부 파일을 열면 Sykipot EXE가 저장되고 실행되는 방식이다. Sykipot의 동작방식은 (그림 1)과 같다.

해당 맬웨어는 키 로깅을 이용해 정보를 얻

고, 백도어를 활용해 공격자가 원하는 서버로 연결한다[3]. 본 논문에서는 백도어를 이용하여 발생 가능한 취약점을 제시하고, 그에 대한 개선 방안을 제시하고자 한다.

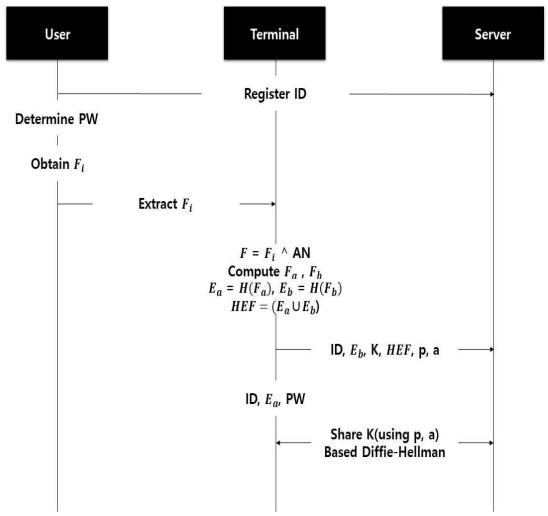


(그림 1) Sykipot 동작방식

2.2 Rohan's Proposed Protocol

Rohan등은 클라우드 환경에서의 스마트카드를 활용한 인증 프로토콜을 새롭게 제안하였다[2]. 제안한 프로토콜의 동작방식은 (그림 2)와 같다.

본 절에서는 Rohan등이 제안한 프로토콜을 분석하고, sykipot 맬웨어를 활용한 취약점을 분석한다.



(그림 2) Rohan's Proposed Protocol

2.2.1 Rohan's Proposed Protocol Parameter

Rohan등이 제시한 프로토콜에서 사용되는 파라미터는 (표 1)과 같다.

대부분의 연산은 Terminal에서 이루어진다.

(표 1) Rohan's Proposed Protocol Parameter

Parameter	Contents
ID, PW	- 사용자의 ID/PW
F_i	- 사용자의 지문
AN	- 고유한 비밀 값
F	- F_i, AN의 XOR 값
F_a, F_b	- F의 짝수, 홀수 비트열
E_a, E_b	- F_a, F_b의 Hash 연산 값
K	- AES 키
HEF	- $E_a \cup E_a$ 의 Hash 연산 값
p	- 임의의 큰 소수
a	- Root A

2.2.2 프로토콜 동작 순서

Rohan등이 제안한 프로토콜 동작 순서는 아래와 같다.

Step 1. 비밀 값 생성

프로토콜에 사용될 파라미터(a, p, K, AN)을 생성한다.

Step 2. ID 등록(User→Server)

User는 Server에 본인의 ID를 등록한다.

Step 3. PW 결정(User)

User는 인증에 사용할 PW를 설정하고 스마트카드에 저장한다.

Step 4. 지문 정보 확보 및 전송(User→Terminal)

User는 본인의 지문 정보를 확보하고 Terminal은 Orientation Maps(OM)을 활용하여, 확보된 지문정보로부터 값을 추출한다.

Step 5. 파라미터 형성(Terminal)

Terminal은 User로부터 받은 지문정보를 활용해 인증에 활용될 파라미터들을 확보한다. 이 때, (표 1)의 F, F_a, F_b, E_a, E_b, HEF 가 생성된다.

Step 6. 파라미터 전송(Terminal→Server)

Terminal은 확보된 파라미터 중 ID, E_b, K, HEF, p, a 값들을 전송한다.

Step 7. 파라미터 저장(Terminal)

Terminal은 ID, PW, E_a 값들을 저장한다.

Step 8. 키 공유(Terminal↔Server)

Sever와 Terminal은 p, a 기반 Diffie-Hellman 키 교환 방식을 활용하여 AES 키 K를 공유한다.

Step 9. 인증(User↔Server)

User에게 받은 실시간 지문 정보를 이용하여 F를 추출하고, 그에 대한 E_a, E_b, HEF 값 정당성을 비교하여 인증을 수행한다.

2.2.3 sykipot 맬웨어 기반 프로토콜 취약점

Rohan등이 제안한 프로토콜은 Terminal에 대한 인증을 진행하지 않고, 지문 정보 및 스마트카드에 등록된 정보를 보낸다.

이에 따라, 공격자가 2.1절에서 분석한 sykipot 맬웨어를 활용하여, 스마트카드에 접근하고 백도어를 이용해 공격자가 원하는 서버와 연결하여, 정당한 Terminal인 척 위장하는 공격이 가능하다. 공격 진행 과정은 아래와 같다.

Step 1. 맬웨어 전송(Attacker→User)

Attacker는 User에게 sykipot 맬웨어를 삽입한 전자 메일을 보낸다.

Step 2. 맬웨어 삽입(User)

User는 수신한 전자 메일의 첨부 파일을 다운로드 받게 되고 그에 따라 sykipot 맬웨어에 감염되게 된다.

Step 3. 백도어 연결(Attacker→User)

sykipot 맬웨어 감염으로 인해 User의 인증을 위한 Terminal은 Attacker가 원하는 서버가 된다.

Step 4. Rohan 프로토콜 수행(User→Attacker)

User는 Terminal에 대한 인증 작업을 수행하지 않고 Rohan등이 제시한 프로토콜을 수행한다.

Step 5. 인증 파라미터 확보(Attacker)

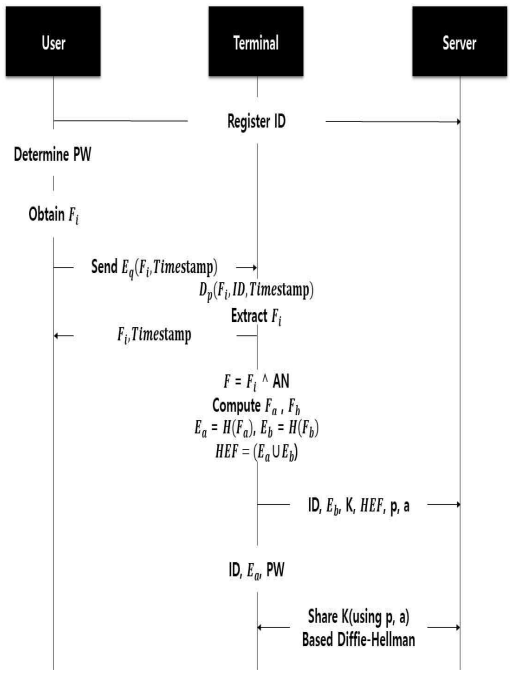
Attacker는 *Step 1.*~*Step 4.*과정에서 확보한 파라미터를 이용하여 서버와의 인증을 수행한다. 그 결과, 클라우드 자원에 접근 가능하다.

III. 제안 사항

3.1 개선된 인증 방안

2.2.3절에서 제시한 취약점에 대응하기 위한 개선된 인증 프로토콜은 (그림 3)과 같다.

본 절에서 제안하는 프로토콜은 기존에 생성된 난수 p를 활용한 공개키 암호화를 이용해 Terminal 인증을 수행하고 지문 정보 재전송 공격 방지를 위한 Timestamp를 활용한다.



(그림 3) Proposed Protocol

3.1.1 제안 방안에서 사용되는 파라미터

본 절에서 제안하는 프로토콜에서 추가로 사용되는 파라미터는 (표 2)와 같다.

(표 2) Proposed Protocol Parameter

Parameter	Contents
q	- p과 함께 사용할 공개키
Timestamp	- 재전송 공격을 방지하기 위한 값

3.1.2 제안 프로토콜 동작 순서

Step 1. 비밀 값 생성

프로토콜에 사용될 파라미터(a, p, K, AN)을 생성한다.

Step 2. ID 등록(User→Server)

User는 Server에게 본인의 ID를 등록한다.

Step 3. PW 결정(User)

User는 인증에 사용할 PW를 설정하고 스마트카드에 저장한다.

Step 4. 지문 정보 확보 및 암호화 전송

(User→Terminal)

User는 본인의 지문 정보를 확보하여 확보한 지문 정보를 User의 Timestamp와 함께 가지고 있는 난수 q를 이용해 암호화하여 보낸다.

Step 5. Terminal 인증(Terminal→User)

Terminal은 *Step 4.*에서 수신한 값을 난수 p를 이용해 복호화하여, User에게 보낸다. 또한 확보한 지문 정보로부터 Orientation Maps(OM)을 활용하여, 지문 값을 추출한다.

Step 6. Terminal 검증(User)

User는 수신한 값들을 이용해 Terminal이 정당한지 검증한다.

이후의 동작 순서는 2.2.2절의 *Step 5.*~*Step 8.*와 동일하다.

IV. 안전성 분석

본 논문에서 제시한 프로토콜에 대하여 sykipot 맬웨어 기반 프로토콜 취약점 공격을 수행한 과정은 아래와 같다.

Step 1.~*Step 4.*과정은 3.1.2절과 동일하다.

Step 5. Terminal 인증(Terminal→User)

이 때 기존의 Terminal은 복호화가 불가능하기 때문에 User의 Terminal 인증을 수행할 수 없다. 따라서, sykipot 맬웨어 취약점을 이용한 공격을 방지할 수 있다.

또한, Terminal 인증을 통과하더라도 공격자는 등록된 당시의 지문 정보를 가지고 있기 때문에 Timestamp와 함께 전송함으로써 지문 정보의 재전송을 방지할 수 있다.

V. 결론

본 논문을 통해 Rohan등이 제시한 클라우드 환경에서의 스마트카드를 이용한 인증 프로토콜을 분석하고, 스마트카드 대상으로 동작하는 맬웨어를 분석하였다. 또한, 해당 맬웨어를 이용한 공격 시나리오를 제시하고 기존 프로토콜의 취약점 개선을 위한 인증 프로토콜을 제시하였다.

개선된 인증 프로토콜은 Terminal에 대한 인증 없이 인증 프로토콜을 수행하는 Rohan등의 인증 프로토콜에 추가적으로 난수 q와 Timestamp를 활용하여 sykipot 맬웨어와 재전송 공격을 방지하고, Terminal에 대한 인증 과정을 추가하였다.

향후 클라우드 환경에서의 경량화 인증 프로토콜에 대해 연구를 수행할 예정이다.

[참고문헌]

[1] Sarang V. Hatwar, "Cloud Computing Security Aspects, Vulnerabilities and Countermeasures", Jun. 2015

[2] Rohan H. Shah, Prof. D. P. Salapurkar, "A multifactor authentication system using secret splitting in the perspective of Cloud of Things", Feb. 2017

[3] SANS Institute InfoSec Reading Room, "Detailed Analysis Of Sykipot"

PwnETH: 이더리움 스마트 컨트랙트 취약점 공격코드 라이브러리

이진우[1], 김경민[2], 정윤서[3], 이진경[4], 강연호[5], 김규헌[2], 이상섭[6], 김경곤[7]

[1]서울여자대학교 정보보호학과 [2]고려대학교 사이버국방학과 [3]한국디지털미디어고등학교
[4]선린인터넷고등학교 [5]거제제일고등학교 [6]삼성리서치 [7]고려대학교

PwnETH: Ethereum Smart Contract Exploit Library

Jin-Woo Lee[1], Kyoung-Min Kim[2], Yun-Seo Jung[3], Jin-Kyung Lee[4],

Yeon-Ho Kang[5], Kyu-Heon Kim[2], Sang-Sup Lee[6], Kyoung-Gon Kim[7]

[1]Sunrin Internet High School. [2]Department of Cyber Defense, Korea University.

[3]Korea Digital Media High School. [4]Department of Information Security, Seoul Women's University. [5]Geoje Jeil High School.

[6]Samsung Research. [7]Korea University.

요 약

이더리움은 web3라는 HTTP기반 JSON RPC API를 이용해 이더리움 블록체인과 통신하고 스마트 컨트랙트 배포 및 컨트랙트 함수를 호출하는 등 트랜잭션을 처리한다. 하지만 web3에서 트랜잭션을 생성하거나 지갑을 연동하는 등의 환경을 구성하는 것이 복잡하다. 또한 JavaScript는 비동기적으로 동작하는데 공격자가 정확한 공격코드를 위해서 데이터를 수집할 때 동기적으로 처리해야 할 때가 있다. 이러한 문제점을 해결하고 공격자의 관점에서 익스플로잇을 편리하게 할 수 있는 도구인 PwnETH를 소개한다. 해당 논문은 PwnETH가 web3에서 발생하는 문제점을 어떻게 해결했는지와 해당 라이브러리를 통해 취약한 스마트 컨트랙트의 공격코드를 작성하는 방안을 보여준다.

I. 서론

2008년 사토시 나카모토가 발표한 보고서에서 등장한 블록체인이라는 개념은[1], 네트워크 참여자가 생산하는 정보 및 가치를 공동으로 기록하고 저장하는 분산화된 공개 장부 관리 기술을 의미한다. 블록체인 내 참여한 모든 구성원이 네트워크를 통해 서로 데이터를 검증하고 저장함으로써 특정인의 임의적인 조작이 어렵도록 설계된 저장 플랫폼이다.

그동안 블록체인을 이중지불문제를 해결한 암호화폐의 거래에만 활용하였지만 블록체인을 다양한 기능으로 활용하고, 확장성을 해결하기 위해 이더리움이 나타났다[2]. 이더리움은 블록체인의 합의 프로토콜에 의해 실행되는 프로그램인 스마트 컨트랙트를 구동시킨다는 점이 가장 큰 특징이다. 스마트 컨트랙트는 1994년 Nick Szabo에 의해 고안되었으며, 이의 목적은 신뢰할 수 없는 컴퓨터 네트워크 환경에서 고도로 발달된 자동 계약 이행 방법을 제시하는 것이다. 스마트 컨트랙트에서 주어진 계약 조건은 프로그램화 되어 자동적으로 수행되면서 계약 조건이 만족되면 예외 없이 강제적으로 실행이 된다.

이러한 스마트 컨트랙트는 블록체인의 신뢰할 수 있는 탈중앙화된 특징 때문에 이더리움을 통해 본격적으로 사용될 수 있었다. 하지만 이러한 스마트 컨트랙트는 주체자가 없으며 코드에 취약점이 있을 경우 수정할 수 없는 불변성, 그리고 암호화폐라는 금전적인 요소가 들어가기 때문에 공격자에게 좋은 타겟이 되었다. 따라서 스마트 컨트랙트 보안의 필요성이 대두되고 있다. 보안을 위해 스마트 컨트랙트의 공격코드를 작성하고 검증하는 일이 필요해졌다. 따라서 보안에 특화된 스마트 컨트랙트 분석 라이브러리 개발하였다.

본 논문에서는 취약점 공격코드 라이브러리 pwneth를 소개하고, 기존에 이더리움에서 제공하는 공식 구현체 go ethereum, web3와 비교하였을 때 장점들을 설명한다. 또한, 취약한 스마트 컨트랙트들의 예시를 소개하고, PwnETH를 통해 작성한 공격코드를 예시로 소개한다.

II. 배경 및 선행연구

2.1 스마트 컨트랙트의 취약점들

현재 다양한 블록체인 보안 연구 회사들이 취약점들을 분류하고 있다[3]. 가장 대표적인 버그 클래스 분류는 2018년 스마트 컨트랙트 공동체를 위한 보안프로젝트 중 하나인, NCC그룹의 Decentralized Application Security Project (DASP)이다. 해당 프로젝트에서는 스마트 컨트랙트의 대표적인 9가지 취약점을 분류하였다.[4]

- Re-entrancy (재귀 호출 취약점)
- Access Control (접근 제어 취약점)
- Arithmetic Issues (연산 오류)
- Unchecked Return Values For Low Level Calls (저수준 호출의 반환값 문제)
- Denial of Service (DoS)
- Bad Randomness (취약한 난수 활용)
- Front-Running (트랜잭션 순서 문제)
- Time Manipulation (채굴자의 시간 조작)
- Short Address Attack (잘못된 주소 길이 활용으로 발생하는 취약점)

이러한 취약점들의 유형들을 파악하고 라이브러리에서 활용할 수 있는 기능들을 고민하였다.

2.2 이더리움의 구현체 Geth와 web3

Go Ethereum(Geth)은 JSON RPC, IPC, WebSocket을 지원한다. 기존에는 Geth를 이용하여 트랜잭션을 발생시켰지만 좀 더 편리하게 JSON RPC 라이브러리를 사용할 수 있도록 web3 라이브러리가 존재하게 되었다. 스마트 컨트랙트에서 이벤트를 처리하거나 반환되는 자료형을 처리하는데 web3 라이브러리는 유용하다. 단위 변환 유틸리티 클래스 등 다양한 기능이 있다.

Go ethereum에서 동작하고 있는 이더리움 프라이빗 네트워크에 스마트 컨트랙트를 배포하고, 배포된 스마트 컨트랙트와 상호작용하기 위해선 web3가 필요하다. 이러한 web3를 자바스크립트로 구현한 것이 web3.js, 파이썬으로 구현한 것이 web3.py다.

하지만 자바스크립트에서는 기본적으로 비동기적으로 동작하고 있다. 따라서 먼저 처리해야 할 작업이 오랜시간이 걸린다면 다음작업을 처리하는 문제가 발생할 수 있다. 또한 getStorageAt함수를 통해서 변수에 값을 가져오는데 만약 순서대로 가져오지 않는다면 문제가 발생할 것이다.

Go ethereum에 비해 web3는 개발에 훨씬 유용하며 많이 간결해졌다. 하지만 비동기처리를 동기처리를 바꾸는것 외에 컨트랙트를 배포하는 등 트랜잭션을 처리하는데 불편함이 있다. PwnETH는 이러한 문제를 해결할 뿐만아니라, 스마트 컨트랙트 보안 담당자의 입장에서 컨트랙트를 공격해보고, 보안을 검증하는데 중점을 두고 개발하였다. 예를 들어 PwnETH를 통해서 개발자가 웹에서 설정한 대로 함수를 호출하는 것이 아니라 직접적으로 함수를 호출하고 인자를 자유롭게 설정할 수 있도록 하였다. 또한 이더리움 메인넷에 올라간 블록, 코드 등 다양한 정보를 가져오기 편하게 하였다.

III. PwnETH

PwnETH는 가장 기본적으로 공격자 입장에서 간결한 공격코드를 작성할 수 있게 하기 위해 노력하였다. 원래 web3는 스마트 컨트랙트를 배포하거나, 컨트랙트 함수를 호출하는 등 트랜잭션을 생성할 때 트랜잭션에 대한 nonce, address, gas 등을 직접 기록하여 생성해야된

다. PwnETH에서는 트랜잭션 생성할 때 필요한 정보를 자동으로 기입하여 최소한에 정보만 입력하여 손 쉽게 사용할 수 있다. 또한 Local에서 뿐만 아니라 Mainnet, Ropsten 등 다양한 네트워크 환경에서 접속하는 것에 대한 설정이 불편하였다. infura.io에서 API KEY를 발급받아서 원하는 네트워크에 편하게 접속할 수 있게 개발하였다. 마지막으로 블록체인에 올라간 스마트 컨트랙트에 대한 검증된 코드는 etherscan.io의 API를 이용해 컨트랙트 이름과 코드를 반환하도록 구현하였다.

Bad Randomness 취약점은 스마트 컨트랙트 내에서 의사 난수 함수를 구현할 때 예측가능한 의사 난수를 활용하여 문제가 발생하는 것을 말한다. 이러한 스마트 컨트랙트들은 block에 대한 정보, storage 변수의 값 등을 활용하게 된다. 따라서 PwnETH는 블록 번호, 블록 생성시간(timestamp), nonce 등 모든 정보를 가져올 수 있고, storage에 대한 정보를 Index만 입력해준다면 편하게 가져올 수 있도록 개발하였다.

PwnETH에 큰 장점은 다른 설정없이 함수호출만으로 트랜잭션을 생성하고, 네트워크에 연결하기 때문에 전문적인 지식 없이도 빠르게 사용할 수 있다. 기존에 web3는 개발자 입장에서 만들어졌다면 PwnETH는 해커에 입장으로 공격을 진행하는데 필요한 기능을 추가하여 독창성을 가지고 있다.

아래의 표를 보면 컨트랙트 배포, 트랜잭션 생성 등의 기능들을 PwnETH를 활용함으로서 코드의 길이가 크게 줄어든 것을 확인할 수 있다.

	컨트랙트 배포	트랜잭션 생성	블록 정보
Go ethereum	24	18	
web3(js/py)	19/15	6/10	8/1
PwnETH (js/py)	1/1	1/1	1/1

표 1. 각 기능별 코드 간소화 정도

다음 코드는 CVE-2018-15552 취약한 스마트 컨트랙트 소스이다. 이 스마트 컨트랙트는 티켓을 사서 복권에 당첨 되면 일정 이더리움을 받는 게임이다. 하지만 이 게임에서는 안전하지 않은 난수를 생성하는 Bad Randomness 취약점이 발생한다. numtickets과 maxTickets은 복권에 당첨자를 결정짓는 변수이다. 이 컨트랙트에 개발자는 maxTickets을 private로 선언하였다. 하지만 이는 접근제어자로 변수 값을 수정할 수 없을뿐 web3에 getSotrageAt함수를 통해서 값을 가져올 수 있다.

```
uint256 private maxTickets;
uint256 numtickets;
function AddTicket() public payable {
        bool success = numtickets == maxTickets;
        if(success) {
            PayWinner(msg.sender);
        }
}
function PayWinner( address winner ) private {
        require(numtickets == maxTickets);
        winner.transfer(winnerPrice);
}
```

그림 1. 취약한 스마트 컨트랙트 소스

```
function getStorage(address, idx){
  var ret = undefined;
  web3.eth.getStorageAt(address, idx).then(e=>{ret=e;})
  require('deasync').loopWhile(function(){return ret == undefined})
  return ret
}

var contract = new web3.eth.Contract(contract_abi, contract_address)
console.log('maxTickets : ' + getStorage(contract_address , 0))
console.log('numtickets : ' + getStorage(contract_address , 5))
```

그림 2. web3로 작성한 exploit

```
getStorage('0x1e217ADC6a6ADC16E248af109Ab7EFA4d1bb252D',0)
getStorage('0x1e217ADC6a6ADC16E248af109Ab7EFA4d1bb252D',5)
```

그림3. pwneth를 활용한 exploit

IV. 결론 및 향후 연구

본 논문에서는 공격자의 입장에서 스마트 컨트랙트의 취약점 패턴을 분석하고 공격 코드를 작성하기 쉬운 방법을 제시하기 위한 PwnETH를 소개하였다.

아직 공격을 진행하는데 필요한 정보만 가져주기 때문에 스마트 컨트랙트가 취약한지는 사용자가 직접 찾아내야 하는 불편함이 있다. 또한 취약한 코드를 찾더라도 어떻게 공격을 할 수 있는지 어떻게 코드를 패치해야 되는지도

사용자의 능력에 맡겨진다. 스마트 컨트랙트가 이더리움에 올라가면 더 이상 변경할 수 없기 때문에 취약점을 진단해주는 기능은 매우 유용할 것이다. 진단 방법으로는 크게 정적분석과 동적분석으로 나눌 수 있다. 동적분석의 경우 대표적으로 Mythril, Oyente에서 사용하는 기호 실행(symbolic execution) 방법이 존재한다. 하지만 Access Control, Bad Randomness의 취약점의 경우 아직 완벽하게 잡지 못하고 있다. 정적분석의 경우 소스 코드를 기반으로 검사를 하는 Smart Check가 있다. 이러한 기능들을 추가할 수 있는 방안을 모색해보려고 한다.

[참고문헌]

[1] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).

[2] Wood, Gavin, Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper, 151, 2014.

[3] Nicola AtzeiMassimo, Bartoletti, Tiziana Cimoli, A Survey of Attacks on Ethereum Smart Contracts (SoK), Conference paper, 28 March 2017.

[4] NCC Group, Decentralized Application Security Project(DASP), 2018

[5] V Buterin, A next-generation smart contract and decentralized application platform, white paper, 2014

오픈소스 저장소의 데이터를 활용한 가상화폐 프로젝트 유지보수성 평가†

최원석, 김형식

성균관대학교 전자전기컴퓨터학과

On the evaluation of software maintainability for cryptocurrency projects with their open source repository data

Wonseok Choi, Hyoungshick Kim

Department of Computer Science and Engineering, Sungkyunkwan University.

요 약

많은 가상화폐가 무분별하게 생겨나면서, 그에 따른 문제점들이 생겨나기 시작했다. 가상화폐에 대한 가치를 백서에서만 찾는 사람들을 이용해, 백서에 작성한 독창성 있는 기술을 전혀 개발하지 않고 외부 활동에만 집중하는 개발자들이 생겨나기 시작했다. 이러한 문제를 해결하기 위해, 오픈소스 저장소에 올라와 있는 가상화폐 프로젝트에 대한 정보를 이용하여 그들의 유지보수성을 평가하는 시스템을 개발하였다. 이 정보를 통해 우리는 4개의 클러스터로 나누었고 가상화폐들의 유지보수성을 확인하였다.

I. 서론

블록체인 기술을 이용한 가상화폐는 탈중앙화라는 장점으로 인해, 비트코인을 시작으로 많이 생겨나기 시작했다. 대부분의 가상화폐들의 소스코드는 오픈소스 저장소에 업로드하기 때문에, 누구든지 다운받아서 자신만의 가상화폐를 만들 수 있다. 또는, 기존의 가상화폐를 forking하여 몇 가지의 기능을 수정해 새로운 가상화폐를 만들 수 있다. 예를 들어, 비트코인 캐시(Bitcoin Cach) [1] 같은 경우, 2017년에 기존 비트코인의 거래 속도의 문제점을 해결하기 위해 등장했다.

하지만 이러한 무분별한 가상화폐의 복제로 인해, 많은 문제점들이 발생하기 시작했다. 대부분의 사람들은 가상화폐의 오픈소스가 아닌 백서 형태의 문서만 참고하여 투자하기 때문에, 투자한 가상화폐가 실제로 개발이 진행되고 있는지 확인하기 힘들다. 2017년 말, 비트코인 플래티넘이 사기인 것으로 밝혀졌다 [2]. 비트코인 플래티넘 개발자로 드러난 한국인 고등학생이 공식 비트코인 플래티넘 트위터 계정으로 장난을 치면서 사건이 드러나게 됐다. 결국 한국인 고등학생은 비트코인 플래티넘이 사기라고 밝혔고, 공식 트위터 계정과 관련 정보들이 모두 사라졌다. 또한, 일부 가상화폐 개발자들은 오픈소스 프로젝트들이 몇 개월이 지나도 개발을 진행하지 않은 채, 온라인에서만 홍보하고 있었다. 이 사건을 통해 우리는 표준화된 가상화폐 평가 시스템의 필요성을 보았다.

가상화폐를 평가하는 기준으로 백서, 프로젝트 팀 구성원, 프로젝트 소스코드 등 다양한 것들이 있다. 기본적으로, 백서를 통해 가상화폐의 합의 알고리즘 같은 기술적인 것을 알 수 있지만, 그것만으로 판단하기엔 위험이 크다.

† 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2015-0-00403)

소스코드의 유사도를 이용하여 그 가상화폐의 독창성을 판단할 수 있고, 오픈소스 저장소에서 얻을 수 있는 commit log나 개발자 기여도를 통해서 얼마나 프로젝트가 잘 유지되고 있는지를 판단할 수 있다.

이 논문에서, 우리는 가상화폐 오픈소스 프로젝트에서 얻을 수 있는 정보를 통해 가상화폐의 유지보수성을 평가하였다. Coinmarketcap [3] 은 가상화폐의 가격, 오픈소스 프로젝트 URL 등의 정보를 얻을 수 있는 유명한 가상화폐 웹사이트다. 우리는 Coinmarketcap에 등록된 가상화폐들 중 오픈소스 프로젝트가 Github URL과 연결되어 있는 것을 목표로 했다. Github에서 얻을 수 있는 정보들 중에서, 고정돼 있는 값이 아닌 개발이 지속적으로 진행되었는지를 확인할 수 있는 정보를 이용했다. 수집 된 정보를 이용해 클러스터링 알고리즘을 적용하여 가상화폐들을 분류 하였고, 클러스터링 성능 평가도 진행하였다. 또한, 각 클러스터에 속해있는 가상화폐들 중 일부를 직접 확인하여 실제로 같은 클러스터에 있는 가상화폐들이 비슷한 성질을 갖는지 확인하였다.

II. 유지보수성 평가 지표

선행연구로 오픈 소스 프로젝트가 얼마나 잘 유지 되고 있는지 평가할 수 있는 하나의 지표인 Maintainability index (MI)가 있다 [4]. MI는 Line of Code (LOC), Halstead Volume (HV), Cyclomatic Complexity (CC) 그리고 percent of lines of Comment (CM) 으로 계산된다. 또 다른 선행 연구에서는 Measuring Maintainability Index Software (MMIS) [5] 를 제안 하면서, LOC, HV 그리고 CC 모두 강한 상관계가 있기 때문에 LOC만 가지고 유지보수성 평가를 한다고 주장했다 [6]. 하지만 우리는 프로젝트의 고정된 LOC만 가지고는 유지보수성을 평가하는데 부족하다고 판단했다. 대용량의 프로젝트를 업로드 한 후, 오랫동안 개발이 이루어지지 않는다면, 유지보수성은 낮아져야 하지만, 제안된 MMIS는 일정한 값을 계속 유지 하게 된다. 따라서 우리는 수동적인 LOC를 이용하는 것이 아니라, 일정한 기간 동안 프로젝트 commit log의 수정된 LOC를 이용해서 MMIS를 구한다. 또한, 일정 기간 동안 개발에 참여한 개발자가 전체 개발자 중 어느 정도의 비율을 차지하는 Mean Developer Engagement (MDE)를 구한다. 마지막으로, 처음 프로젝트가 release 된 이후부터, 현재까지 주기적으로 얼마나 release 하는지 Frequency of Release (FR)을 구한다. 우리는 이 세 개의 지표를 이용하여, 클러스터링 알고리즘을 통해, 가상화폐의 유지보수성을 관찰 할 예정이다.

III. 클러스터링 feature

각 가상화폐의 MMIS를 구하기 위해 기간을 현재부터 3개월, 6개월, 12개월 전으로 나눈 후 각 기간 마다 수정된 LOC를 수집한다. 수집된 LOC를 이용하여 MMIS를 구하는 공식은 다음과 같다.

- LOC = count of source Lines Of Code
- HV = Halstead Volume $(45*LOC-428)$
- CC = Cyclomatic Complexity $(0.22*LOC+1.9)$
- $MMIS = 171-5.21\ln HV-0.23CC-16.2\ln LOC$

위의 공식으로 구한 MMIS의 평균과 분산을 feature로 한다. MDE는 현재부터 1년까지를 3, 6, 12개의 구간으로 나눈 후, 각 구간마다 개발에 참여한 개발자 수를 전체 개발자 수로 나누어 계산한다. FR은 프로젝트의 release 날짜의 각 차이를 계산하고 마지막 release부터 현재까지 차이를 계산하고 평균을 낸다. 각 값들이 어떻게 나타나는지 확인하기 위해 개발이 꾸준히 되고 있는 비트코인과 최하위 권에 있는 BAS 코인의 값을 비교해 보았다.

	MMIS avg(n=3)	MDE(n=3)	FR(h)
BTC	96.2	0.48	383

BAS	0	0	38832

표 1 비트코인과 BAS코인의 MMIS, MDE 그리고 FR.

표 1과 같이, 비트코인은 3개월 기간 동안의 평균 MMIS, 12개월을 3구간으로 나누었을 때 MDE, FR이 각각 96.2, 0,48, 383인 반면, BAS 코인은 0, 0, 38832로 나왔다. 비트코인은 평균 16일에 한번씩 release를 하고 평균적으로 96줄을 수정하지만 BAS코인은 1년 동안 수정한 기록이 없고 딱 한번 release를 했다. 이처럼, 개발이 활발하게 이루어지는 가상화폐와 그렇지 않은 가상화폐는 수치상의 분명한 차이가 존재한다.

총 10개의 feature 중 중요한 feature를 구하기 위해 PCA를 사용한다. PCA란, 여러 feature들 중에 어떤 feature가 중요하고 어떤 feature가 중요하지 않는가를 알기위해 사용하는 변수 추출 기법이다. PCA를 통해 feature가 얼마나 보존 됐는지를 판단하는 variance가 있다. Variance 값이 1에 가까울수록, feature를 최대한 보존한 상태로 추출 했다고 할 수 있다. 클러스터링 알고리즘은 K-means를 사용했고 클러스터링 결과를 토대로 Silhouette score를 계산 했다. Silhouette이란, 클러스터링을 평가하는 지표로 한 클러스터 안의 데이터들이 얼마나 가깝고, 다른 클러스터들은 얼마나 먼지를 구해 하나의 값으로 표현한 것이다. Silhouette은 -1부터 1사이의 값을 가지며 1에 가까울수록 잘 클러스터링 됐다고 평가할 수 있다.

IV. 클러스터링 결과

Coinmarketcap에 등록된 2000여개의 가상화폐 중 Github URL이 연결되어 있고, core를 업로드 한 657개의 가상화폐에 대하여 데이터를 수집하였다. 수집된 정보를 이용하여 K-means를 적용할 때, k=3, k=4, k=5로 각각의 PCA와 Silhouette 값을 구하였다.

	k=3	k=4	k=5

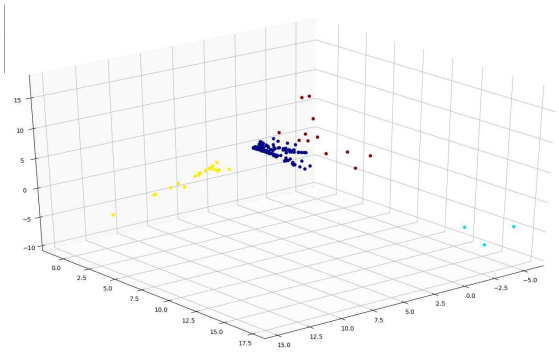


그림 1 클러스터링 이미지

표 2는 각 k값마다의 PCA variance와 Silhouette score를 보여준다. k값에 상관없이 PCA variance 값은 일정하지만 Silhouette 값은 k는 4에서 최댓값을 갖는다.

그림 1은 k=4일 때 클러스터링 한 결과를 보여준다. 노란색 클러스터는 대부분 MMIS, MDE 값이 높고 FR이 낮은 것으로 확인 되었다. 파란색 클러스터는 MMIS가 높으나 MDE값이 노란색 클러스터보다 상대적으로 낮았다. 갈색 클러스터는 전체적으로 모든 값이 낮았지만 3,6, 12 개월의 MMIS 중 일부 값이 높은 것을 확인했다. 하늘색 클러스터는 모든 지표가 거의 최저였다. 노란색 클러스터는 비트코인이 포함된 클러스터로 이더리움, 리플 같은 흔히 사람들에게 알려져 있는 가상화폐들이 포함되어 있었다. 파란색 클러스터에는 중간정도 개발이 진행되는 대부분의 가상화폐들이 포함되어 있었고, 갈색 클러스터에는 BAS코인 같은 몇 개월 이상 개발이 이루어지지 않는 가상화폐들이 포함되어 있었다. 하늘색 클러스터에 포함되어 있는 가상화폐들은 생성된 날이 1년도 지나지 않은 가상화폐들이었다.

V. 결론

가상화폐의 유지보수성을 평가하기 위해 오픈소스 프로젝트에서 얻을 수 있는 정보를 이용하여 클러스터링을 실시하였다. 그 결과 비트코인과 같이 개발이 잘 이루어지고 있는 클러스터와 중간단계의 클러스터, 몇 개월간 개발이 이루어지지 않은 가상화폐들의 클러스터, 그리

고 생긴 지 얼마 안 된 가상화폐들의 클러스터로 나누어졌다. 이 클러스터링 결과를 통해 우리는 어떤 가상화폐들이 개발이 잘 이루어지고 있는지를 시각적으로 확인 할 수 있었다. 하지만 개발을 잘 되고 있는 오픈소스 프로젝트를 그대로 fork한다면, 기존의 정보들을 가져올 수 있기 때문에 마치 잘 개발되는 것처럼 보일 수 있다. 따라서 우리는 가상화폐의 유지보수 평가뿐만 아니라, 서로간의 유사도를 비교해서 가상화폐들이 얼마나 서로 유사한지 또는 얼마나 독자적인 기술을 갖고 있는지를 평가 할 예정이다.

[참고문헌]

[1] NAKAMOTO, S. Bitcoin: a peer-to-peer electronic cashsystem. http://bitcoin.org/bitcoin.pdf, 2008.

[2] YOUNG, J. Bitcoin Platinum Created By South Korean Teenager As a Scam, Local Market Erupts, De-cember 2017. cointelegraph.com

[3] CoinMarketCap: Cryptocurrency Market Capitalizations. https://coinmarketcap.com

[4] HOWARD, W. Software Measurement and Estimation: A Practical Approach, vol. 36. Wiley, 2007.

[5] VAN DER MEULEN, M. J., AND REVILLA, M. A. Correlations between internal software metrics and software dependability in a large population of small C/C++ programs. In Software Reliability, 2007. IS-SRE'07. The 18th IEEE International Symposium on(2007), IEEE.

[6] SPINELLIS, D., AND GOUSIOS, G. Evaluating the quality of open source software. Electronic Notes in Theoretical Computer Science 233(2009)

비정형 트랜잭션의 비식별 처리 품질 평가

용우석*, 이원석*
*연세대학교 컴퓨터과학과

Quality Evaluation Non-identified Unstructured Transaction Data

Woo-Seok Yong*,Won-Seok Lee*
*Dept. of Computer Science, Yonsei University.

요 약

트랜잭션 데이터는 개인의 마켓 구매 정보나 환자의 증상 및 행동에 대한 의료 데이터에 이르기까지 다양한 형태의 항목 집합 형태로 나타난다. 트랜잭션 데이터를 연구 목적으로 공개할 때, 여러 개인의 트랜잭션들이 모인 데이터베이스에서 특정 개인을 식별할 수 있게 되면 개인의 사생활 정보가 노출되는 위험성이 존재하게 된다. 이 재식별 위험성을 방지하기 위해 원본 데이터를 변환하는 비식별화 모델들이 연구되어 왔다. 변환된 비식별 데이터 세트는 원본 데이터 세트와 반드시 다른 값을 갖게 되고 이는 분석 결과에도 영향을 주기 때문에 원본 데이터 세트와 얼마나 유사한지에 대해 측정하여 변환된 데이터 세트의 활용성을 평가해야 한다. 본 연구에서는 비식별 데이터와 원본 데이터 간의 유사도에 대한 지표를 세 가지 측정 기법을 제안하며 그 유효성을 실제 트랜잭션 데이터 실험을 통해 검증하였다.

I. 서론

최근에 들어 정보 통신 기술의 급속한 발전과 폭발적인 성장으로 인하여 데이터의 양이 늘어나고 이에 따라서 고도화된 분산 처리 시스템을 기반으로 하여 대용량 데이터의 수집 및 관리 기술과 분석 기술이 발전하고 있다. 이에 따라 빅데이터 기반의 데이터 마이닝과 여러 기계 학습 모델들이 주목을 받고 있으며 여러 활용 모델이 연구되고 있는 추세이다. 이러한 데이터 기반의 연구가 수행이 되기 위해선 연구를 진행 할 데이터들의 유통이 필수적이다. 유통되는 대용량 데이터 중 비정형 데이터인 트랜잭션 데이터는 개개인의 슈퍼마켓 구매 정보 및 병원의 진단 내역을 포함하여 다양한 정보를 갖으며 다양한 형태의 항목 집합 형태로 나타난다. 연구를 위해 트랜잭션 데이터를 유통할 때, 트랜잭션 데이터 안에 포함되어 있는 개개인의 기록을 신원 정보에 연결하게 되면 개인 정보 침해가 발생할 위험성이 존재한다. 이와 같은 개인 정보의 침해를 막기 위해 개인정보보호 관련 법률이 발의되면서 개인의 사생활 정보 유출 관련 문제가 발생하지 않도록 현행 법규에서 제시하는 개인 식별 가능 정보가 포함된 데이터 활용 시 데이터의 비식별화 변환이 필수적으로 요구된다.[1] 트랜잭션 데이터에서는 개인의 기록에서 특정 값을 제거하거나 더 범용적인 값으로 대체하여 데이터를 익명화[2,3,5]하는 방법들과 변환 후의 재식별 가능성을 최소화 하기 위한 p-중복 익명성 프라이버시 모델이 적용된 비식별화 기법들이 활용 되고 있다. 본 논문은 본 논문에서 제안하는 원본 유사도 검증 지표를 기반으로 비식별 데이터 세트에 대한 평가 지표를 제시한다. 원본 유사도는 세 가지 측정 기법을 통해 그 값을 수치적으로 나타내어 변환된 데이터에 대해 통계적 유사성을 정량적으로 평가한다.

결과적으로 원본 유사도를 트랜잭션 비식별 조치에 대한 품질 평가 지표로 사용하고 실제

트랜잭션 데이터 실험을 통해 검증한다.

II. 관련연구

트랜잭션 비식별화에 대한 평가는 비식별된 데이터의 활용성(Utility)으로 평가되는데 이를 수치적으로 계산하여 표현한 것은 정보 손실율(Information loss)이 있다. 트랜잭션 데이터는 비식별화가 일어날 때, 항목들의 계층 구조에서 상위 항목으로 일반화가 일어나기 때문에 정보 손실을 초래한다. 정보 손실율 계산은 다양한 방식으로 계산할 수 있다. 정보 손실율의 계산 방식은 일반적으로 *Discernability Metric*[4]와 *Normalized Certainty Penalty*[4]를 활용하여 각 비식별 프라이버시 모델에 적용해 계산한다. *Discernability Metric*은 같은 계층에 있는 항목의 수 대비 일반화되는 비율을 계산하여 정보 손실율을 계산하고 *Normalized Certainty Penalty*는 이를 더 일반화시켜 범용적으로 정보 손실율을 계산한다. 살펴본 것처럼 기존의 트랜잭션 비식별화에 대한 평가는 트랜잭션 데이터의 정보 손실만을 계산하여 변환된 비식별 데이터의 품질을 평가하였다.

III. 비식별 트랜잭션 원본 유사도

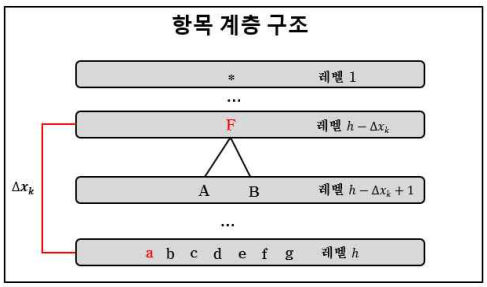
비식별 모델에 관한 검증 방법으로 원본 유사도는 원본 레코드 세트와 결과 레코드 세트 간의 차이를 수치적으로 측정한 지표이다. 생성된 비식별 레코드와 원본 레코드 간의 유사도를 기준 삼아 유통용 데이터의 활용성을 총 세 가지 지표를 사용하여 그 값의 평균으로 평가한다.

3.1 잔존율

잔존율은 비식별 변환 과정에서 해당 비식별 처리에 위반하는 레코드를 제거하여 이에 따라 원본 레코드 수보다 적은 수의 결과 레코드를 보유하고 있다. 이렇게 제거되는 레코드의 수가 많을수록 데이터 세트의 활용성은 떨어지게 되므로 원본레코드 수 대비 결과레코드 수를 비율로 잔존율을 정의하여 데이터 활용성에 대한 지표로 평가한다.

3.2 항목 계층 레벨 변화량

트랜잭션 데이터는 일반적으로 항목에 대한 계층 구조를 갖으며 비식별 처리가 될 때 해당 항목의 상위 계층으로 항목이 일반화 된다. 평균 계층 레벨 변화량은 트랜잭션의 각 항목에 대하여 일반화가 얼마나 일어났는지를 계산하여 원본 레코드 세트 간의 차이를 수치적으로 표현한다. 평균 계층 레벨 변화량을 표현하기 위해 계층의 총 레벨을 $|h|$, 트랜잭션 항목 집합을 U_{xk}, 트랜잭션에 속한 항목을 x_k 그리고 해당 항목이 일반화 된 폭을 $\triangle x_k$로 표현한다. 일반적인 항목 계층 구조는 그림1 과 같이 나타난다.



[그림 1] 항목 계층 구조

위에 제시된 정보들과 그림 1을 이용하여 변환된 트랜잭션에 대한 평균 계층 레벨 변화량을 계산하고 이를 원본 유사도에 대한 지표로 활용하여 평가한다.

3.3. 항목 길이 차이

트랜잭션 데이터가 비식별 처리되며 항목이 최고 상위 계층인 루트 계층(*) 항목으로 일반화 되는 경우 해당 항목은 트랜잭션에서 탈락된다. 항목 길이 차이는 탈락되는 항목들의 정도를 원본 트랜잭션 항목의 수 대비 계산하여 원본 유사도에 대한 지표로 평가한다.

TRANSACTION	USER_ID	ITEMS
TID1	UID1	우유,계란,버터,식빵
TID2	UID2	우유,계란,식빵
TID3	UID3	계란,식빵
TID4	UID1	우유,계란,커피
TID5	UID2	식빵,라면
TID6	UID1	식빵,버터

[표 1] 원본 트랜잭션 데이터베이스

TRANSACTION	USER_ID	ITEMS
TID1	UID1	우유,계란,식빵
TID2	UID2	우유,계란,식빵
TID3	UID3	계란,식빵
TID4	UID1	우유,계란
TID5	UID2	식빵,음식
TID6	UID1	식빵,음식

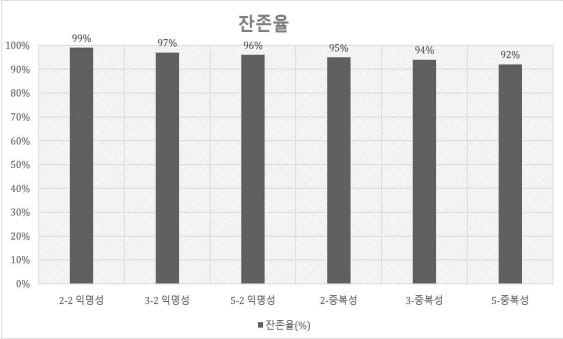
[표 2] 비식별 트랜잭션 데이터베이스

위의 예 [표 1]은 원본 트랜잭션 데이터베이스고 [표 2]는 비식별 처리된 트랜잭션 데이터베이스이다. <TID1, UID1>를 보면 항목 (우유,계란,버터,식빵)에서 (우유,계란,식빵)으로 변환이 일어난 것을 볼 수 있다. 이는 버터가 일반화가 일어나다 루트(*) 항목으로 일반화되어 탈락한 경우로 <TID1,UID1>의 항목 길이 차이는 $\frac{3}{4}$ (원본 항목의 길이 4, 비식별 항목의 길이 3)으로 계산된다. 그리고 [표 2]의 비식별 트랜잭션 데이터베이스 전체의 항목 길이 차이는 $\frac{7}{8}$로 계산된다. (원본 트랜잭션 데이터베이스의 1-항목 개수 16, 비식별 트랜잭션 데이터베이스의 1-항목 개수 14)

IV. 실험 검증

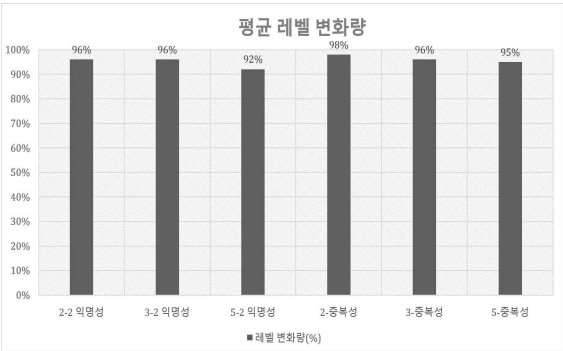
본 논문에서 실험한 데이터는 실제 교육 사이트의 여름방학 기간 동안 수집한 클릭 로그 데이터로 각 개인 유저가 클릭한 시간 기준 30초 이내의 클릭한 페이지(항목)들을 하나의 트랜잭션으로 만들어 원본 레코드 수 77,082건, 유저 수 121명, 항목에 대한 계층 레벨은 5인 데이터 세트이다. 해당 원본 데이터를 대상으로 k^m-익명성(k=2,3,5 m=2)과 p-중복 익명성(p=2,3,5) 프라이버시 모델로 비식별화된 두 비식별 데이터 세트의 활용성을 판단하기 위해 앞서 정의한 세 가지 지표의 원본 유사도를 계산하고 분석하였다.

첫 번째로 잔존율은 두 프라이버시 모델에서 큰 차이를 보이지 않았지만 k^m-익명성이 미세하게 좋은 잔존율을 보임을 알 수 있었다. p-중복 익명성 모델은 여러 명의 개인에 대한 개인 정보를 담고 있는 트랜잭션 데이터베이스에서 동일한 항목 집합들이 최소 서로 다른 p명의 해당 항목 집합에 나타나게 비식별 조치를 취하기 때문에 개인 재식별 가능성은 낮지만 잔존율이 미세하게 낮음을 확인할 수 있다. 또한 k값과 p값이 커질수록 비식별 조치가 엄격하게 이루어지므로 k값과 p값에 따라 잔존율이 감소하는 것도 확인할 수 있다.



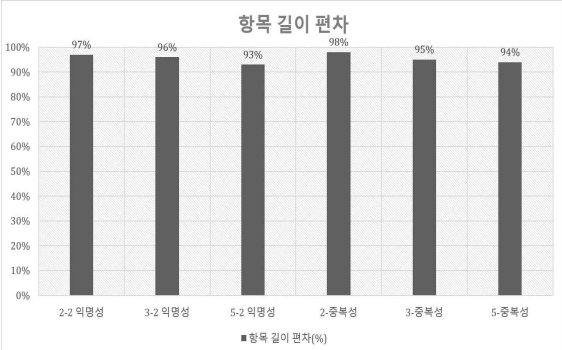
[그림 2] 잔존율

두 번째로 평균 레벨 변화량도 두 프라이버시 모델에서 큰 차이를 보이지 않았다. 하지만 비식별 조치를 취할 때, 각 1-항목에 대하여 일반화를 시키는 p-중복 익명성 모델이 해당 항목 계층 전체를 일반화시키는 k^m-익명성보다 더 활용성이 좋은 레벨 변화량을 나타내었다.



[그림 3] 평균 레벨 변화량

마지막으로 항목 길이 차이는 앞서 말한 평균 레벨 변화량과 같은 이유로 p-중복 익명성 모델이 k^m-익명성보다 항목 길이 차이 측면에서 더 좋은 활용성을 보였다.



[그림 4] 항목 길이 차이

결과적으로 원본 유사도의 세 가지 지표 측면에서 p-중복 익명성 모델이 잔존율은 조금 떨어지지만 비식별 처리를 하여 항목의 일반화가 일어날 때, 각 1-항목에 대하여 일반화를 시키기 때문에 k^m-익명성보다 평균적으로 더 높은 원본 유사도가 측정되어 그 사용가치가 높다고 판단되어진다.

V. 결론

본 논문에서는 트랜잭션 비식별화 변환 기법의 검증 지표 중 정보 손실율 만을 검증 지표로 두었던 기존 검증 지표의 한계를 원본 유사도의 세 가지 측정 기법으로 나누어 비식별화 검증 할 것을 제안했다.

향후 연구에서는 검증 수행 시간을 줄이기 위해 오픈 소스 빅데이터 처리 엔진 기반의 병렬 기술을 적용 시키는 연구와 원본 유사도와 트레이드-오프(trade-off) 관계인 재식별 위험성 및 개인 추정도 검증에 대한 연구가 추가적으로 필요할 것으로 보인다.

Acknowledgement

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진행센터의 지원을 받아 수행된 연구임(No.2015-0-00579, 빅데이터 환경에서 비식별화 기법을 이용한 개인정보 보호 기술 개발)

[참고문헌]

[1] Vatsalan, Dinusha, Peter Christen, and Vassilios S.Verykios. "A taxonomy of privacy-preserving record linkage techniques." Information Systems 38.6 (2013)

[2] Terrovitis M, Mamoulis N, Kalnis P. Local and global recoding methods for anonymizing set-valued data. VLDB J. (2011)

[3] Loukides G, Gkoulalas-Divanis A, Shao J. Anonymizing transaction data to eliminate sensitive inferences. DEXA. (2010)

[4] Terrovitis M, Mamoulis N, Kalnis P. Privacy-preserving anonymization of set-valued data. PVLDB. (2008)

[5] Grigorios Loukides, Aris Gkoulalas-Divanis Utility-preserving transaction data anonymization with lowinformation loss. PMC (2013)

로그 기반 변종 악성코드 탐지 방법1)

조우진, 김형식
충남대학교 컴퓨터공학과

Log-based variant malicious code detection method

Woo-Jin Joe, Hyong-Shik Kim
Dept of Computer Science & Engineering, Chungnam National University.

요 약

백신은 시그니처 기반으로 탐지하기 때문에 변종 악성코드를 탐지하는데 한계가 있다. 기업에서는 위협 탐지 목적으로 호스트에서 발생되는 이벤트를 수집하지만 분석가의 역량 부족 등의 이유로 대용량 로그를 제대로 활용하지 못하고 있다. 본 논문에서는 호스트에서 생성되는 로그를 이용한 변종 악성코드 탐지 방법을 제안한다. 로그를 사용하여 악성 행위에 수반되는 이벤트를 정의하고, 유사한 방식으로 이벤트를 발생시키는 기능을 수행하는 프로그램을 찾은 후에, 추가적으로 다른 행위 발생여부를 고려해 오탐을 줄인다. 우리는 실험을 통해 악성코드의 특징 이벤트를 정의하였고, 오탐이 발생하지 않음을 확인하였다.

I. 서론

2014년 미국의 보안 회사인 lastline에서 수행한 실험에 따르면 새로운 악성코드 샘플이 주어졌을 때 47개의 백신 중 50%가 탐지하지 못하며, 1년이 지나도 10%는 탐지하지 못 한다고 한다[1]. 이는 백신이 시그니처 기반으로 탐지하기 때문에 신종, 변종 악성코드 탐지에 한계가 있어 발생한 것으로 추측된다.

2018년 보안 연구소 Av-test의 연구에 따르면 매년 악성코드의 수는 증가하지만 대부분 새로운 악성코드가 아닌 변종 악성코드라고 한다[2]. 변종 악성코드는 코드 난독화 기술을 이용해 쉽게 제작될 수 있어 시그니처 생성 속도가 변종 생성 속도를 따라 잡을 수 없다. 따라서 변종 악성코드를 효율적으로 탐지하기 위한 연구가 필요하다.

기업에서는 이러한 한계를 극복하기 위해 보안 정보 및 이벤트 관리(Security Information & Event Management)를 도입하여 호스트에서 발생하는 이벤트를 수집하여 분석하지만, 기업의 규모가 클수록 대용량의 데이터가 수집되고, 분석가의 역량 부족으로 인해 활용하지 못하고 있다.

우리는 호스트에서 생성되는 로그를 이용한 변종 악성코드 탐지 방법을 제안한다. 제안하는 방법은 악성코드가 주된 목적을 위해 발생시키는 이벤트를 로그를 이용해 정의하고, 이를 이용하여 유사한 기능을 수행하는 프로그램을 찾는 것이다. 추가적으로 다른 행위 발생 여부를 고려해 오탐을 제거한다. 유사한 프로그램은 로그로부터 조회되기 때문에 단일 호스트가 아닌 대규모 조직의 호스트로부터 변종 악성코드를 찾을 수 있다. 실험을 통하여 유사한 프로그램

1) 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2018-2016-0-00304)

을 찾아 추가 이벤트를 적용한 결과 오탐이 없음을 확인하였다.

II. 관련 연구

문대성 외 2인은 보안 장비가 APT(Advanced Persistent Threat) 공격에 취약하다는 점을 보완하기 위해 호스트에서 발생하는 프로세스의 행위정보를 기반으로 악성코드를 탐지하는 연구를 했다[3]. 해당 논문에서는 39개의 윈도우 API를 이용해 프로세스의 특징 정보를 정의하였고, 실제 실험을 통해 정의한 특징 정보를 수집하여 결정트리를 이용한 악성코드 분류 실험을 하였다. 실험 결과 2.0%의 오탐율(False Positive)과 5.8%의 미탐율(False Negative)를 보여 해당 특징 정보가 유효함을 입증하였다.

윈도우 API가 아닌 프로세스가 적재하는 DLL(Dynamic-link library)을 이용하는 연구도 있었다[4][5]. 이는 윈도우 패스워드를 탈취하는 악성코드인 mimikatz가 반드시 적재하는 필수 DLL 리스트를 이용하여 탐지하는 것에 관한 것이다. Wataru는 적재하는 DLL이 운영체제 버전, 악성코드 버전에 따라 달라질 수 있음을 보이고, 버전별로 공통되는 필수 리스트를 뽑아내어 탐지하려는 연구를 수행했다[6].

그렇지만 이러한 탐지 방법들은 필수 DLL 리스트를 정의하기 위해 탐지하려는 악성코드를 여러 번 실행시켜야 한다는 번거로움이 있다. 우리는 프로그램이 흔적으로 남긴 로그를 이용하여 효율적으로 필수 DLL 리스트를 정의하는 방법을 제안한다. 또한 DLL 뿐만 아니라 다른 행위와 결합하여 더 좁은 범위에서 유사한 프로그램을 찾고, 추가적으로 오탐을 제거하기 위해 다른 행위 발생 여부를 고려한다.

III. 로그 기반 프로그램 특징 분석

우리는 변종 악성코드를 찾기 위해 프로그램을 구분할 수 있는 특징 정보를 로그를 이용하여 정의 하려한다. 로그는 프로그램이 남긴 흔적으로 이를 이용할 수 있으면 대규모 조직의 호스트로부터 침입 여부를 판단할 수 있다. 프로그램이 특수한 목적을 위해 발생시키는 이벤트는 이러한 특징 정보가 될 수 있으며, 이 이벤트를 발생시키는 프로그램들은 유사한 프로그램이라 간주할 수 있다. 변종 악성코드는 기존의 것과 수행하는 기능에는 큰 변화가 없기 때문에 이러한 방법으로 탐지할 수 있게 된다.

우리는 프로그램이 발생시키는 이벤트 중 흔한 이벤트를 제외하면 특수한 목적을 위해 발생시킨 이벤트라는 전제로 시작하였다. 프로그램의 특징이 될 수 있는 이벤트를 찾기 위해선 먼저 흔하게 발생되는 이벤트를 수집할 필요가 있다. 우리는 이를 위해 윈도우 7에서 약 250개의 정상 프로그램을 실행 시켰다. 실행한 프로그램은 기본 프로그램과 여러 유틸리티를 실행시켰다. 그리고 커널에 접근하는 등 고급 기능을 수행하는 프로그램을 실행하기 위해 74개의 Sysinternals 도구를 실행시켰다.

프로세스 행위를 모니터링하고 로그를 생성하기 위해 Sysmon이라는 모니터링 도구를 사용한다. Sysmon은 Sysinternals에서 제공하는 도구로 윈도우 로그를 보완하기 위해 같은 형식을 따른다. 즉, 윈도우 로그에 고유 Event Id를 부여하는 것처럼 Sysmon도 1부터 18까지 Id를 분류한다. 우리는 표 2와 같이 2가지 로그를 이용해 특징 이벤트를 정의한다.

Event Id	설 명	이벤트 대상
7	DLL 적재	ImageLoaded
10	타 프로세스 주소 공간 접근	TargetImage

<표 2> Sysmon 로그

Sysmon이 생성하는 로그에는 발생 시간, 프로세스 Id 등 해당 이벤트에 대한 정보가 기록되는 여러 필드 존재한다. 그 중에서 이벤트 대상 필드는 이벤트의 대상이 기록되는 필드이다. 적재한 DLL 명은 'imageLoaded' 필드에, 접근할 타 프로세스의 실행 이미지 명은 'TargetImage'에 저장된다.

3.1 프로그램 특징 정보 1

첫 번째 프로그램 특징 정보는 DLL 적재 이벤트를 이용해 정의한다. 프로세스는 목적하는 기능을 수행하기 위해 그 기능을 구현한 DLL을 반드시 적재해야 한다. 따라서 프로그램의 목적에 따라 적재하는 DLL은 달라질 수밖에 없다. 대부분의 프로세스가 적재하는 DLL은 식별성이 없지만 적은 수의 프로세스가 적재하는 DLL은 특수한 기능이 구현된 DLL이라 간주할 수 있고 프로그램의 특징이 될 수 있다.

우리는 정상 프로그램이 적재하는 DLL로부터 흔한 DLL을 정의한다. 표 3은 250개의 정상 프로세스가 적재한 2,035개의 DLL을 적재하는 프로세스(SourceImage) 수가 많은 순서대로 정렬한 것이다. ntdll.dll의 경우 250개 프로세스 중 241개의 프로세스가 적재했으므로 흔한 DLL이라고 말할 수 있지만, WMIsvc.dll은 2개의 프로세스만 적재했으므로 특수한 기능이 구현되었다고 간주할 수 있다.

ImageLoaded	SourceImage 개수
ntdll.dll	241
kernel32.dll	240
user32.dll	236
...	
WMIsvc.dll	2

<표 3> Event Id 7 이벤트

이번 실험에서는 SourceImage 개수가 전체 프로세스 개수의 10% 이하인 DLL을 적재하는 이벤트를 특징 정보로 간주한다. 250개의 프로세스가 실행되는 동안 총 2,035개의 DLL이 적재되었고, 각 DLL마다 적재한 프로세스 수를 알기 위해 SourceImage 개수를 셈하였다. SourceImage 수가 10% 이상(25개 이상)인 DLL은 137개이고 이 리스트는 흔한 DLL이라고 볼 수 있다. 우리는 이 리스트를 저장하고, 리스트에 속하지 않은 DLL을 적재하는 이벤트를 특징 정보로 정의한다.

3.2 프로그램 특징 정보 2

두 번째 프로그램 특징 정보는 타 프로세스 접근 이벤트를 이용해 정의한다. 타 프로세스 주소 공간에 접근하는 것은 다양한 이유가 있을 수 있다. 이 중에서 특정 목적을 달성하기 위해선 특정 프로세스에 접근할 수밖에 없고, 이는 프로세스를 구분하는 특징 정보가 된다.

유저 프로세스에 접근하는 이벤트는 흔하다고 할 수 있지만 시스템 프로세스에 접근하는 경우는 흔하지 않은 이벤트라고 간주할 수 있다. 예를 들어 윈도우 패스워드를 탈취하는 mimikatz 악성코드의 경우 lsass.exe 시스템 프로세스 주소공간에 접근한다. 따라서 Event Id 10 로그의 'TargetImage' 필드에 시스템 프로세스가 기록되면 특징 이벤트라고 간주할 수 있다.

IV. 오탐 제거 방법

이번 절에서는 특징 정보를 통해 유사하다고 판단된 프로그램에 대해 오탐을 제거하는 방법을 설명한다. 악성코드에 대하여 특정 행위가 발생하지 않았다면 변종 악성코드도 해당 행위를 수행하지 않을 가능성이 크다. 따라서 3절에서 정의한 특징 정보 이벤트가 발생했더라도 악성코드에서 발견되지 않은 행위를 수행할 경우 변종이 아닌 것으로 간주한다.

추가 행위는 악성코드가 주로 수행하는 행위인 파일 생성, 네트워크, 레지스트리를 선별했다. 악성코드는 추가 작업을 위해 실행 파일을 생성시키거나 정보를 탈취하기 위해 네트워크 연결을 시도한다. 레지스트리의 경우 악성코드를 서비스 혹은 예약된 작업으로 등록하기 위해 발생된다. 표 4와 같이 4가지 로그를 이용하여 정의하였으며, Event Id 12, 13의 경우 레지스트리 활동으로 결합하여 네트워크, 파일, 레지스트리 총 3가지 행위로 분류할 수 있다.

Event Id	설명
3	네트워크 연결
11	파일 생성
12	레지스트리 생성, 삭제
13	레지스트리 수정

<표 4> 추가 로그

V. 실험

이번 절에서는 로그를 이용하여 실제 악성코드의 특징 정보를 정의하고 추가 행위로 오탐을 제거하여 변종 악성코드를 찾아낸다. 우선 3절에서 설명한 것처럼 137가지 흔한 DLL 리스트와 시스템 프로세스에 접근하는 이벤트를 이용해 악성코드 특징 정보 이벤트를 정의한다. 정의한 이벤트를 기반으로 유사한 기능을 수행하는 프로그램을 찾아내고 4절에서 설명한 추가 행위를 이용해 오탐을 제거한다.

5.1 악성코드 특징 정보 정의

실험에 사용할 악성코드는 mimikatz를 이용했다. Event Id 7의 경우 47개의 DLL을 적재하였고, 이 중에서 특징 정보로 정의된 DLL은 13개였다. Event Id 10의 경우 패스워드 탈취를 위해 lsass.exe 시스템 프로세스에 접근하였으므로 특징 정보 이벤트로 정의된다.

5.2 유사 프로그램 탐지

250개의 정상 프로그램을 대상으로 특징 정보 이벤트의 발생 여부를 검사하였다. 먼저 6개의 프로그램에서 Event Id 10 특징 이벤트가 탐지되었다. 이 6개의 프로그램에 대해 Event Id 7 특징 이벤트를 적용한 결과 총 3가지 프로그램(ALSong343, ALUpdate, Procmon)이 탐지되었다.

5.3 오탐 제거

표 5는 탐지된 3가지 프로그램에 대하여 오탐을 제거하기 위해 추가 이벤트를 적용한 것이다. P1은 'ALSong343', P2는 'ALUpdate', P3는 'Procmon'을 지칭한다.

Event Id	mimikatz	P1	P2	P3	오탐 제거율
3	F	F	T	T	2/3
11	F	T	T	T	3/3
12, 13	F	T	F	T	2/3

<표 5> 로그별 오탐 제거율

실험 결과 3개의 유사한 기능을 수행하는 프로그램을 탐지하였지만, 추가 이벤트를 적용하여 오탐을 제거한 결과 정상으로 분류되었다. 예로 P1의 경우 mimikatz가 발생시키지 않은 네트워크(Event Id 11), 레지스트리(Event Id 12, 13) 행위를 수행했기 때문에 오탐으로 제거되었다. 제거율을 확인해보면 파일 생성 행위만 이용해도 모두 정상으로 분류할 수 있었지만 본 실험은 mimikatz로 제한되었으므로 다른 프로세스 활동도 고려해야 한다.

VI. 결론

우리는 로그를 이용한 변종 악성코드 탐지 방안을 제안하였다. 로그를 이용해 악성코드의 특징 이벤트를 정의하였고, 이로 유사한 기능을 수행하는 프로그램을 탐지하였다. 그리고 유사한 프로그램에 대한 오탐을 제거하기 위해 추가 이벤트를 적용하였고, 효과적으로 오탐을 제거했음을 확인하였다. 향후 프로그램의 특징이 될 수 있는 행위를 추가적으로 정의함으로써 탐지 정확도를 향상 시킬 필요가 있다.

[참고문헌]

[1] lastline, "Antivirus isn't Dead, It just Can't Kepp Up" https://www.lastline.com/labsblog/antivirus-isnt-dead-it-just-cant-keep-up/

[2] AV-Test, "latest malware statistics and trends report", https://www.av-test.org/en/statistics/malware/

[3] 문대성, "APT 공격 탐지를 위한 호스트 기반 특징 표현 방법", 정보보호학회논문지, 24(5), pp839-850, 2014. 10

[4] "DETECTING IN-MEMORY MIMIKATZ", https://securityriskadvisors.com/blog/detecting-in-memory-mimikatz/

[5] "Hunting for In-Memory Mimikatz with Sysmon and ELK", https://cyberwardog.blogspot.com/2017/03/chronicles-of-threat-hunter-hunting-for.html

[6] Wataru, "Tracking mimikatz by Sysmon and Elasticsearch", https://hitcon.org/2017/CMT/slide-files/d2_s1_r1.pdf