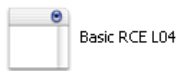


Reverse L04

2009년 12월 23일 수요일

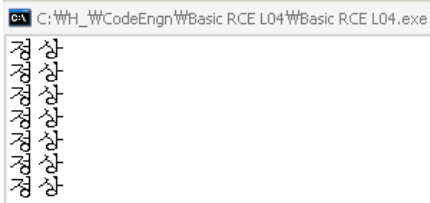
오전 9:35

파일 확인

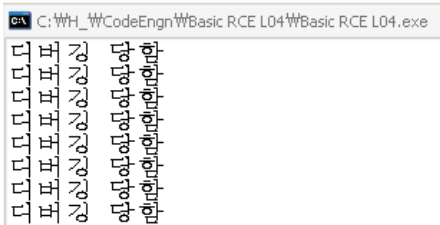


프로그램 실행

Debugger를 물리지 않고 프로그램을 실행 시켰을 경우



Debugger를 물리고 프로그램을 실행 시켰을 경우



이로 인해 이 프로그램에는 **Debugger**를 탐지 하는 함수가 있을 것이라고 판단, IDA 를 통해 좀더 상세하게 알아 보았다.

With IDA Pro

```

_main_0 proc near
var_40= byte ptr -40h

push    ebp
mov     ebp, esp
sub     esp, 40h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_40]
mov     ecx, 10h
mov     eax, 0CCCCCCCCh
rep stosd

```

```

loc 401048:
mov     esi, esp
push    3E8h           ; dwMilliseconds
call    ds:Sleep
cmp     esi, esp
call    __chkesp
mov     esi, esp
call    ds:IsDebuggerPresent
cmp     esi, esp
call    __chkesp
test    eax, eax
jz      short loc_40107E

```

```

push    offset aIC      ; "디버깅 당함 %n"
call    _printf
add     esp, 4
jmp     short loc_40108B

```

```

loc_40107E:
push    offset aD       ; "정상 %n"
call    _printf
add     esp, 4

```

```

loc 40108B:
jmp     short loc_401048
_main_0 endp

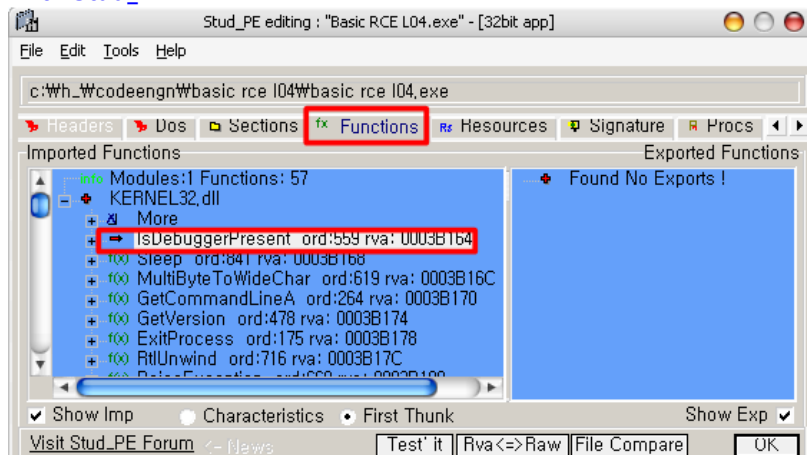
```

main 함수로 이동하여 Graph 모드로 확인한 결과, Sleep 함수와 IsDebuggerPresent 함수를 호출하고 __chkesp 를 수행하는 것을 확인하였다.

- check 한 값에 따라 "디버깅 당함" 과 "정상" 출력 구문이 달라지게 되고, 출력후 다시 00401048 로 돌아가 검사를 재수행한다.
- Imports 함수 목록에서도 확인 가능하다.

Address	Ordinal	Name	Library
0043B164		IsDebuggerPresent	KERNEL32
0043B168		Sleep	KERNEL32
0043B1...		MultiByteToWideChar	KERNEL32

With Stud_PE



Stud_PE 라는 PE 파일 검사 툴로도 IsDebuggerPresent 함수를 확인 할 수 있었다.

- Function 에서 Kernel32.dll 에 포함된 함수이며, Import 된 함수이다.

With Ollydbg

0040104A	. 68 E8030000	PUSH 3E8	Timeout = 1000. ms
0040104F	. FF15 68B14300	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	Sleep
00401055	. 3BF4	CMP ESI,ESP	
00401057	. E8 B4710000	CALL Basic_RC.00408210	
0040105C	. 8BF4	MOV ESI,ESP	
0040105E	. FF15 64B14300	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	IsDebuggerPresent
00401064	. 3BF4	CMP ESI,ESP	
00401066	. E8 A5710000	CALL Basic_RC.00408210	
0040106B	. 85C0	TEST EAX,EAX	
0040106D	. 74 0F	JE SHORT Basic_RC.0040107E	
0040106F	. 68 24104300	PUSH Basic_RC.00431024	Arg1 = 00431024
00401074	. E8 17710000	CALL Basic_RC.00408190	Basic_RC.00408190
00401079	. 83C4 04	ADD ESP,4	
0040107C	. EB 0D	JMP SHORT Basic_RC.0040108B	
0040107E	> 68 1C104300	PUSH Basic_RC.0043101C	Arg1 = 0043101C
00401083	. E8 08710000	CALL Basic_RC.00408190	Basic_RC.00408190
00401088	. 83C4 04	ADD ESP,4	
0040108B	> EB BB	JMP SHORT Basic_RC.00401048	
0040108D	. CC	INT3	
0040108E	. 00	JMP	
00431024=Basic_RC.00431024 (ASCII "디버깅 당함 ■")			

마우스 우클릭 -> Search for -> All referenced text strings 에서 "디버깅 당함" 을 찾아 들어간 결과에도 위에 IsDebuggerPresent 함수를 발견

보충 설명

IsDebuggerPresent

TIB 블록 (FS : [18h]) 에서 PEB 구조체 주소값을 구한 뒤, PEB의 BeingDebugged 필드를 반환하는 함수이다.

7C7E3133	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
7C7E3139	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
7C7E313C	0FB640 02	MOVZX EAX,BYTE PTR DS:[EAX+2]

Windbg 로 확인

```
> dt teb
0:000> .symlfx
0:000> .sympath
Symbol search path is: srv*
Expanded Symbol search path is: cache*.;SRV*http://msdl.microsoft.com/download/symbols
0:000> .reload
Reloading current modules
...
0:000> dt teb
ntdll!_TEB
+0x000 NtTib : _NT_TIB
+0x01c EnvironmentPointer : Ptr32 Void
+0x020 ClientId : _CLIENT_ID
+0x028 ActiveRpcHandle : Ptr32 Void
+0x02c ThreadLocalStoragePointer : Ptr32 Void
+0x030 ProcessEnvironmentBlock : Ptr32 _PEB
+0x034 LastErrorValue : Uint4B
+0x038 CountOfOwnedCriticalSections : Uint4B
+0x03c CsrClientThread : Ptr32 Void
```

0x030은 PEB 구조체의 시작 주소이다.

```
> dt _peb
0:000> dt _peb
ntdll!_PEB
+0x000 InheritedAddressSpace : UChar
+0x001 ReadImageFileExecOptions : UChar
+0x002 BeingDebugged : UChar
+0x003 SpareBool : UChar
+0x004 Mutant : Ptr32 Void
+0x008 ImageBaseAddress : Ptr32 Void
+0x00c Ldr : Ptr32 _PEB_LDR_DATA
+0x010 ProcessParameters : Ptr32 _RTL_USER_PROCESS_PARAMETERS
+0x014 SubSystemData : Ptr32 Void
```

BeingDebugged라는 멤버 변수의 값을 읽어서 1이면 debugger 탐지, 0이면 실행을 수행한다.

선언 예)

선언 : public declare functon InDebuggerPresent Lib "kernel32.dll" () As Long

사용 : If IsDebuggerPresent Then

MsgBox "Debugger Found ", vbCritical, "Program will be exited"

End

Endif

문제 해결

Ollydbg Plugin 을 이용하거나, 직접 Being Debugged 변수를 0으로 바꿔 준다.

7C7E3133	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
7C7E3139	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
7C7E313C	0FB640 02	MOVZX EAX,BYTE PTR DS:[EAX+2]
7C7E3140	C3	RETN
7C7E3141	90	NOP
7C7E3142	90	NOP
7C7E3143	90	NOP
7C7E3144	90	NOP
7C7E3145	90	NOP
7C7E3146	8BFF	MOV EDI,EDI
7C7E3148	55	PUSH EBP
7C7E3149	8BEC	MOV EBP,ESP
7C7E314B	57	PUSH ESI
DS:[7FFDA002]=00		
EAX=7FFDA000		
Address	Hex dump	ASCII
7FFDA002	00 FF FF FF FF 00 00 40 00 A0 1E 24 00 00 00	.jjjjj..

IsDebuggerPresent 함수에 Breakpoint 를 설정한 후, Debugging 을 하다보면, DS=7FFDA002 가 1로 설정되어있다.

- Ctrl+G 를 Hex Dump 창에서 입력한 후, 값을 0 으로 바꿔 주었다.

7C7E3133	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
7C7E3139	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
7C7E313C	0FB640 02	MOVZX EAX,BYTE PTR DS:[EAX+2]
7C7E3140	C3	RETN

C:\WH_₩₩CodeEngn₩₩Basic RCE L04₩₩Basic RCE L04.exe

```

정상
정상
정상
정상
정상
정상

```

- Debugger 가 켜져있음에도 불구하고, 정상으로 출력이 된다.

답

IsDebuggerPresent