

- Ezbeat -

1,2번 문제에 비해 간단하네요.

이런 문제는 보통 두 개의 문자열을 비교함으로써 실패인지 성공인지를 검사하는데요.

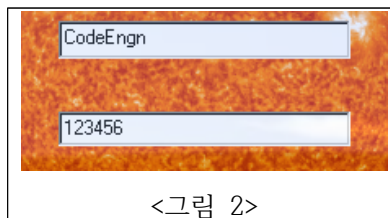
예상을 빗나가지 않고 너무 정직하게 함수도 나와 있더군요 ^^

프로그램에서 어떠한 함수를 사용하는지 봐보겠습니다.

00401002	CALL <JMP,&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
00401020	CALL <JMP,&user32.DialogBoxParamA>	user32.DialogBoxParamA
00401027	CALL <JMP,&kernel32.ExitProcess>	kernel32.ExitProcess
00401066	CALL <JMP,&winmm.PlaySoundA>	winmm.PlaySoundA
00401076	CALL <JMP,&user32.LoadIconA>	user32.LoadIconA
00401086	CALL <JMP,&user32.SendMessageA>	user32.SendMessageA
00401099	CALL <JMP,&kernel32.VirtualProtect>	kernel32.VirtualProtect
004010A6	CALL <JMP,&user32.GetDlgItem>	user32.GetDlgItem
004010AC	CALL <JMP,&user32.SetFocus>	user32.SetFocus
004010B6	CALL <JMP,&user32.GetWindowLongA>	user32.GetWindowLongA
004010C6	CALL <JMP,&user32.SetWindowLongA>	user32.SetWindowLongA
004010D7	CALL <JMP,&user32.SetLayeredWindowAttri>	user32.SetLayeredWindowAttributes
004010EC	CALL <JMP,&user32.EndDialog>	user32.EndDialog
00401120	CALL <JMP,&user32.GetDlgItemTextA>	user32.GetDlgItemTextA
00401138	CALL <JMP,&user32.MessageBoxA>	user32.MessageBoxA
0040113D	JMP Reverse2,004011D1	(Initial CPU selection)
00401162	CALL <JMP,&user32.GetDlgItemTextA>	user32.GetDlgItemTextA
00401177	CALL <JMP,&user32.wsprintfA>	user32.wsprintfA
00401196	CALL <JMP,&kernel32.lstrcmpA>	kernel32.lstrcmpA
004011AC	CALL <JMP,&user32.MessageBoxA>	user32.MessageBoxA
004011C8	CALL <JMP,&user32.MessageBoxA>	user32.MessageBoxA
004013A6	CALL <JMP,&user32.MessageBoxA>	user32.MessageBoxA

<그림 2>

너무 친절히도 비교하는 함수가 나와있네요. 해당 부분에 브표를 걸고 트레이스를 해보겠습니다. 먼저 이름은 CodeEngn으로 하겠습니다.



<그림 2>

위와 같이 입력하고 체크를 해주면

0040118C	68 64324000	PUSH Reverse2,00403264	String2 = "123456"
00401191	68 84324000	PUSH Reverse2,00403284	String1 = "3265754874"
00401196	E8 25020000	CALL <JMP,&kernel32.lstrcmpA>	lstrcmpA

<그림 3>

두 개의 스트링을 비교하고 있는 것을 볼 수 있습니다.

이제 아래를 자세히 봐보겠습니다.

lstrcmp는 같으면 0을 리턴하고 다르면 1을 리턴합니다.

아래 코드를 보겠습니다.

00401196	E8 25020000	CALL <JMP,&kernel32.lstrcmpA>	lstrcmpA
0040119B	99	CDQ	
0040119C	F7F8	IDIV EAX	
0040119E	6A 10	PUSH 10	Style = MB_OK MB_ICONHAND MB_APPLMODAL
004011A0	68 16314000	PUSH Reverse2,00403116	Title = "You failed..."
004011A5	68 F1304000	PUSH Reverse2,004030F1	Text = "No, that is not the right answer :)"
004011AA	6A 00	PUSH 0	hOwner = NULL
004011AC	E8 3F020000	CALL <JMP,&user32.MessageBoxA>	MessageBoxA

<그림 4>

비교를 한 다음 나누기를 하기 위해 CDQ로 더블을 쿼드로...(EDX:EAX 확장)해주고 EAX에

있는 값을 EAX로 나눈다. 몫은 EAX로 나머지는 EDX로 들어간다.

EAX가 1이라면 아무런 방해 없이 아래로 쭉 내려가서 실패 했다는 메시지 박스가 뜰 것입니다. 하지만 EAX가 0이라면 0을 나눈다는 것은 말도 안 되므로 예외처리가 발생할 것입니다. 예외처리가 발생했을 경우 어떠한 루틴으로 가는지 봐보면

00401392	\$	8B2D 5C324001	MOV EBP,DWORD PTR DS:[40325C]	Structured exception handler
00401398	.	6A 30	PUSH 30	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL
0040139A	.	68 E0304000	PUSH Reverse2,004030E0	Title = "You succeeded,..."
0040139F	.	68 87304000	PUSH Reverse2,00403087	Text = "Yes, you see the Good Boy Message :)"
004013A4	.	6A 00	PUSH 0	hOwner = NULL
004013A6	.	E8 45000000	CALL <JMP,&user32,MessageBoxA>	MessageBoxA

<그림 5>

성공 메시지를 출력하는 루틴으로 가는 것을 볼 수 있다.

답은 : 3265754874

추가 내용 : 글자 수는 3글자 넘게 써야되네요 ^^