

## Advance RCE L03

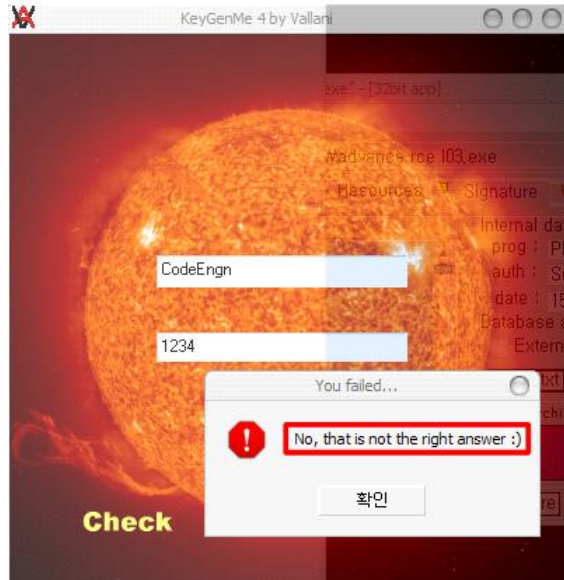
2010년 9월 21일 화요일

오전 1:22

### 파일 확인

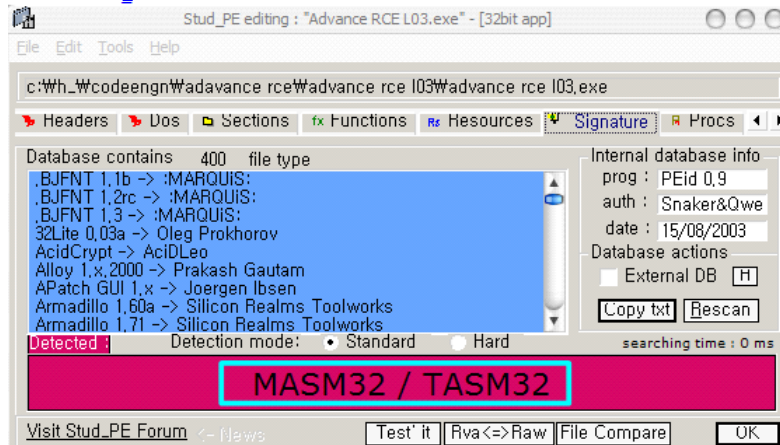


### 프로그램 실행



2개의 TextBox 에 값을 입력 받아서 name 에 따른 Serial 을 생성하여 검사하는 프로그램이다.

### With Stud\_PE



MASM32 로 이루어져 있으므로, Assembly 언어로 보기에는 가독성이 쉬울 것이라 예측

### With Ollydbg

Address	Hex dump	Disassembly	Comment
00401000	6A 00	PUSH 0	pModule = NULL
00401002	E8 AD030000	CALL <JMP.&kernel32.GetModuleHandleA>	GetModuleHandleA
00401007	A3 30324000	MOV DWORD PTR DS:[403230],EAX	
0040100C	6A 00	PUSH 0	lParam = NULL
0040100E	68 2C104000	PUSH Advance_.0040102C	DlgProc = Advance_.0040102C
00401013	6A 00	PUSH 0	hOwner = NULL
00401015	68 E9030000	PUSH 3E9	pTemplate = 3E9
0040101A	FF35 30324000	PUSH DWORD PTR DS:[403230]	hInst = NULL
00401020	E8 A7030000	CALL <JMP.&user32.DialogBoxParamA>	DialogBoxParamA
00401025	6A 00	PUSH 0	ExitCode = 0
00401027	E8 82030000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess

Packing 은 되지 않았으며, Handle 을 받아, DialogBox 를 출력하고 프로그램을 종료하는 프로그램이다.

Search For -> Name in current modules 에서 Text Box 에서 값을 읽어들이는 GetDlgItemTextA 함수를 확인하여 가보았다.

Address	Section	Type	Name
00402018	.rdata	Import	user32.DialogBoxParamA
0040201C	.rdata	Import	user32.EndDialog
00402020	.rdata	Import	kernel32.ExitProcess
00402024	.rdata	Import	user32.GetDlgItemTextA
00402028	.rdata	Import	kernel32.GetModuleHandleA
0040202C	.rdata	Import	user32.GetWindowLongA
00402030	.rdata	Import	user32.LoadIconA
00402034	.rdata	Import	kernel32.lstrcpA
00402038	.rdata	Import	user32.MessageBoxA
0040203C	.text	Export	<ModuleEntryPoint>
00402040	.rdata	Import	winmm.PlaySoundA
00402044	.rdata	Import	user32.SendMessageA
00402048	.rdata	Import	user32.SetFocus
0040204C	.rdata	Import	user32.SetLayeredWindowAttributes
00402050	.rdata	Import	user32.SetWindowLongA
00402054	.rdata	Import	kernel32.VirtualProtect
00402058	.rdata	Import	user32.wsprintfA

2개의 GetDlgItemTextA 함수가 보였고, 하나의 Box 를 읽어 들일때마다 MessageBox 가 출력될수 있는 가능성을 확인하였다

00401111	. 6A 20	PUSH 20	Count = 20 (32.)
00401113	. 68 38324000	PUSH Advance_.00403238	Buffer = Advance_.00403238
00401118	. 68 EC030000	PUSH 3EC	ControlID = 3EC (1004.)
0040111D	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401120	. E8 B9020000	CALL <JMP.&user32.GetDlgItemTextA>	GetDlgItemTextA
00401125	. 83F8 03	CMPL EAX,3	
00401128	. 73 18	JNB SHORT Advance_.00401142	
0040112A	. 6A 10	PUSH 10	Style = MB_OK MB_ICONHAND MB_APPLMODAL
0040112C	. 68 16314000	PUSH Advance_.00403116	Title = "You failed..."
00401131	. 68 F1304000	PUSH Advance_.004030F1	Text = "No, that is not the right answer :)"
00401136	. 6A 00	PUSH 0	hOwner = NULL
00401138	. E8 B0020000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
0040113D	. E9 8F000000	JMP Advance_.004011D1	
00401142	. B8 38324000	MOV EAX,Advance_.00403238	
00401147	. A3 58324000	MOV DWORD PTR DS:[403258],EAX	
0040114C	. 6A 00	PUSH 0	
0040114E	. E8 24010000	CALL Advance_.00401277	
00401153	. 6A 20	PUSH 20	Count = 20 (32.)
00401155	. 68 64324000	PUSH Advance_.00403264	Buffer = Advance_.00403264
0040115A	. 68 ED030000	PUSH 3ED	ControlID = 3ED (1005.)
0040115F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401162	. E8 77020000	CALL <JMP.&user32.GetDlgItemTextA>	GetDlgItemTextA
00401167	. FF35 00304000	PUSH DWORD PTR DS:[403000]	<%u> = 0
0040116D	. 68 84304000	PUSH Advance_.00403084	Format = "%u"
00401172	. 68 84324000	PUSH Advance_.00403284	s = Advance_.00403284
00401177	. E8 4A020000	CALL <JMP.&user32.wsprintfA>	wsprintfA
0040117C	. 83C4 0C	ADD ESP,0C	
0040117F	. 33C0	XOR EAX,EAX	
00401181	. A3 00304000	MOV DWORD PTR DS:[403000],EAX	
00401186	. 892D 5C324000	MOV DWORD PTR DS:[40325C],EBP	
0040118C	. 68 64324000	PUSH Advance_.00403264	String2 = ""
00401191	. 68 84324000	PUSH Advance_.00403284	String1 = ""
00401196	. E8 25020000	CALL <JMP.&kernel32.lstrcpA>	lstrcpA

여기서 EAX 와 3 을 비교하여 JMP 를 수행하게 되는데, 이는 Name 값이 3글자 이상이어야 함을 가르쳐 준다.

- 아래에 보면 lstrcpA 함수가 보이는데, 이를 이용하여 Serial 값과 사용자가 입력한 값을 비교하여 분기함을 예측할 수 있다.

2번째 GetDlgItemA 함수 에 Breakpoint 설정 후 확인

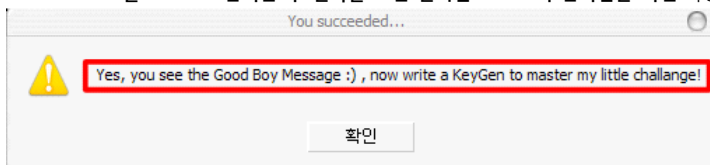
00401155	. 68 64324000	PUSH Advance_.00403264	Buffer = Advance_.00403264
0040115A	. 68 ED030000	PUSH 3ED	ControlID = 3ED (1005.)
0040115F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401162	. E8 77020000	CALL <JMP.&user32.GetDlgItemTextA>	GetDlgItemTextA
00401167	. FF35 00304000	PUSH DWORD PTR DS:[403000]	<%u> = C2A776FA (3265754874.)
0040116D	. 68 84304000	PUSH Advance_.00403084	Format = "%u"
00401172	. 68 84324000	PUSH Advance_.00403284	s = Advance_.00403284
00401177	. E8 4A020000	CALL <JMP.&user32.wsprintfA>	wsprintfA
0040117C	. 83C4 0C	ADD ESP,0C	
0040117F	. 33C0	XOR EAX,EAX	
00401181	. A3 00304000	MOV DWORD PTR DS:[403000],EAX	
00401186	. 892D 5C324000	MOV DWORD PTR DS:[40325C],EBP	
0040118C	. 68 64324000	PUSH Advance_.00403264	String2 = ""
00401191	. 68 84324000	PUSH Advance_.00403284	String1 = ""
00401196	. E8 25020000	CALL <JMP.&kernel32.lstrcpA>	lstrcpA

00403000 에 있는 값인 C2A776FA 이 output Buffer 인 00403284 에 들어가게 된다.

lstrcpA 함수에 의해 사용자가 입력한 serial 값과 비교하는 값 확인

0040118C	. 68 64324000	PUSH Advance_.00403264	String2 = "12345"
00401191	. 68 84324000	PUSH Advance_.00403284	String1 = "3265754874"
00401196	. E8 25020000	CALL <JMP.&kernel32.lstrcpA>	lstrcpA

3265754874 를 serial 로 입력한 후 결과를 보면 올바른 Serial 이 입력됨을 확인 가능하다.

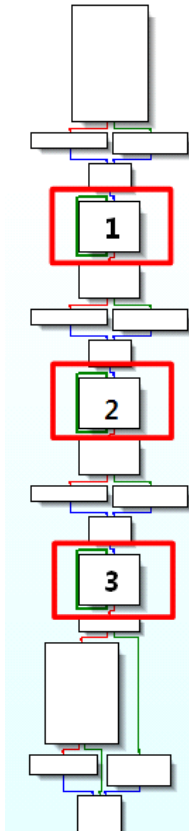


## 문제 해결

Address	Hex dump	Disassembly	Registers (FPU)
0040114E	. E8 24010000	CALL Advance_.00401277	EAX 00403238 ASCII "CodeEngn"
00401153	. 6A 20	PUSH 20	ECX 770121CC user32.770121CC
00401155	. 68 64324000	PUSH Advance_.00403264	EDX 7C93E514 ntdll.KiFastSystemCallRet
0040115A	. 68 ED030000	PUSH 3ED	EBX 00000000
0040115F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	ESP 0012FC30
00401162	. E8 77020000	CALL <JMP.&user32.GetDlgItemTextA>	EBP 0012FC3C
00401167	. FF35 00304000	PUSH DWORD PTR DS:[403000]	ESI 0040102C Advance_.0040102C
0040116D	. 68 84304000	PUSH Advance_.00403084	EDI 0012FCA4

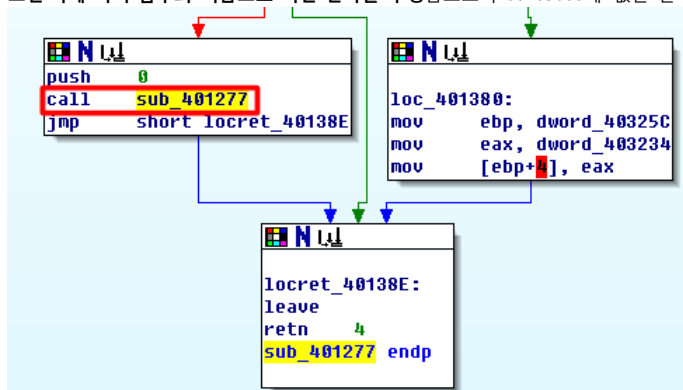
2번째 GetDlgItemTextA 함수를 호출하기 전에 Call 하는 부분을 보면 사용자가 입력한 값을 인자로 받아들인다.

00401277 Code 확인 ( With IDA Pro )



3개의 Loop 문을 가지고 있다.

- 조건 하에 다시 함수의 처음으로 가는 반복을 수행함으로써 00403000에 값을 쓴다. ( Code 가 너무 난독하므로 생략 )

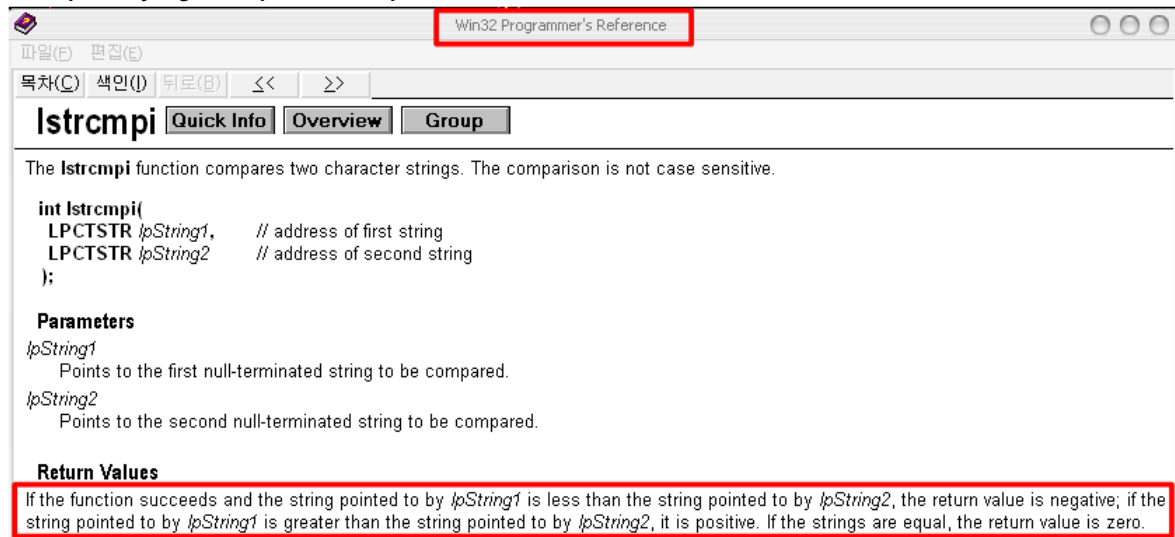


OllyDbg 로 확인해 보면 함수 수행후 00403000 의 값이 바뀌었고, Little Endian 방식으로 값을 읽어온다.

0040114E	. E8 24010000	CALL Advance_.00401277	Count = 20 (32.) Buffer = Advance_.00403264 ControlID = 3ED (1005.) hWnd GetDlgItemTextA <%u> = C2A776FA (3265754874.)
00401153	. 6A 20	PUSH 20	
00401155	. 68 64324000	PUSH Advance_.00403264	
0040115A	. 68 ED030000	PUSH 3ED	
0040115F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00401162	. E8 77020000	CALL <JMP.&user32.GetDlgItemTextA>	
00401167	. FF35 00304000	PUSH DWORD PTR DS:[403000]	
0040116D	. 68 84304000	PUSH Advance_.00403084	
Address	Hex dump	ASCII	
00403000	FA 76 A7 C2 3B 3D 38 08 78 38 37 8E 35 38 08 78	?ms;=8x87?8x	

## 보충 설명

IstrcmpA ( Ollydbg 에 Help 에 win32.hlp 를 등록하여 쉽게 확인 )

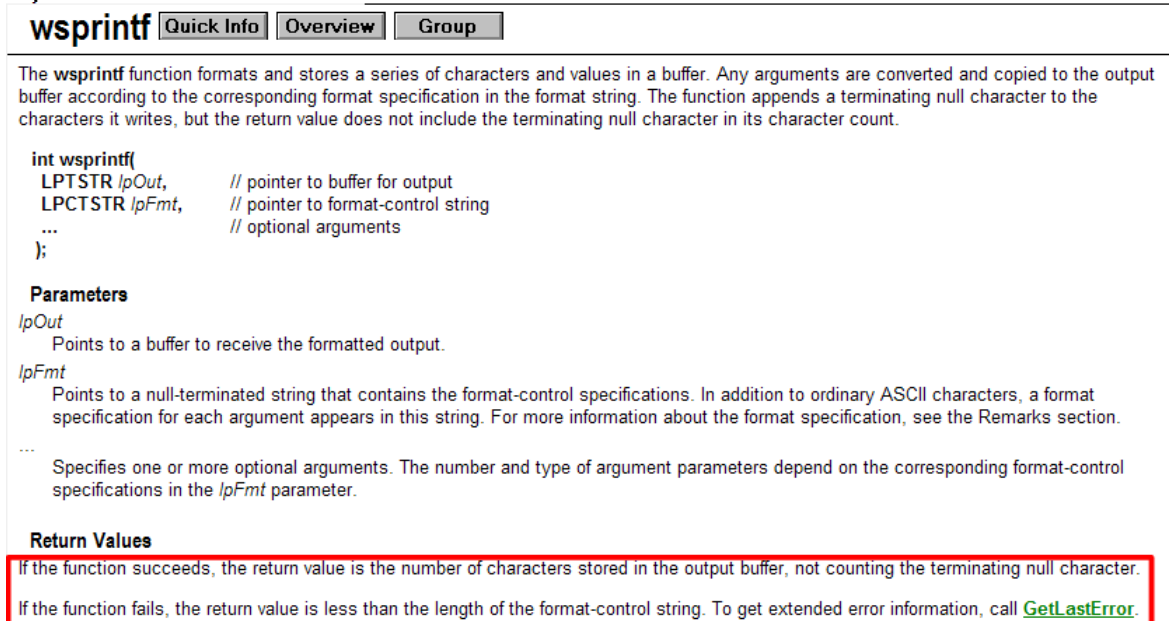


The screenshot shows the Win32 Programmer's Reference help window. The title bar says "Win32 Programmer's Reference". The menu bar includes "파일(F)", "편집(E)", "목차(C)", "색인(I)", "뒤로(B)", and navigation arrows. The "Istrcmpi" function is selected, with tabs for "Quick Info", "Overview", and "Group". The "Quick Info" tab is active, showing the function signature: `int Istrcmpi(LPCTSTR lpString1, LPCTSTR lpString2);` with comments for each parameter. Below the signature are sections for "Parameters" and "Return Values". The "Return Values" section is highlighted with a red box and contains the text: "If the function succeeds and the string pointed to by *lpString1* is less than the string pointed to by *lpString2*, the return value is negative; if the string pointed to by *lpString1* is greater than the string pointed to by *lpString2*, it is positive. If the strings are equal, the return value is zero."

String1 < String2 : -1, String1 = String2 : 0, String1 > String2 : 1 을 Return 한다.

- 함수의 Return 값은 EAX 레지스터에 반환 되므로, 위의 프로그램에서 `cmp EAX, 0` 을 하여 `JNZ` 를 이용하여 실패 성공 구문 분기를 시켰다.

## wsprintf



The screenshot shows the Win32 Programmer's Reference help window for the `wsprintf` function. The title bar says "Win32 Programmer's Reference". The menu bar includes "파일(F)", "편집(E)", "목차(C)", "색인(I)", "뒤로(B)", and navigation arrows. The "wsprintf" function is selected, with tabs for "Quick Info", "Overview", and "Group". The "Quick Info" tab is active, showing the function signature: `int wsprintf(LPTSTR lpOut, LPCTSTR lpFmt, ...);` with comments for each parameter. Below the signature are sections for "Parameters" and "Return Values". The "Return Values" section is highlighted with a red box and contains the text: "If the function succeeds, the return value is the number of characters stored in the output buffer, not counting the terminating null character. If the function fails, the return value is less than the length of the format-control string. To get extended error information, call [GetLastError](#)."

매개 변수 순서에 따라서 위의 코드에서는 `s = Out, %u = Fmt, <%u> = arguments` 가 된다.

## 답

3265754874