

2nd CodeEngn Seminar

스타크래프트 맵핵 제작을 통해 알아보는 리버싱

발표자 : 강봉구

발표일 : 08.11.08

어디서부터 손을 대야 할까?

호랑이한테 물려가도 정신만 차리면
산채로 잡아먹는 고통을 느낄 수 있다!



1. 과연 기능 구현이 가능한가?

2. 변하는 값을 찾아라.

예) 맵이 다 보이면 TRUE, 아니면 FALSE ..., Enable ...

3. 값을 참조하는 부분을 찾아라.

맵을 밝혀주는 루틴이 주변에 있을 가능성이 높기 때문

4. 찾은 함수를 디버거로 분석하자.

맵을 밝혀주는 정확한 루틴을 찾아 분석하기 위함

5. 공격 목표 설정

맵핵 구현을 위해 어느 곳을 어떻게 수정할지를 결정

2. 변하는 값을 찾아라.



3. 값을 참조하는 부분을 찾아라.

The screenshot shows the OllyDbg interface. On the left, a memory dump table is visible with the following data:

Address	Size	Value
0064158D	4 Bytes	100
006D5A53	4 Bytes	0
006D5A54	4 Bytes	0
0064158C	4 Bytes	25701
013906CC	Text(13)	Cheat Enabled

A context menu is open over the selected address 013906CC, with the option "Find out what reads from this address" highlighted. The main window displays assembly code with the instruction "6C 65 64 00 43 68 Cheat Enabled Ch" highlighted in red. A dialog box titled "The following opcodes read from the selected ..." is open, showing the following opcodes:

- 1502ac8a - 8b 0a - mov ecx,[edx]
- 1502acbd - 8b 0a - mov ecx,[edx]

The dialog box also contains buttons for "Replace", "Show disassembler", "Add to the codelist", "More information", and "Stop".

4. 찾은 함수를 디버거로 분석하자.

```
DebOilly - StarCraft.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
CPU - main thread, module StarCraf
004807D0 55 PUSH EBP
004807D1 8BEC MOV EBP,ESP
004807D3 83EC 10 SUB ESP,10
004807D6 53 PUSH EBX
004807D7 56 PUSH ESI
004807D8 57 PUSH EDI
004807D9 E8 72F7FFFF CALL StarCraf.0047FF50
004807DE A1 74846200 MOV EAX,DWORD PTR DS:[628474]
004807E3 8B00 90846200 MOV ECX,DWORD PTR DS:[628490]
004807E9 8B3D F05B6D00 MOV EDI,DWORD PTR DS:[6D5BF0]
004807EF 8B35 F45B6D00 MOV ESI,DWORD PTR DS:[6D5BF4]
004807F5 83E0 1F AND EAX,1F
004807F8 BB C0020000 MOV EBX,2C0
004807FD 2BD8 SUB EBX,EAX
004807FF 83E1 1F AND ECX,1F
00480802 B8 E0010000 MOV EAX,1E0
00480807 2BC1 SUB EAX,ECX
00480809 8D90 20FEFFFF LEA EDX,DWORD PTR DS:[EAX-1E0]
0048080F 83C7 19 ADD EDI,19
00480812 83C6 19 ADD ESI,19
00480815 3BD0 CMP EDX,EAX
00480817 8945 F0 MOV DWORD PTR SS:[EBP-10],EAX
0048081A 8955 F4 MOV DWORD PTR SS:[EBP-C],EDX
0048081D 0F8D 98000000 JGE StarCraf.004808BB
00480823 8D8B 40FDFFFF LEA ECX,DWORD PTR DS:[EBX-2C0]
00480829 3BCB CMP ECX,EBX
0048082B 894D FC MOV DWORD PTR SS:[EBP-4],ECX
0048082E 7D 77 JGE SHORT StarCraf.004808A7
00480830 8007 MOV AL,BYTE PTR DS:[EDI]
```

kernel32.GetCurrentProcessId

전체맵을 그려주는 부분

kernel32.GetCurrentProcessId

4. 찾은 함수를 디버거로 분석하자.

Address	Hex dump	Disassembly	Comment
00480006	EB 2E	JMP SHORT StarCraf.00480036	
00480008	8A15 847A6500	MOV DL, BYTE PTR DS:[657A84]	
0048000E	8810	MOV BYTE PTR DS:[EAX], DL	
00480010	EB 24	JMP SHORT StarCraf.00480036	
00480012	8B0F	MOV ECX, DWORD PTR DS:[EDI]	
00480014	8B15 B8EE5700	MOV EDX, DWORD PTR DS:[57EEB8]	
0048001A	85CA	TEST EDX, ECX	
0048001C	74 05	JE SHORT StarCraf.00480023	맵핵 패치 전
0048001E	C600 00	MOV BYTE PTR DS:[EAX], 0	
00480021	EB 13	JMP SHORT StarCraf.00480006	
00480023	8B15 94F05700	MOV EDX, DWORD PTR DS:0048000E	
00480029	85CA	TEST EDX, ECX	
0048002B	74 07	JE SHORT StarCraf.00480012	
0048002D	8A4D FF	MOV CL, BYTE PTR SS:[00480014]	
00480030	8808	MOV BYTE PTR DS:[EAX], CL	
00480032	EB 02	JMP SHORT StarCraf.0048001A	
00480034	8818	MOV BYTE PTR DS:[EAX], BL	
00480036	66:8B0D BCF157	MOV CX, WORD PTR DS:[0048001E]	
0048003D	0FB7D1	MOUZX EDX, CX	
00480040	4A	DEC EDX	
00480041	8BDE	MOV EBX, ESI	
00480043	46	INC ESI	
00480044	3BDA	CMP EBX, EDX	
00480046	73 03	JNE SHORT StarCraf.00480048	
00480048	83C7 04	ADD EDI, 4	
0048004B	8B55 F0	MOV EDX, DWORD PTR SS:00480036	
0048004E	40	INC EAX	
0048004F	4A	DEC EDX	
00480050	8955 F0	MOV DWORD PTR SS:[EBP], EDX	
00480053	^ 0F85 73FFFFFF	JNZ StarCraf.0047FFC1	
00480059	0FB715 BEF15700	MOUZX EDX, WORD PTR DS:00480044	
00480060	8B45 EC	MOV EAX, DWORD PTR SS:00480046	
00480063	4A	DEC EDX	
00480064	8BF0	MOV ESI, EAX	
00480066	40	INC EAX	
00480036=StarCraf.00480036			
0048004F			ntdll.KiFastSystemCallRet
00480050			ntdll.KiFastSystemCallRet
00480053			ntdll.KiFastSystemCallRet
00480059			ntdll.KiFastSystemCallRet
00480060			ntdll.KiFastSystemCallRet
00480063			ntdll.KiFastSystemCallRet
00480064			ntdll.KiFastSystemCallRet
00480066			ntdll.KiFastSystemCallRet
0048002D=StarCraf.0048002D			

Address	Hex dump	ASCII
00480022	0A 8B 15 94 F0 57 00 85 CA 74 07 8A 4D FF 88 08	.?B.M.2.t-5 ?
00480032	EB 02 88 18 66 8B 0D BC F1 57 00 0F B7 D1 4A 8B	??f?B.W.*J
00480042	DE 46 3B DA 73 03 83 C7 04 8B 55 F0 40 4A 89 55	?;?B+J?J

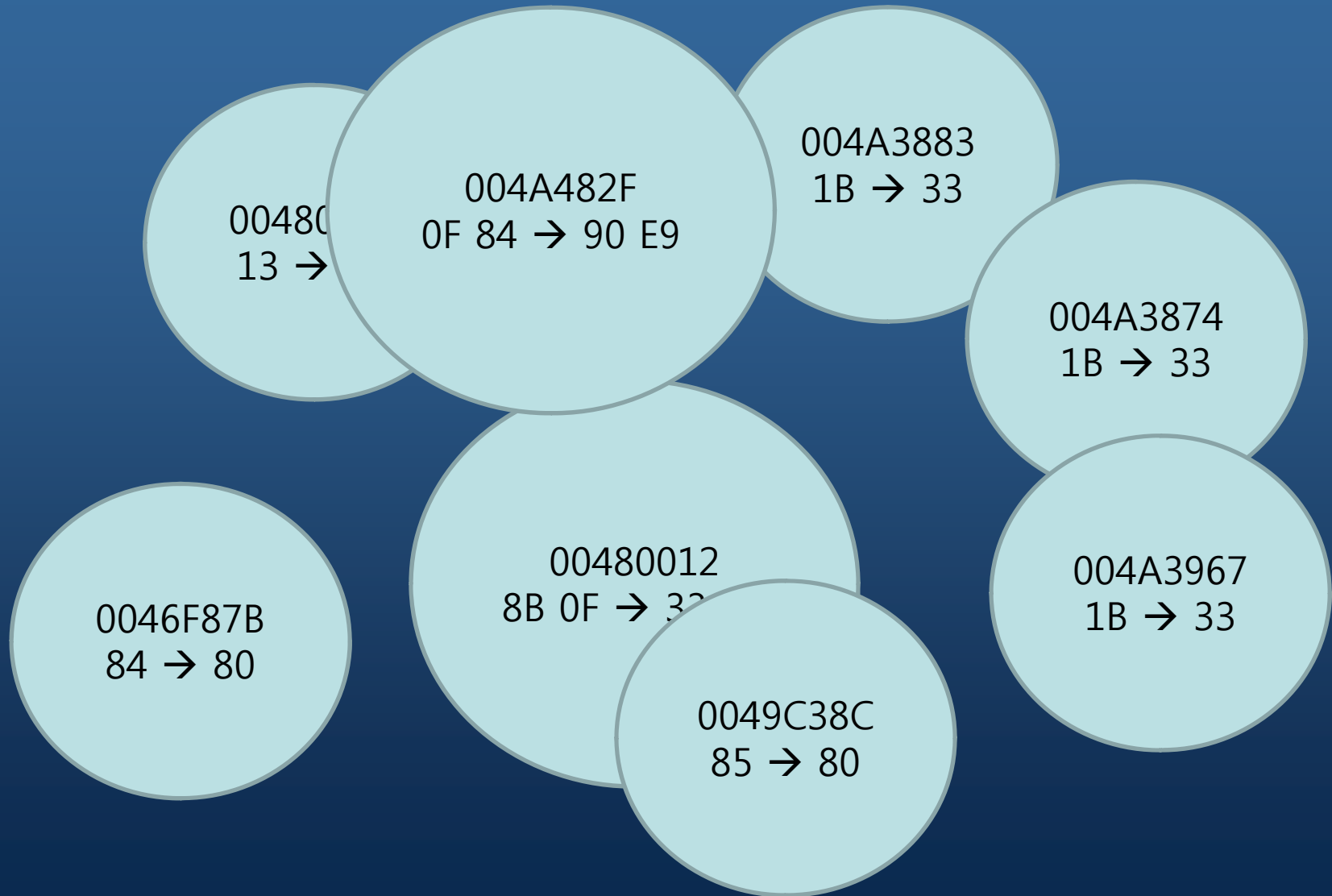
4. 찾은 함수를 디버거로 분석하자.

004A3820	55	PUSH EBP	// 미니 트를 그려주는 부분
004A3821	8BEC	MOV EBP,ESP	
004A3823	83EC 08	SUB ESP,8	
004A3826	A1 80C15900	MOV EAX,DWORD PTR DS:[59C180]	
004A382B	0FB70D 5CCC5900	MOVZX ECX,WORD PTR DS:[59CC5C]	
004A3832	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
004A3835	0FB705 4CCC5900	MOVZX EAX,WORD PTR DS:[59CC4C]	
004A383C	0FAFC1	IMUL EAX,ECX	
004A383F	85C0	TEST EAX,EAX	
004A3841	56	PUSH ESI	
004A3842	8B35 48126D00	MOV ESI,DWORD PTR DS:[6D1248]	
004A3848	57	PUSH EDI	
004A3849	8B3D 78C15900	MOV EDI,DWORD PTR DS:[59C178]	
004A384F	✓ 0F84 94000000	JE StarCraf.004A38E9	
004A3855	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
004A3858	53	PUSH EBX	kernel32.GetCurrentProcessId
004A3859	8DA424 00000000	LEA ESP,DWORD PTR SS:[ESP]	
004A3860	8B16	MOV EDX,DWORD PTR DS:[ESI]	// 미니 트 그리기
004A3862	8B0D 94F05700	MOV ECX,DWORD PTR DS:[57F094]	
004A3868	8B1D B8EE5700	MOV EBX,DWORD PTR DS:[57EEB8]	
004A386E	8BC2	MOV EAX,EDX	
004A3870	23C1	AND EAX,ECX	
004A3872	F7D8	NEG EAX	// 2의 보수 (부호 반전)
004A3874	1BC0	SBB EAX,EAX	// = sub (캐리 포괄과 로스)
004A3876	8BCA	MOV ECX,EDX	
004A3878	23CB	AND ECX,EBX	kernel32.GetCurrentProcessId
004A387A	8B1D FC0E6D00	MOV EBX,DWORD PTR DS:[6D0EFC]	
004A3880	40	INC EAX	
004A3881	F7D9	NEG ECX	
004A3883	1BC9	SBB ECX,ECX	// ecx를 값상 0으로 고정하면 미니 트 가리지의 보이게 함
004A3885	41	INC ECX	
004A3886	85DB	TEST EBX,EBX	kernel32.GetCurrentProcessId
004A3888	✓ 74 32	JE SHORT StarCraf.004A38BC	
004A388A	A1 040F6D00	MOV EAX,DWORD PTR DS:[6D0F04]	
004A388F	33C9	XOR ECX,ECX	
004A3891	85C0	TEST EAX,EAX	

4. 찾은 함수를 디버거로 분석하자.

0046F7F0	55	PUSH EBP	
0046F7F1	8BEC	MOV EBP,ESP	
0046F7F3	83EC 14	SUB ESP,14	
0046F7F6	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
0046F7F9	53	PUSH EBX	
0046F7FA	56	PUSH ESI	
0046F7FB	50	PUSH EAX	
0046F7FC	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
0046F7FF	33DB	XOR EBX,EBX	
0046F801	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
0046F804	C745 F8 FFFFFFFF	MOV DWORD PTR SS:[EBP-8],7FFFFFFF	
0046F80B	E8 4018FCFF	CALL StarCraf.00431050	
0046F810	8B30	MOV ESI,DWORD PTR DS:[EAX]	
0046F812	85F6	TEST ESI,ESI	
0046F814	8945 F0	MOV DWORD PTR SS:[EBP-10],EAX	
0046F817	✓ 0F84 80010000	JE StarCraf.0046F99D	
0046F81D	57	PUSH EDI	
0046F81E	8BFF	MOV EDI,EDI	
0046F820	0FB74E 64	MOUZX ECX,WORD PTR DS:[ESI+64]	
0046F824	F6048D 68406600	TEST BYTE PTR DS:[ECX*4+664068],10	
0046F82C	✓ 74 03	JE SHORT StarCraf.0046F831	
0046F82E	8B76 70	MOV ESI,DWORD PTR DS:[ESI+70]	
0046F831	A1 FC0E6D00	MOV EAX,DWORD PTR DS:[6D0EFC]	
0046F836	85C0	TEST EAX,EAX	
0046F838	8B7E 0C	MOV EDI,DWORD PTR DS:[ESI+C]	
0046F83B	8A4F 0C	MOV CL,BYTE PTR DS:[EDI+C]	
0046F83E	✓ 74 33	JE SHORT StarCraf.0046F873	
0046F840	A1 000F6D00	MOV EAX,DWORD PTR DS:[6D0F00]	
0046F845	84C8	TEST AL,CL	
0046F847	✓ 75 0E	JNZ SHORT StarCraf.0046F857	
0046F849	8B0D 040F6D00	MOV ECX,DWORD PTR DS:[6D0F04]	
0046F84F	85C9	TEST ECX,ECX	
0046F851	✓ 0F84 2E010000	JE StarCraf.0046F985	
0046F857	8B8E DC000000	MOV ECX,DWORD PTR DS:[ESI+DC]	
0046F85D	F6C5 03	TEST CH,3	
0046F860	✓ 74 35	JE SHORT StarCraf.0046F897	
0046F862	8586 E4000000	TEST DWORD PTR DS:[ESI+E4],EAX	
0046F868	✓ 75 2D	JNZ SHORT StarCraf.0046F897	
0046F86A	A1 040F6D00	MOV EAX,DWORD PTR DS:[6D0F04]	
0046F86F	85C0	TEST EAX,EAX	
0046F871	✓ EB 1E	JMP SHORT StarCraf.0046F891	
0046F873	A1 94F05700	MOV EAX,DWORD PTR DS:[57F094]	
0046F878	84C8	TEST AL,CL	
0046F87A	✓ 0F84 05010000	JE StarCraf.0046F985	// NOP -> Abuse
0046F880	8B8E DC000000	MOV ECX,DWORD PTR DS:[ESI+DC]	
0046F886	F6C5 03	TEST CH,3	
0046F889	✓ 74 0C	JE SHORT StarCraf.0046F897	
0046F88B	8586 E4000000	TEST DWORD PTR DS:[ESI+E4],EAX	
0046F891	✓ 0F84 EE000000	JE StarCraf.0046F985	

5. 공격 목표 설정



서
오
!

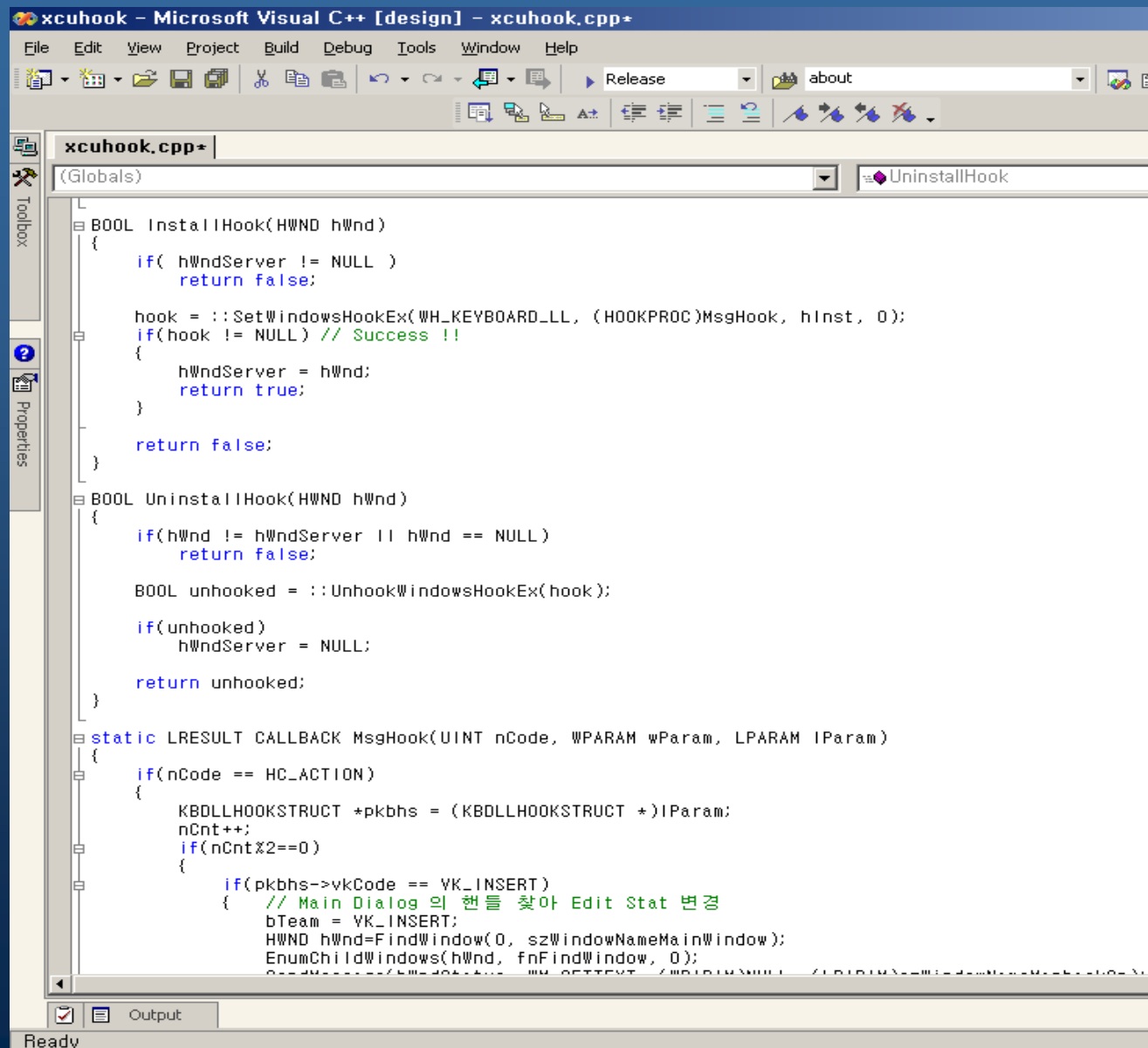
전역 키
훅할 DLL

대상프로세스
(Starcraft)에
삽입되어 코드
패치할 DLL

DLL 인젝션
& 추가기능
수행할 로더



전역 키 훅



```
xcuhook - Microsoft Visual C++ [design] - xcuhook.cpp*
File Edit View Project Build Debug Tools Window Help
Release about
xcuhook.cpp*
(Globals) UninstallHook
BOOL InstallHook(HWND hWnd)
{
    if( hWndServer != NULL )
        return false;

    hook = ::SetWindowsHookEx(WH_KEYBOARD_LL, (HOOKPROC)MsgHook, hInst, 0);
    if(hook != NULL) // Success !!
    {
        hWndServer = hWnd;
        return true;
    }

    return false;
}

BOOL UninstallHook(HWND hWnd)
{
    if(hWnd != hWndServer || hWnd == NULL)
        return false;

    BOOL unhooked = ::UnhookWindowsHookEx(hook);

    if(unhooked)
        hWndServer = NULL;

    return unhooked;
}

static LRESULT CALLBACK MsgHook(UINT nCode, WPARAM wParam, LPARAM lParam)
{
    if(nCode == HC_ACTION)
    {
        KBDLLHOOKSTRUCT *pkbhs = (KBDLLHOOKSTRUCT *)lParam;
        nCnt++;
        if(nCnt%2==0)
        {
            if(pkbhs->vkCode == VK_INSERT)
            {
                // Main Dialog 의 핸들 찾아 Edit Stat 변경
                bTeam = VK_INSERT;
                HWND hWnd=FindWindow(0, szWindowNameMainWindow);
                EnumChildWindows(hWnd, fnFindWindow, 0);
                HWND hWndEdit=FindWindow(hWnd, "EDIT");
                if(hWndEdit != NULL)
                {
                    SendMessage(hWndEdit, WM_SETTEXT, 0, (LPARAM)szTeam);
                }
            }
        }
    }
    return CallNextHookEx(NULL, nCode, wParam, lParam);
}
```

Ready

```

char    szProcName[MAX_PATH] = "Starcraft.exe";
BOOL    bForceChange = TRUE;

struct StructEd:t
{
    long m_lStartAddr;
    long m_lEndAddr;
    int  m_nChangeSize;
    unsigned char m_szFindString[MAX_BUF];
    unsigned char m_szChangeString[MAX_BUF];
} structedit[MAX_ARR];

#define fnSetStruct()
{
    int idx=0;
    structedit[idx].m_lStartAddr           = 0x004A482F;
    structedit[idx].m_nChangeSize         = 2;
    sprintf((char*)structedit[idx].m_szChangeString, "%x90%xE9");
    idx++;
    structedit[idx].m_lStartAddr           = 0x0049C38C;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x80");
    idx++;
    structedit[idx].m_lStartAddr           = 0x00480022;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x0a");
    idx++;
    structedit[idx].m_lStartAddr           = 0x004A3795;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x33");
    idx++;
    structedit[idx].m_lStartAddr           = 0x004A3883;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x33");
    idx++;
    structedit[idx].m_lStartAddr           = 0x004A3974;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x33");
    idx++;
    structedit[idx].m_lStartAddr           = 0x0046F5BD;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x80");
    idx++;
    structedit[idx].m_lStartAddr           = 0x0046F87B;
    structedit[idx].m_nChangeSize         = 1;
    sprintf((char*)structedit[idx].m_szChangeString, "%x80");
}

```

코드 패치할 DLL – 패치하는 함수

```
BOOL fnForceChangeCode(long lTargetAddr, unsigned const char *szForceChangeString, int nPatchSize, int cnt)
{
    int    i, nDumpSize;
    DWORD  dwOld;

    nDumpSize = ((int)((sizeof(szForceChangeString)-1)/16)+1)*16;

    OutputDebugString("----- 수정전 덤프 -----");
    dumpcode((unsigned char*)lTargetAddr, nDumpSize);
    VirtualProtect((void *)lTargetAddr, nPatchSize, PAGE_EXECUTE_READWRITE, &dwOld);

    for(i=0; i<nPatchSize; i++)
    {
        *(char*)(lTargetAddr+i) = szForceChangeString[i];
    }

    VirtualProtect((void *)lTargetAddr, nPatchSize, PAGE_EXECUTE_READ, &dwOld);

    OutputDebugString("===== 수정!! 後!! 덤프 =====");
    dumpcode((unsigned char*)lTargetAddr, nDumpSize);

    return TRUE;
}
```


로더 - 스타크래프트 프로세스 실행

```
STARTUPINFO si;
PROCESS_INFORMATION pi;
memset(&si, 0, sizeof(si));
memset(&pi, 0, sizeof(pi));
si.cb = sizeof(si);

hWnd = AfxGetMainWnd()->GetSafeHwnd();

if(fnGetProgramPath(szPathFile)==FALSE)
{
    Print2(hWnd, szWindowNameNotInstalled);
    return;
}

fnGetPathDir(szPathFile, szPathDir);

if(!CreateProcess(szPathFile, NULL, NULL, NULL, FALSE, CREATE_NEW_CONSOLE,
    /*+CREATE_SUSPENDED*/ NULL, szPathDir, &si, &pi))
{
    Print2(hWnd, "CreateProcess() Error");
    return;
}

pThread = AfxBeginThread(fnThreadFunc, NULL, THREAD_PRIORITY_NORMAL, 0, CREATE_SUSPENDED, NULL);
pThread->m_bAutoDelete = FALSE;
pThread->ResumeThread();
```

로더 - 코드 패치할 DLL 인젝션

```
hld = OpenProcess(PROCESS_ALL_ACCESS, FALSE, (DWORD)hProcessID);
if(!hld)
{
    Print2(hWnd, "DLL Inject Fail : OpenProcess() ERROR");
    return 1;
}

hMod = LoadLibrary("kernel32.dll");
if(!hMod)
{
    Print2(hWnd, "Dll Inect Fail : LoadLibrary() ERROR");
    CloseHandle(hld);
    return 1;
}

addrLoadLibrary = GetProcAddress(hMod, "LoadLibraryA");

nLen = (int)strlen(szDllPathFile);
lpStr = (char*)VirtualAllocEx(hld, NULL, nLen, MEM_COMMIT, PAGE_READWRITE);
if(!lpStr)
{
    Print2(hWnd, "Dll Inect Fail : VirtualAllocEx() ERROR");
    return 1;
}

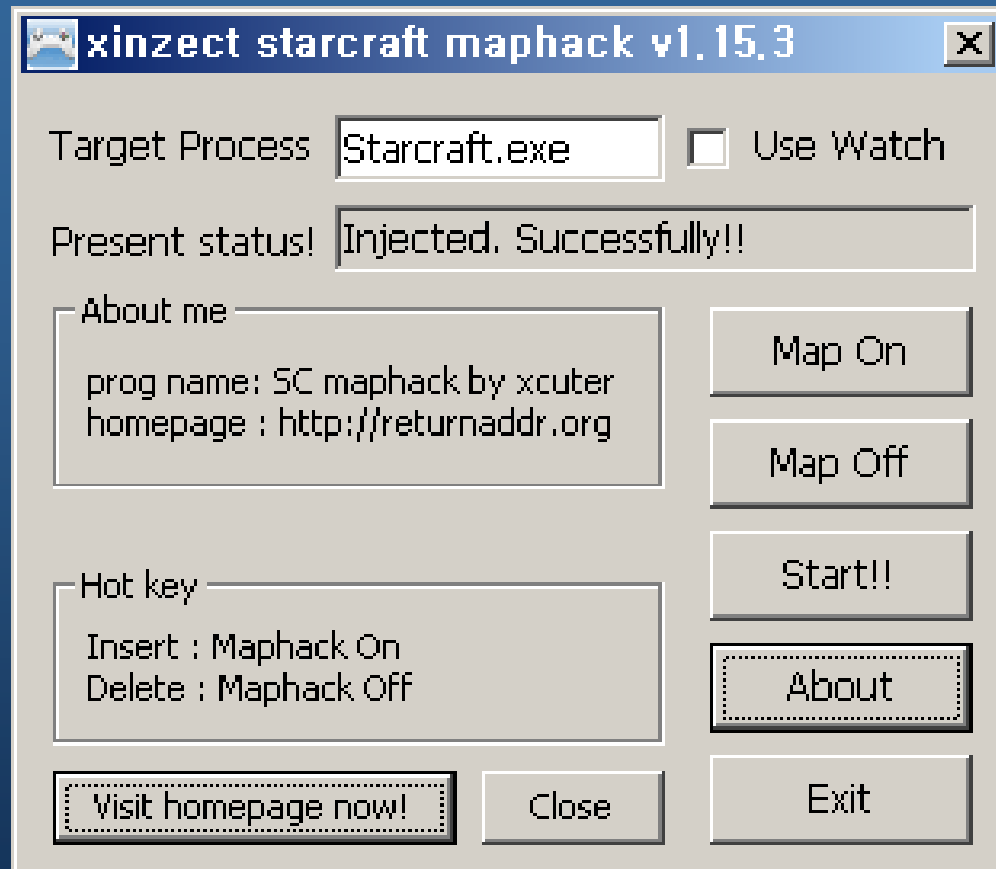
VirtualProtectEx(hld, (void *)lpStr, nLen, PAGE_EXECUTE_READWRITE, NULL);
WriteProcessMemory(hld, (void *)lpStr, (void *)szDllPathFile, nLen, &dwsz);

VirtualProtectEx(hld, (void *)lpStr, nLen, PAGE_READWRITE, NULL);

VirtualFreeEx(hld, (void *)lpStr, nLen, MEM_RELEASE);
hThread=CreateRemoteThread(hld, NULL, NULL, (LPTHREAD_START_ROUTINE)addrLoadLibrary, (void *)lpStr, NULL, NULL);

WaitForSingleObject(pThread->m_hThread, INFINITE);
CloseHandle(pThread->m_hThread);
fnProgramRun();
```

로더 – 완성된 로더



2nd CodeEngn Seminar

감사합니다.

xcuter

www.returnaddr.org