Beist
Security Study Group

초보자를 위한 예제와 함께
배워보는 OllyDbg사용법 -2부

By Beist Security Study Group
(http://beist.org)

< contents >

0. Prolog

Ollydbg의 강력한 장점 중 하나인 plugin에 대해서 알아보겠습니다. Plugin은 Ollydbg의 사용을 보다 편리하게 도와주는 프로그램들을 말합니다. Plugin을 개발하기 위해서 Ollydbg에서 제공하는 Plugin Development Kit 1.10을 이용할 수 있습니다.
Ollydbg의 plugin은 dll형태로 만들어져 배포될 수 있습니다. 이 개발툴은 Ollydbg홈페이지

에서 제공되고 있으며, 이 개발툴에는 간단한 샘플 파일과 헤더파일과 라이브러리 등이 있습니다. 그리고 Ollydbg Plugin API와 구조체 등을 help file(plugins.hlp)로 제공 받을 수 있습니다.

1. 준비 과정

1.1 compiler setting

Ollydbg와 plugin 사이에 통신을 위해서 컴파일러에서 필요한 설정이 아래와 같이 있습니다.

-callback 함수는 이름으로 익스포트 해야 합니다.
-c++ 컴파일러에서는 name magling을 비활성 시켜야 합니다.
(extern "C"로 선언해주세요)
-callback함수의 호출규약은 __cdecl을 사용합니다.
-plugin.h에 모든 구조체들은 BYTE alignment되있습니다.
-기본 charactertype을 UNSIGNED로 설정해야합니다.

1.2 샘플 코드로 만들어보는 플러그인

홈페이지에서는 vc++ 5.0으로 작성된sample파일들이 있습니다.(plug110 ,plug110₩Vc50)
Sample plugin은 두 가지 프로그램이 있습니다. Ollydbg에서 command line으로 명령어를 실행시킬 수 있게 하는 Cmdline과 프로그램에서의 tagging기능을 가능케 하는 Bookmark가 있습니다. Sample plugin들을 vc++ 6.0에서 컴파일 해보는 방법에 대해서 설명하겠습니다. vc외에서의 컴파러(BC5.5 BCB5 VC5 Visual-Studio)에서의 컴파일 방법은 help file(plugins.hlp)에 자세히 기술되어있으므로 참고하시기 바랍니다.

1. 컴파일 할 sample과,공통 파일을 같은 폴더에 위치 복사합니다.
2. vc++ 6.0을 실행한 뒤 sample의 work space파일(.dsw)을 엽니다.
3. sample file(*.c)파일을 빌드 합니다.
4. 완성된 dll을 Ollydbg가 위치한 폴더 안에 위치시키고 실행시키면
   우리가 컴파일한 plugin이 로드 된 것을 볼 수 있습니다.

Bookmark project에 관련된 파일들
-bookmark.c
-bookmark.dsp

-bookmark.dsw
-bookmark.mak

Cmdline project에 관련된 파일들
-cmdexec.c
-command.c
-cmdline.dsp
-cmdline.dsw
-cmdline.mak

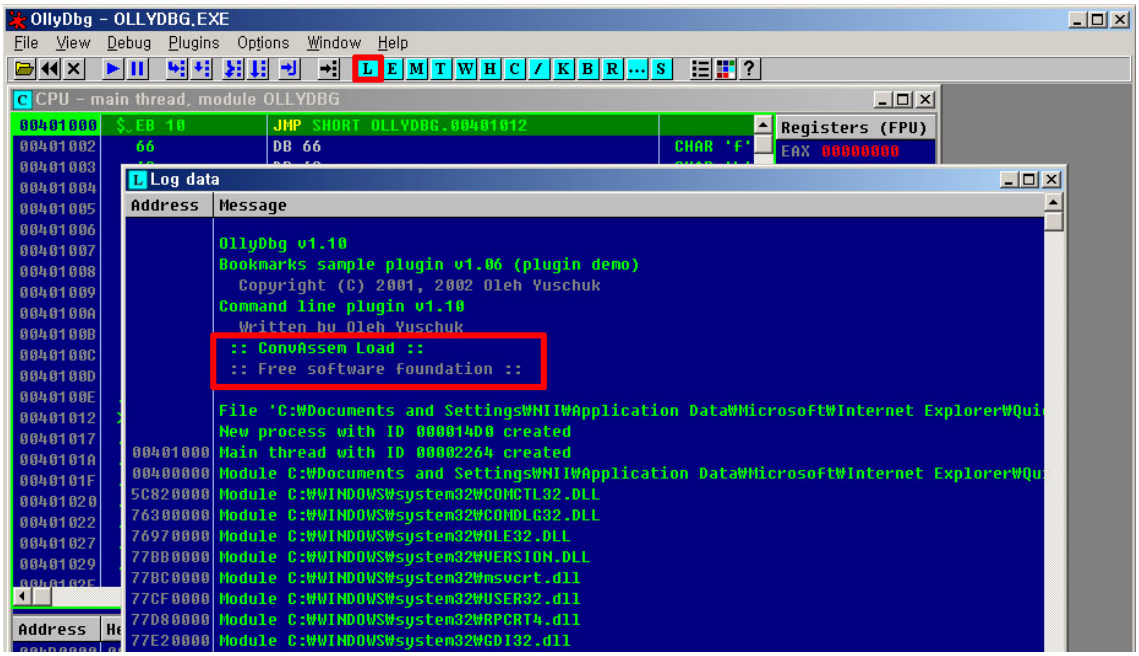
공통 파일들
plugin.h
ollydbg.lib

2. 간단한 plugin제작 및 분석

지금부터 직접 제작한 간단한 plugin 소스 코드를 보면서 plugin의 구조를 살펴 보겠습니다. 제작된 plugin은 사용자로부터 어셈블 코드를 입력 받고 바이너리 코드로 변환해 Dialog에 출력해 줍니다. 제작된 plugin 소스코드들은 첨부파일(Test_plug.zip)에 포함되어 있습니다.

```
extern int _export cdecl ODBG_Plugindata(char shortname[32]){
        //ODBG_Plugindata()와 ODBG_Plugininit()는 꼭 있어야 할 함수들입니다.
        //plugin이름을 받고 plugin 버전을 리턴 해주는 함수입니다.

        strcpy(shortname,"ConvAssem");//plugin이름
        return PLUGIN_VERSION;//plugin.h에 정의되어 있음.
}

extern int _export cdecl ODBG_Plugininit(int ollydbgversion,HWND hw,ulong *features){
        //초기화 작업과 리소스 할당을 하는 함수입니다.
        if (ollydbgversion<PLUGIN_VERSION)//Ollydbg와 plugin버전을 비교
                return -1; //plugin버전이 상위버전이면 -1을 리턴 시켜
                        //plugin이 로드 되지 않게 합니다.
        hWmain=hw;
```

```
        Addtolist(0,0, " :: ConvAssem Load :: ");//Addtolist()함수는 Log Window에
                                                  //문자열을 출력하는 함수입니다.
        Addtolist(0,-1," :: Free software foundation :: ");
        return 0;
}
```



[그림 1] Addtolist()로 Log Window에 출력한 모습

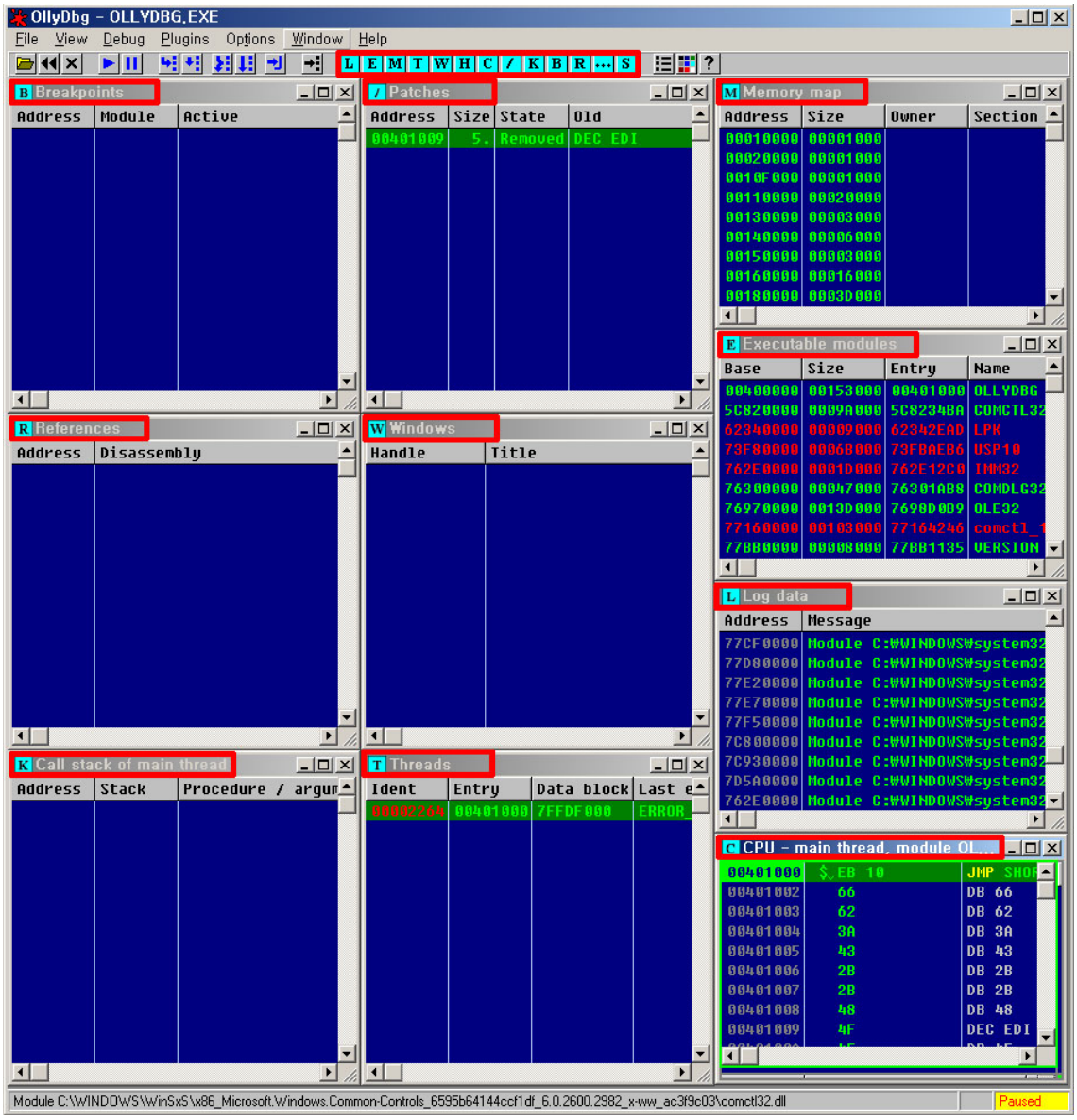
ODBG_Plugindata() ODBG_Plugininit() 이 두 함수는 plugin을 구성하기 위해서 가장 기본적인 함수입니다. Ollydbg에서 plugin으로 로드 되기 위해서 있어야 함수들입니다.

ODBG_Plugindata()는 plugin의 이름을 받고 plugin interface의 버전을 리턴 합니다. 여기서 리턴하는 버전은 plugin.h에 정의되어 있습니다.(PLUGIN_VERSION) 이 함수가 없거나 호환이 되지 않는 버전이라면 plugin이 설치되지 않습니다. 여기에 인자인 shortname은 plugin을 식별할 수 있는 이름입니다.

ODBG_Plugininit()는 plugin의 초기화 작업과 리소스 할당을 하는 함수입니다. 초기화 작업이 성공적으로 진행되면 함수는 0을 리턴하고, 에러가 있다면 -1을 리턴합니다. plugin의 버전과 Ollydbg의 버전을 비교해서 다르다면 -1 값을 ollydbg에 돌려줍니다. Ollydbg는 -1 값을 받는다면 plugin을 로드하지 않습니다.

```
extern int _export cdecl ODBG_Pluginmenu(int origin,char data[4096],void *item){
        //여러 윈도우의 메뉴를 설정해주는 함수입니다.
        int n;
        t_dump *pd;

        switch (origin) {
        case PM_MAIN://Main Window의 Menu를 의미합니다.
                strcpy(data,"0 &ConvAssem|1 &About");// [그림 3]
                return 1;                       //data에 메뉴 format을 넣습니다.

        case PM_DISASM:// CPU Disassembler에서 팝업메뉴를 의미합니다.
                pd=(t_dump *)item;
                if(pd==NULL || pd->size==0)
                        return 0;

                strcpy(data,"ConvAssem{0 Copy BaseAddr|1 ConvAssem}");
                //[그림 4] data에 메뉴 format을 넣습니다.
                return 1;

        default: break;
        }
        return 0;
}
```

Ollydbg는 아래 [그림 2]와 같이 프로그램의 여러 가지 정보를 출력하는 Widnow를 가지고 있습니다. 우리가 Window들에 메뉴나 팝업을 등록하기 위해서는 ODBG_Pluginmenu()라는 함수를 사용합니다.



[그림 2] Ollydbg의 여러 윈도우

ODBG_Pluginmenu()함수의 인자를 살펴보시면, origin은 Ollydbg의 여러 윈도우의 상수 값을 가집니다.

PM_MAIN Main window	
PM_DUMP	Any Dump window
PM_MODULES	Modules window
PM_MEMORY	Memory window
PM_THREADS	Threads window
PM_BREAKPOINTS	Breakpoints window

PM_REFERENCES	References window
PM_RTRACE	Run trace window
PM_WATCHES	Watches window
PM_WINDOWS	Windows window
PM_DISASM	CPU Disassembler
PM_CPUDUMP	CPU Dump
PM_CPUSTACK	CPU Stack
PM_CPUREGS	CPU Registers

---

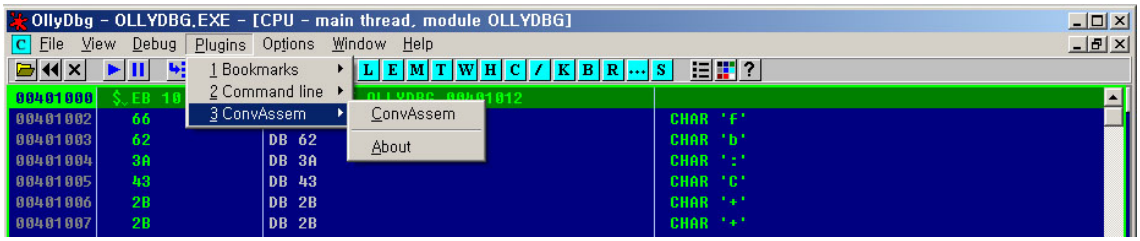
data는 메뉴와 팝업에 등록될 문자열을 포함합니다. 메뉴와 팝업에 등록되는 문자열은 각각 다른 format을 가집니다.

메뉴 format: 0 &ConvAssem|1 &About ·····

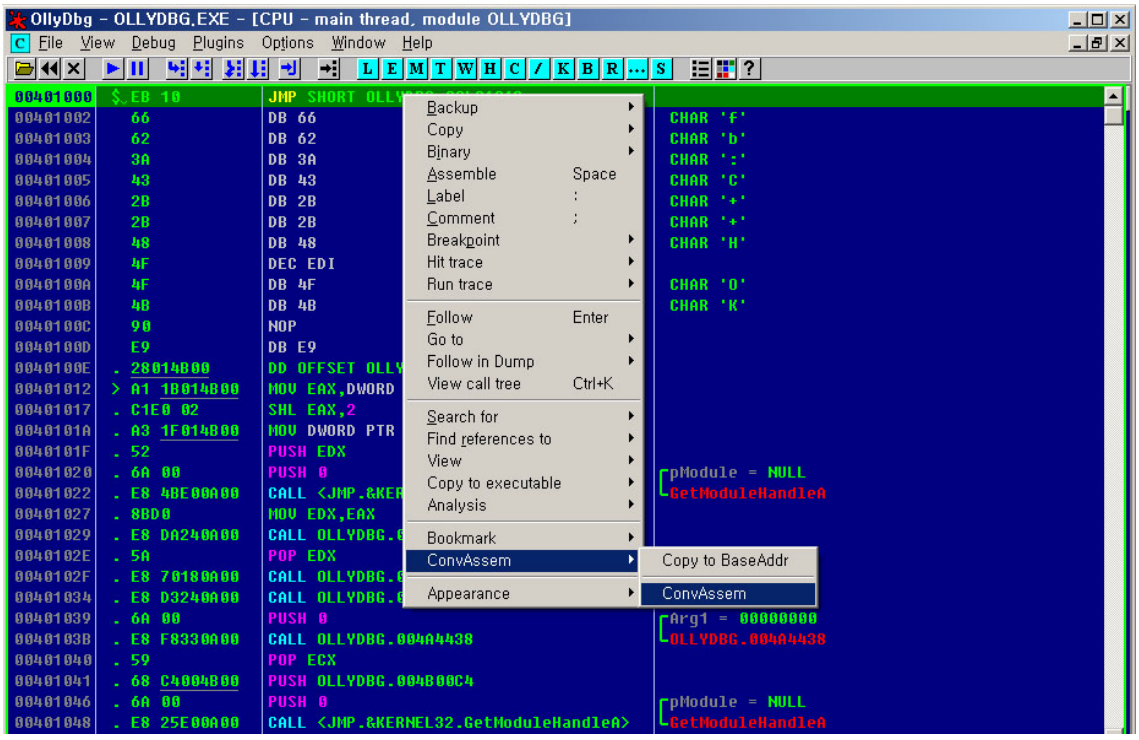
0,1은 메뉴들을 식별할 수 있는 ID이고, '&'뒤에 있는 문자가 단축키가 됩니다. 위 같은 경우에는 ConvAssem이란 메뉴의 'C'와 About메뉴의 'A'가 단축키가 됩니다. 그리고 '|'키는 메뉴 사이를 분리해놓는 역할을 합니다.

팝업 format: Z{0Aaa|1Bbb}

Z는 팝업에서의 메뉴이름을 의미합니다. 0,1는 메뉴들을 식별할 수 있는 ID이고 A, B는 Z메뉴에 속한 메뉴들입니다. 보통 Z에는 plugin이름이 들어갑니다.



[그림 3] PM_MAIN(Main Window)에 메뉴를 등록한 모습

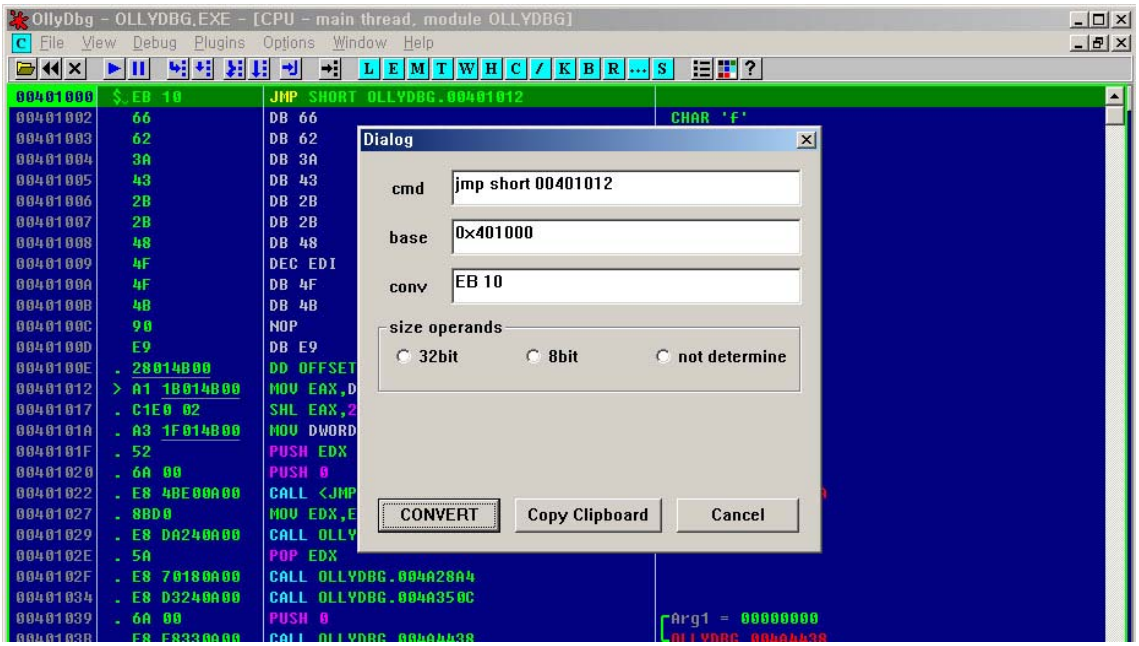


[그림 4] PM_DISASM(CPU Disassembler)에 팝업메뉴 등록한 모습.

```
extern void _export cdecl ODBG_Pluginaction(int origin,int action,void *item){
        //메뉴를 선택 할 때 마다, 메뉴에 해당하는 루틴을 설정해줍니다.
        t_dump *pd;
        char *pClipData;
        char tmp_buf[MAX_STRING];
        ulong ul_addr=0;

        memset(tmp_buf,0,sizeof(tmp_buf));
        h_Inst=(HINSTANCE)GetCurrentModule();
        //Dialog를 띄우기 위해서는 HMODULE이 필요합니다.
        //이 함수는 할당된 DLL내에서 HMODULE을 찾아주는 함수입니다.

        if (origin==PM_MAIN){// Main Window의 Menu를 의미합니다.
                switch(action){
                case 0://ConvAssem를 선택했을 때의 루틴입니다.
                        hDlg=CreateDialog(h_Inst,MAKEINTRESOURCE(IDD_CONV),hWm
                        ain,ConvDlgProc);
```

```
                ShowWindow(hDlg,SW_SHOW);
                break;
        case 1://About를 선택했을 때의 루틴입니다.
                MessageBox(hWmain,"ConvAssem Plugin","ConvAssem
                Plugin",MB_OK|MB_ICONINFORMATION);
                break;
        default: break;
        }
}else if(origin==PM_DISASM){// CPU Disassembler에서 팝업메뉴를 의미합니다.
        switch(action){
        case 0://Copy to BaseAddr을 선택했을 때의 루틴입니다.
                pd=(t_dump *)item;
                ul_addr=pd->sel0;//선택된 주소를 ul_addr에 저장합니다.
                hClipboard=GlobalAlloc(GMEM_MOVEABLE,256);
                pClipData=(char *)GlobalLock(hClipboard);
                wsprintf(tmp_buf,"0x%lx",ul_addr);
                strcpy(pClipData,tmp_buf);
                GlobalUnlock(hClipboard);
                strcat(tmp_buf," Copy to clipboard");

                if(OpenClipboard(0)){
                        EmptyClipboard();
                        SetClipboardData(CF_TEXT,hClipboard);
                        //선택된 base주소를 클립보드에 복사합니다.
                        CloseClipboard();
                        MessageBox(NULL,tmp_buf,"success",MB_OK);
                }
                break;

        case 1:
                hDlg=CreateDialog(h_Inst,MAKEINTRESOURCE(IDD_CONV),hWm
                ain,ConvDlgProc);
                ShowWindow(hDlg,SW_SHOW);
                break;

        }
```

```
        }
}
```



[그림 5] Dialog로 띄운 plugin

우리가 ODBG_Pluginmenu() 에서 정의한 메뉴나 팝업들을 선택할 때마다, ODBG_Pluginaction()이 호출됩니다. 이 함수에서 각각 메뉴에 해당하는 루틴을 처리 해줍니다. ODBG_Pluginaction()의 인자를 살펴보면, origin은 유저가 선택한 윈도우의 상수 값을 갖고, action은 에서 정의한 포맷의 ID를 갖습니다. 이 함수에서 CreateDialog()함수로 Dialog를 생성한 뒤, assemble기능을 구현해 줍니다. Ollydbg plugin API에서 제공하는 Assemble()이라는 함수를 이용하여 Assemble기능을 구현할 수 있습니다. Assemble의 소스코드는 Ollydbg 홈페이지에서 제공하는 Disasm.zip 파일에 포함되어 있습니다.

3. 마치는 말

지금까지 plugin을 작성, 컴파일하는 과정을 통해서 plugin과 Ollydbg가 어떻게 작동하는지, 그리고 어떻게 구성 되어있는지 알아보았습니다. 본 문서의 plugin은 기본적인 함수들만 사용해서 작성하였는데, 이 구조를 이용해서 다른 함수들을 추가하여 자신만의 plugin을 만들어 볼 수 있습니다. 그리고 Ollydbg plugin을 제공하는 사이트(대표적으로 Openrec)같은 경우에는 소스파일도 같이 첨부하고 있기 때문에 참고하시면 더 강력한 plugin을 제작하는데 도움이 될 수 있을 것입니다.

4. References

[1] Test_plug files, http://beist.org/research/public/beginnerolly2/test_plug.zip

[2] OllyDbg, http://www.ollydbg.de/

[3] Devpia, http://www.devpia.com

[4] Openrec, http://www.openrce.org/downloads/browse/OllyDbg_Plugins

[5] 80x66 Disassembler & Assembler, http://www.ollydbg.de/srcdescr.htm

[6] Codeguru, http://www.codeguru.com