

Hooking 사용 실습

발표자: 양유진

링크: https://youtu.be/3lCq5g1y_5g

1. Hooking이란?

운영체제/응용 프로그램이 사용하는 동적 라이브러리의 함수가 호출될 때
중간에서 해당 함수를 가로채 제어권을 얻어내는 기술 및 행위

1. Hooking이란?

- 함수 호출 전/후에 사용자 코드 실행
 - Hooking된 함수의 파라미터/리턴값 확인 및 조작
 - 기존의 함수 호출 방해
 - 프로그램의 실행 흐름 변경 등
- 동적 라이브러리를 이용하여 Hooking 구현 가능
- Linux, Windows 환경에서 구현 가능
 - Linux: dlsym 함수 사용 → **so**(shared object) 생성
 - Windows: Win32 API 사용 → **dll**(dynamic link library) 생성

동적 라이브러리 vs 정적 라이브러리

- 동적 라이브러리는 프로그램 실행 중 특정 라이브러리를 필요로 할 때 로드
- 정적 라이브러리는 프로그램에 라이브러리를 모두 포함시켜 시작부터 라이브러리를 로드

	동적 라이브러리	정적 라이브러리
장점	실행 파일 크기 줄일 수 있음	실행 시간 줄일 수 있음
단점	실행 시간 증가	실행 파일 크기 증가

2. 실습 환경 구성

1) 윈도우 환경 구성

- Visual Studio 2022 사용 [설치](#)

2) 우분투 환경 구성

- Ubuntu Linux 22.04 LTS
- 가상머신 Oracle Virtual box / VMware [설치](#)

* 모든 설치를 마치고 우분투를 실행했을 때, 화면이 계속 멈춰 있는 문제가 발생할 경우
[→ 링크](#)

- 그래픽 컨트롤러의 문제일 수도 있음. ([링크](#))

* Virtual box를 이용할 경우 중간 중간 Snapshot 기능을 이용하여 상태 저장을 권장함.

3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

dll 생성하는 프로젝트 생성 방법

KeyHook.cpp

```
// KeyHook.dll
#define _CRT_SECURE_NO_WARNINGS //C4996 에러 해결
#include "stdio.h"
#include "windows.h" hooking하고자 하는 응용프로그램

#define DEF_PROCESS_NAME "notepad.exe" C:\windows\system32\

HINSTANCE g_hInstance = NULL;
HHOOK g_hHook = NULL;
HWND g_hWnd = NULL;
FILE* f1;

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD dwReason, LPVOID lpvReserved)
{
    switch (dwReason)
    {
        case DLL_PROCESS_ATTACH:
            g_hInstance = hinstDLL;
            break;

        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}
```

3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

KeyHook.cpp

```
extern "C" {
    LRESULT __declspec(dllexport) CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)
    {
        char szPath[MAX_PATH] = { 0, };
        char* p = NULL;
        char ch;
        HWND pWnd, pEdit = NULL;

        if (nCode >= 0)
        {
            // bit 31 : 0 => key press, 1 => key release
            if (!(lParam & 0x80000000))
            {
                GetModuleFileName(NULL, szPath, MAX_PATH); //파일 경로를 szPath에 담음
                p = strrchr(szPath, '\\');

                if (!_stricmp(p + 1, DEF_PROCESS_NAME)) {
                    ch = (char)wParam; //wParam을 char로 강제형변환.. 대소문자 구분이 안되고 일부 깨짐
                    f1 = fopen("hooking.log", "a"); //키보드 로그를 저장할 경로
                    fputc(ch, f1);
                    fclose(f1);
                    if (wParam == VK_RETURN) { //엔터가 입력된 경우 메모장의 윈도우를 얻어와 "hooking"을 출력 (VK_RETURN: ENTER)
                        pWnd = FindWindow("notepad", NULL); //실행파일의 윈도우 얻어오기
                        if (pWnd == NULL) {
                            printf("null!");
                        }
                    }
                    else
                        pEdit = GetWindow(pWnd, GW_CHILD); //실행파일의 하위 윈도우
                }
            }
        }
    }
}
```

3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

KeyHook.cpp

```
if (pEdit != NULL) { //메모장에 출력
    SendMessage(pEdit, WM_CHAR, 'h', 0);
    SendMessage(pEdit, WM_CHAR, 'o', 0);
    SendMessage(pEdit, WM_CHAR, 'o', 0);
    SendMessage(pEdit, WM_CHAR, 'k', 0);
    SendMessage(pEdit, WM_CHAR, 'i', 0);
    SendMessage(pEdit, WM_CHAR, 'n', 0);
    SendMessage(pEdit, WM_CHAR, 'g', 0);

    return 0;
}
return 0; //0을 리턴하면 메시지는 정상 전달됨
}
}
} // 일반적인 경우에는 CallNextHookEx() 를 호출하여 응용프로그램 (혹은 다음 훅) 으로 메시지를 전달함
return CallNextHookEx(g_hHook, nCode, wParam, lParam);
}
```

3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

KeyHook.cpp

```
#ifdef __cplusplus
extern "C" {
#endif

__declspec(dllexport) void HookStart()
{
    g_hHook = SetWindowsHookEx(WH_KEYBOARD, KeyboardProc, g_hInstance, 0); //후킹 시작
    //WH_KEYBOARD: GetMessage()/PeekMessage() 함수에서 반환되는 키보드 입력 이벤트 모니터링 가능
    printf("Hook Start\n");
}

__declspec(dllexport) void HookStop()
{
    if (g_hHook)
    {
        UnhookWindowsHookEx(g_hHook); //후킹 종료
        g_hHook = NULL;
    }
}

#ifdef __cplusplus
}
#endif
```


3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

Hookmain.cpp

```
//dll 로더
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "conio.h"
#include "windows.h"

#define DEF_DLL_NAME "KeyHook.dll"
#define DEF_HOOKSTART "HookStart"
#define DEF_HOOKSTOP "HookStop"

typedef void (*PFN_HOOKSTART)();
typedef void (*PFN_HOOKSTOP)();
```

```
void main()
{
    HMODULE hDll = NULL;
    PFN_HOOKSTART HookStart = NULL;
    PFN_HOOKSTOP HookStop = NULL;
    char ch = 0;

    //KeyHook.dll 로드
    hDll = LoadLibrary( dll 경로 );
    if (hDll == NULL) {
        printf("LoadLibrary(%s) failed!!! [%d]", DEF_DLL_NAME, GetLastError());
        return;
    }

    //export 함수(HookStart, HookStop)의 주소 얻기
    HookStart = (PFN_HOOKSTART)GetProcAddress(hDll, DEF_HOOKSTART);
    HookStop = (PFN_HOOKSTOP)GetProcAddress(hDll, DEF_HOOKSTOP);

    //후킹 시작
    HookStart();




    //q가 입력될 때까지 로더는 아무런 기능을 하지 않고 대기
    printf("press 'q' to quit!\n");
    while (_getch() != 'q');


    //후킹 종료
    HookStop();

    //KeyHook.dll 언로드
    FreeLibrary(hDll);
}
```

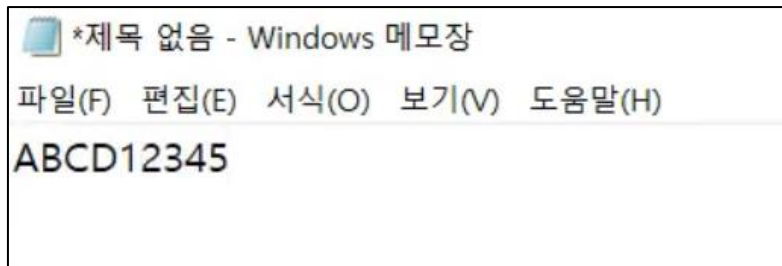
3. 윈도우 Hooking 실습 - 키보드 입력 Hooking

발생 예상 에러

	코드	설명	프로젝트	파일	줄
	E0167	"const char *" 형식의 인수가 "LPCWSTR" 형식의 매개 변수와 호환되지 않습니다.	Hookmain	KeyMain.cpp	22
	E0965	유니버설 문자 이름의 형식이 잘못되었습니다.	Hookmain	KeyMain.cpp	22
	C2664	'HMODULE LoadLibraryW(LPCWSTR)': 인수 1을(를) 'const char [60]'에서 'LPCWSTR'(으)로 변환할 수 없습니다.	Hookmain	KeyMain.cpp	22

	MSB802	v143에 대한 빌드 도구(플랫폼 도구 집합 = 'v143')를 찾을 수 없습니다. v143 빌드 도구를 사용하여 빌드하려면 v143 빌드 도구를 설치하십시오. [프로젝트] 메뉴를 선택하거나 솔루션을 마우스 오른쪽 단추로 클릭한 다음 "솔루션 대상 변경"을 선택하여 현재 Visual Studio 도구로 업그레이드할 수도 있습니다.	KeyHook	Microsoft.CppBuild.targets	439
---	------------------------	---	---------	----------------------------	-----

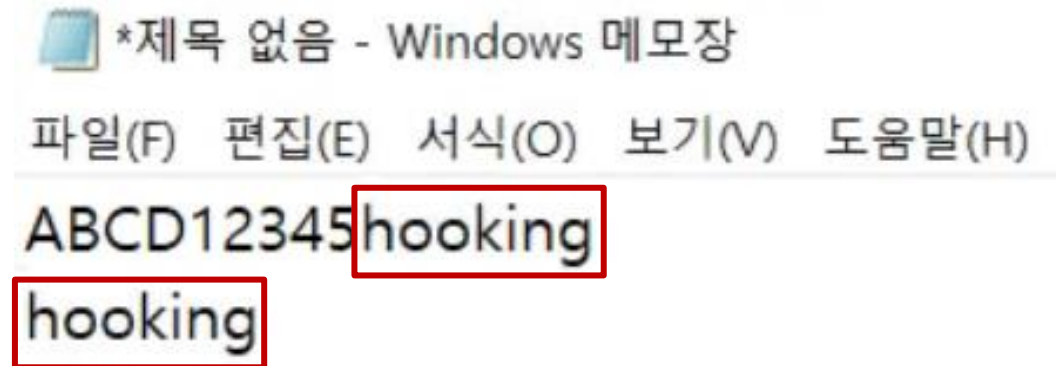
3. 윈도우 Hooking 실습 - 키보드 입력 Hooking



로그파일에도 동일한 문자열이 저장됨



3. 윈도우 Hooking 실습 - 키보드 입력 Hooking



Enter 입력 시, “hooking” 문자가 입력됨

4. 우분투 Hooking 실습 (1) puts 함수 Hooking

usingputs.c

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    puts("Test failed!\n");
    return 0;
}
```

Unix Standard Header

- Unix 시스템 호출 전반을 담당하는 헤더 파일
- 윈도우에서는 사용 불가

<stdio.h>의 출력 함수 puts를 통해 문자열 Test failed! 를 출력하는 코드

→ puts 함수를 Hooking하여 다른 문자열이 출력되도록 할 것

4. 우분투 Hooking 실습 (1) puts 함수 Hooking

puts_hooking.c

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <dlfcn.h>

int puts(const char *message){
    int (*new_puts)(const char *message);
    int result;
    new_puts = dlsym(RTLD_NEXT, "puts");
    if(strcmp(message, "Test failed!\n") == 0){
        result = new_puts("Test Success!\n");
    }
    else{
        result = new_puts(message);
    }
    return result;
}
```

UNIX 환경에서 동적 라이브러리를 로드하기 위한 API가 선언되어 있는 헤더

Hooking하고자 하는 함수와 동일한 이름, 파라미터를 가져야 함.

void *dlsym(void *handle, const char *symbol)

- <dlfcn.h>에 정의된 함수
- so 파일의 포인터 정보를 얻어 함수의 주소값을 알아내고 동적 라이브러리에 존재하는 함수와 매핑함
- 첫 번째 인수(RTLD_NEXT):라이브러리에서 다음으로 찾아오는 함수를 받아오는 handler
- 두 번째 인수: Hooking하고자 하는 함수의 이름

4. 우분투 Hooking 실습 (1) puts 함수 Hooking

1. usingput.c 컴파일

```
~/kcmvp/hooks$ gcc usingput.c -o usingputs
```

2. Hooking 파일의 공유 라이브러리(so) 생성

```
~/kcmvp/hooks$ gcc puts_hooking.c -o putshooking.so -fPIC -shared -ldl -D_GNU_SOURCE
```

-fPIC: 라이브러리 주소를 현재 사용하려는 프로세스 주소에서 접근할 수 있게 재배치 함.

-shared: 공유라이브러리 생성을 위함

-ldl: dlsym 함수를 이용하기 위해서 필요로 하는 option

-D_GNU_SOURCE: RTLD_NEXT를 사용하기 위함(RTLD_NEXT는 <dlfcn.h>에 정의되지 않았음)

3. LD_PRELOAD 환경변수 설정

```
~/kcmvp/hooks$ export LD_PRELOAD="/home/yj/kcmvp/hooks/putshooking.so"
```

LD_PRELOAD 환경변수는 동일한 이름의 함수가 있을 때, Hooking 라이브러리 주소를 먼저 가리키게 함
→ 기존의 라이브러리 로드 전에 지정한 라이브러리가 먼저 로딩되게 만들어 줌.

“=” 근처로 띄어쓰기를 하면 (적절한 식별자 아님) 오류가 발생함. 반드시 붙여서 사용해야 함.

4. 우분투 Hooking 실습 (1) puts 함수 Hooking

4. 프로그램(usingputs) 실행

```
yj@yj-VirtualBox:~/kcmvp/hooks$ ./usingputs
Test Success!
```

- 기존의 puts 함수 → 후킹한 puts 함수로 대체됨
- 후킹 함수에서 변조한 메시지 출력

후킹 해제

```
~/kcmvp/hooks$ unset LD_PRELOAD unset 명령어는 LD_PRELOAD를 해제해줌  
yj@yj-VirtualBox:~/kcmvp/hooks$ ./usingputs  
Test failed!
```

- 기존의 puts 함수가 포함된 공유 라이브러리 호출
- 기존 함수의 메시지 출력

4. 우분투 Hooking 실습 (2) openssl의 SSL_read 함수 Hooking

opensslhook.c

```
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>
#include <openssl/ssl.h> openssl을 제공하는 헤더

int SSL_read(SSL *context, void *buffer, int bytes)
{
    int (*new_ssl_read)(SSL *context, const void *buffer, int bytes);
    new_ssl_read = dlsym(RTLD_NEXT, "SSL_read");
    FILE *logfile = fopen("logfile", "a+");
    fprintf(logfile, "%s", (char *)buffer);
    fclose(logfile);
    return new_ssl_read(context, buffer, bytes);
}
```

openssl에서 제공하는 openssl의 SSL_read 함수를 Hooking

→ buffer에 담긴 데이터(서버에서 받아온 메시지)를 가져와 로그 파일에 기록

4. 우분투 Hooking 실습 (2) openssl의 SSL_read 함수 Hooking

1. Hooking 파일의 공유 라이브러리(opensslhook.so) 생성

```
$ gcc opensslhook.c -o opensslhook.so -fPIC -shared -lssl -D_GNU_SOURCE
```

2. LD_PRELOAD 환경변수 설정

```
$ export LD_PRELOAD="/home/yj/kcmvp/hooks/opensslhook.so"
```

3. curl 명령어를 통해 서버와 통신

```
yj@yj-VirtualBox:~$ curl https://gmail.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://mail.google.com/mail/u/0/">here</A>.
</BODY></HTML>
```

4. 우분투 Hooking 실습 (2) openssl의 SSL_read 함수 Hooking

4. buffer의 값을 로그로 저장(logfile)

```
yj@yj-VirtualBox:~$ ls
ARIA          helloworld      openssl-1.1.1d.tar.gz
KCMVP         helloworld.c    opensslhook.c
control.tar.xz libexample.c     opensslhook.so
data.tar.xz   logfile         snap
```

5. logfile 출력

```
yj@yj-VirtualBox:~$ cat logfile
Process 57950:
PRI * HTTP/2.0

SM

Process 57950:
```

감사합니다