

Basic RCE L04

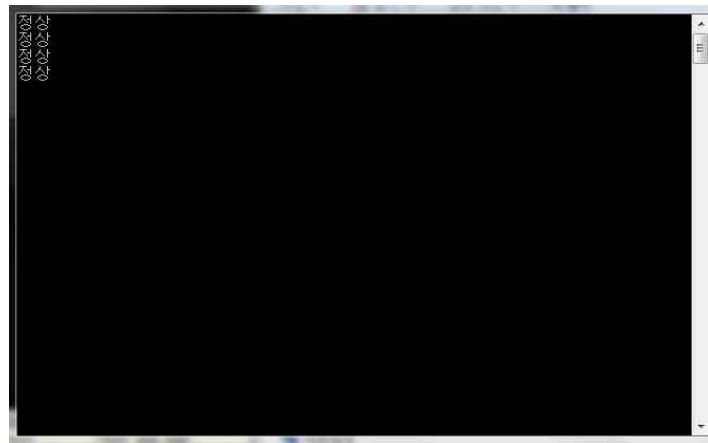
Korea :

이 프로그램은 디버거 프로그램을 탐지하는 기능을 갖고 있다. 디버거를 탐지하는 함수의 이름은 무엇인가

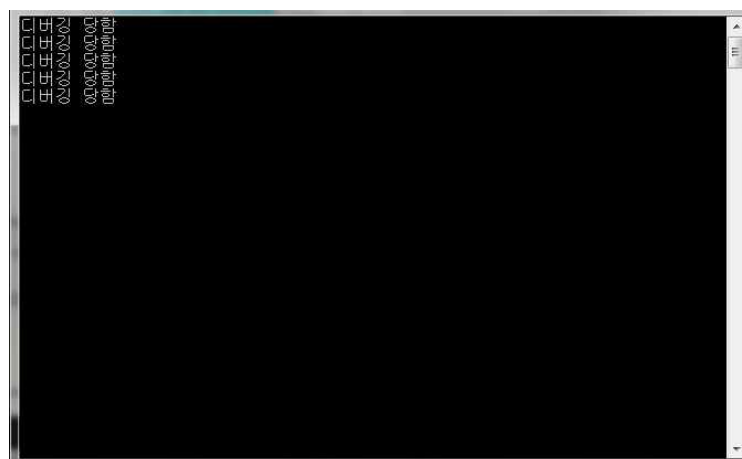
English :

This program can detect debuggers. Find out the name of the debugger detecting function the program uses.

프로그램을 실행시키자.



실행시키니 콘솔창에 정상이라는 메시지가 계속 출력된다. 문제가 디버거 탐지 함수를 구하라는 것이다. 그럼 한번 올리디버거로 디버거를 해보자. F9를 눌러 실행시키기전에는 그저 아무것도 출력하고 있지 않다가 F9를 눌러서 실행 해 보니 디버깅 당함 이라는 메시지가 계속 뜬다.



올리디버거로 열었더니 정상 이라는 메시지에서 디버깅 당함 이라는 메시지로 바뀌었다. 답은 아마도 이렇게 바뀌주는 함수를 찾는 것인가 보다.

일단 오른쪽키를 눌러 [Search for] - [All referenced text strings]를 눌러 문자열들을 봤다. 하지만 딱히 눈에 보이는건 없었다. API함수들까지 보려고 했지만 딱히 생각나는 함수가 없어서 그냥 실행하면서 나가기로 했다.

일단 Ctrl + F2를 눌러서 다시 처음화면으로 가보자.

00408370	55	PUSH EBP	
00408371	8BEC	MOV EBP,ESP	
00408373	6A FF	PUSH -1	
00408375	68 D0134300	PUSH Reverse_.004313D0	
0040837A	68 90F44000	PUSH Reverse_.0040F490	SE handler installation
0040837F	64:A1 000000	MOV EAX,DWORD PTR FS:[0]	
00408385	50	PUSH EAX	
00408386	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
0040838D	83C4 F0	ADD ESP,-10	
00408390	53	PUSH EBX	
00408391	56	PUSH ESI	
00408392	57	PUSH EDI	
00408393	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00408396	FF15 74B14300	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
0040839C	A3 2C894300	MOV DWORD PTR DS:[43892C],EAX	
004083A1	A1 2C894300	MOV EAX,DWORD PTR DS:[43892C]	
004083A6	C1E8 08	SHR EAX,8	
004083A9	25 FF000000	AND EAX,0FF	
004083AE	A3 38894300	MOV DWORD PTR DS:[438938],EAX	
004083B3	8B0D 2C894300	MOV ECX,DWORD PTR DS:[43892C]	
004083B9	81E1 FF000000	AND ECX,0FF	
004083BF	890D 34894300	MOV DWORD PTR DS:[438934],ECX	
004083C5	8B15 34894300	MOV EDX,DWORD PTR DS:[438934]	
004083CB	C1E2 08	SHL EDX,8	
004083CE	0315 38894300	ADD EDX,DWORD PTR DS:[438938]	
004083D4	8915 30894300	MOV DWORD PTR DS:[438930],EDX	
004083DA	A1 2C894300	MOV EAX,DWORD PTR DS:[43892C]	
004083DF	C1E8 10	SHR EAX,10	
004083E2	25 FFFF0000	AND EAX,0FFFF	
004083E7	A3 2C894300	MOV DWORD PTR DS:[43892C],EAX	
004083EC	6A 00	PUSH 0	Arg1 = 00000000
004083EE	E8 1D6F0000	CALL Reverse_.0040F310	Reverse_.0040F310
004083F3	83C4 04	ADD ESP,4	
004083F6	85C0	TEST EAX,EAX	

처음 코드는 이곳이다. 뭐 보이는데 별로 없다. 그래서 F8로 한단계씩 실행을 해 봐야겠다. F8로 한단계씩 실행하거나 꼭 눌러서 실행하다 보면 어느순간 멈추고 콘솔창에는 “디버깅 당함“이라는 메시지가 출력될 때가 있을 것이다. 바로 00408454 주소의 CALL Reverse_.0040100F 코드이다. 이곳에 BP를 걸고 다시 Ctrl + F2를 눌러서 처음으로 간후 F9를 눌러보자.

004083F8	75 0A	JNZ SHORT Reverse_.00408404	
004083FA	6A 1C	PUSH 1C	
004083FC	E8 CF000000	CALL Reverse_.004084D0	
00408401	83C4 04	ADD ESP,4	
00408404	C745 FC 0000	MOV DWORD PTR SS:[EBP-4],0	
0040840B	E8 00690000	CALL Reverse_.0040ED10	
00408410	FF15 70B14300	CALL DWORD PTR DS:[<&KERNEL32.GetCommand	GetCommandLineA
00408416	A3 E4A24300	MOV DWORD PTR DS:[43A2E4],EAX	
0040841B	E8 D0660000	CALL Reverse_.0040EAF0	
00408420	A3 FC884300	MOV DWORD PTR DS:[4388FC],EAX	
00408425	E8 B6610000	CALL Reverse_.0040E5E0	
0040842A	E8 61600000	CALL Reverse_.0040E490	
0040842F	E8 7C430000	CALL Reverse_.0040C7B0	
00408434	8B0D 48894300	MOV ECX,DWORD PTR DS:[438948]	
0040843A	890D 4C894300	MOV DWORD PTR DS:[43894C],ECX	
00408440	8B15 48894300	MOV EDX,DWORD PTR DS:[438948]	
00408446	52	PUSH EDX	
00408447	A1 40894300	MOV EAX,DWORD PTR DS:[438940]	
0040844C	50	PUSH EAX	
0040844D	8B0D 3C894300	MOV ECX,DWORD PTR DS:[43893C]	
00408453	51	PUSH ECX	
00408454	E8 B68BFFFF	CALL Reverse_.0040100F	
00408459	83C4 0C	ADD ESP,0C	

그리고 F7을 눌러서 내부로 들어가보자. 들어가면 **JMP Reverse_.00401030** 명령어가 보이는데, 점프를 하고나면 이런 코드들이 보일 것이다.

00401030	> 455	PUSH EBP	
00401031	8BEC	MOV EBP,ESP	
00401033	83EC 40	SUB ESP,40	
00401036	53	PUSH EBX	
00401037	56	PUSH ESI	
00401038	57	PUSH EDI	
00401039	8D7D C0	LEA EDI,DWORD PTR SS:[EBP-40]	
0040103C	B9 10000000	MOV ECX,10	
00401041	B8 CCCCCCCC	MOV EAX,CCCCCCCC	
00401046	F3:AB	REP STOS DWORD PTR ES:[EDI]	
00401048	> 8BF4	MOV ESI,ESP	
0040104A	68 E8030000	PUSH 3E8	[Timeout = 1000. ms
0040104F	FF15 68B1430	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	[Sleep
00401055	3BF4	CMP ESI,ESP	
00401057	E8 B4710000	CALL Reverse_.00408210	
0040105C	8BF4	MOV ESI,ESP	
0040105E	FF15 64B1430	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	[IsDebuggerPresent
00401064	3BF4	CMP ESI,ESP	
00401066	E8 A5710000	CALL Reverse_.00408210	
0040106B	85C0	TEST EAX,EAX	
0040106D	74 0F	JE SHORT Reverse_.0040107E	
0040106F	68 24104300	PUSH Reverse_.00431024	[Arg1 = 00431024
00401074	E8 17710000	CALL Reverse_.00408190	[Reverse_.00408190
00401079	83C4 04	ADD ESP,4	
0040107C	EB 0D	JMP SHORT Reverse_.0040108B	
0040107E	> 68 1C104300	PUSH Reverse_.0043101C	[Arg1 = 0043101C
00401083	E8 08710000	CALL Reverse_.00408190	[Reverse_.00408190
00401088	83C4 04	ADD ESP,4	
0040108B	> EB BB	JMP SHORT Reverse_.00401048	

대충 해석을 하자면 Sleep 함수로 1초씩(1000ms, ms = 밀리세컨드 = 1/1000초) 기다리고 IsDebuggerPresent API함수를 호출 한다. 이 함수가 무엇인지 알아보기 위해 마우스 오른쪽키를 눌러 [Help on symbolic name]을 눌러보자. 내 경우엔 win32.hlp 파일에 IsDebuggerPresent 함수에 대한 설명이 없었는지 뜨질 않았다. 그럼 구글링을 해보자. MSDN에 설명되어 있는데, 함수 원형은 BOOL WINAPI IsDebuggerPresent(void); 이고, 인수값은 받질 않으며,

Return Value

If the current process is running in the context of a debugger, the return value is nonzero.

If the current process is not running in the context of a debugger, the return value is zero.

이게 리턴값이다. 해석을 하자면

현재 프로그램이 디버거로 실행 되고 있으면은 0이 아닌값을 리턴한다.

현재 프로그램이 디버거로 실행 되고 있지 않으면은 0을 리턴한다.

해석이 틀렸을 수도 있다. 영어를 잘 못하기에.. 하지만 대충 이런뜻인건 분명한 것 같다.

우리가 찾는 디버깅 탐지 함수가 바로 이 함수인 것 같다. 그걸 말해주듯이 IsDebuggerPresent 함수를 실행시키면은 EAX에 1이 반환되는걸 볼 수 있다. 즉 디버거로 실행 됨을 의미한다.

Continue 칸에 IsDebuggerPresent 를 입력해보자. 5번 문제가 나올 것이다.

2011.08.16

Made by hypen1117

hypen1117@daum.net

<http://hypen1117.tistory.com>

리버싱을 그렇게 잘하지 못해 구체적이지 못하고 잘못 될 수 있습니다.
잘못되거나 부족한 부분 지적해 주시길 바랍니다.