
[CodeEngn]

Basic RCE L04

OllyDbg - 04.exe - [Found intermodular calls]		
File View Debug Options Window Help		
L E M T W H C / K B R ... S		
Address	Disassembly	Destination
0040104F	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	KERNELBA.Sleep
0040105E	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	KERNELBA.IsDebuggerPresent
00407D05	CALL DWORD PTR DS:[<&KERNEL32.MultiByteToWideChar>]	KERNEL32.MultiByteToWideChar
00407DA0	CALL DWORD PTR DS:[<&KERNEL32.MultiByteToWideChar>]	KERNEL32.MultiByteToWideChar
00407DEE	CALL DWORD PTR DS:[<&KERNEL32.MultiByteToWideChar>]	KERNEL32.MultiByteToWideChar
00408370	PUSH EBP	(Initial CPU selection)
00408396	CALL DWORD PTR DS:[<&KERNEL32.GetVersion>]	KERNELBA.GetVersion
00408410	CALL DWORD PTR DS:[<&KERNEL32.GetCommandLineA>]	KERNELBA.GetCommandLineA
004084F2	CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]	KERNEL32.ExitProcess
004086E7	CALL <JMP.&KERNEL32.RtlUnwind>	KERNEL32.RtlUnwind
00408A43	CALL <JMP.&KERNEL32.RtlUnwind>	KERNEL32.RtlUnwind
00408763	CALL DWORD PTR DS:[<&KERNEL32.RaiseException>]	KERNELBA.RaiseException
00408CE2	CALL DWORD PTR DS:[<&KERNEL32.IsBadReadPtr>]	KERNEL32.IsBadReadPtr
00408CFA	CALL DWORD PTR DS:[<&KERNEL32.IsBadWritePtr>]	KERNEL32.IsBadWritePtr
00408DB3	CALL DWORD PTR DS:[<&KERNEL32.HeapValidate>]	KERNEL32.HeapValidate
00408E22	CALL DWORD PTR DS:[<&KERNEL32.HeapValidate>]	KERNEL32.HeapValidate
00408E3A	CALL DWORD PTR DS:[<&KERNEL32.HeapValidate>]	KERNEL32.HeapValidate
0040C881	CALL DWORD PTR DS:[<&KERNEL32.GetCurrentProcess>]	KERNELBA.GetCurrentProcess
0040C888	CALL DWORD PTR DS:[<&KERNEL32.TerminateProcess>]	KERNEL32.TerminateProcess
0040C93F	CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>]	KERNEL32.ExitProcess
0040C983	CALL DWORD PTR DS:[<&KERNEL32.DebugBreak>]	KERNELBA.DebugBreak
0040CA2E	CALL DWORD PTR DS:[<&KERNEL32.GetStdHandle>]	KERNEL32.GetStdHandle
0040CA48	CALL DWORD PTR DS:[<&KERNEL32.GetStdHandle>]	KERNEL32.GetStdHandle
0040CB15	CALL DWORD PTR DS:[<&KERNEL32.InterlockedIncrement>]	KERNEL32.InterlockedIncrement
0040CB31	CALL DWORD PTR DS:[<&KERNEL32.LoadLibraryA>]	KERNEL32.LoadLibraryA
0040CB52	CALL DWORD PTR DS:[<&KERNEL32.GetProcAddress>]	KERNEL32.GetProcAddress
0040CB92	CALL DWORD PTR DS:[<&KERNEL32.OutputDebugStringA>]	KERNELBA.OutputDebugStringA
0040CB9D	CALL DWORD PTR DS:[<&KERNEL32.InterlockedDecrement>]	KERNEL32.InterlockedDecrement
0040CD0B	CALL DWORD PTR DS:[<&KERNEL32.InterlockedDecrement>]	KERNEL32.InterlockedDecrement
0040CD65	CALL DWORD PTR DS:[<&KERNEL32.WriteFile>]	KERNELBA.WriteFile
0040CD83	CALL DWORD PTR DS:[<&KERNEL32.OutputDebugStringA>]	KERNELBA.OutputDebugStringA
0040CDFA	CALL DWORD PTR DS:[<&KERNEL32.InterlockedDecrement>]	KERNEL32.InterlockedDecrement
0040CE13	CALL DWORD PTR DS:[<&KERNEL32.InterlockedDecrement>]	KERNEL32.InterlockedDecrement
0040CE6C	CALL DWORD PTR DS:[<&KERNEL32.GetModuleFileNameA>]	KERNEL32.GetModuleFileNameA
0040E2A8	CALL DWORD PTR DS:[<&KERNEL32.UnhandledExceptionFilter>]	KERNEL32.UnhandledExceptionFilter
0040E600	CALL DWORD PTR DS:[<&KERNEL32.GetModuleFileNameA>]	KERNEL32.GetModuleFileNameA
0040E60D	CALL DWORD PTR DS:[<&KERNEL32.GetEnvironmentStringsW>]	KERNELBA.GetEnvironmentStringsW
Analysing 04: 634 heuristical procedures, 161 calls to known, 844 calls to guessed functions		
		Paused

? 가 디버거를 탐지하는 함수라고 한다.

== 변의편 ==

OllyDbg - 04.exe - [CPU - main thread, module 04]

File View Debug Options Window Help

Assembly window showing instructions from 00401039 to 00401093. Instruction 0040105E is highlighted: CALL DWORD PTR DS:[<KERNEL32.IsDebuggerPresent>].

Registers (FPU) window showing values: EAX=00000000, ECX=4A816734, EDI=00000000, ESI=00333000, ESP=0019FEF4, EBP=0019FF40.

Breakpoint at 0040105E

40105E 에 break 를 걸어서 테스트를 해보았다.

자동 실행 버튼을 누를때마다 디버깅당함이라는 텍스트가 나왔다.

이 IsDebuggerPresent() 는 디버거 당할때는 반환값 1, 아닐 경우에는 리턴값 0 을 반환해준다.

OllyDbg - 04.exe - [CPU - main thread, module 04]

File View Debug Options Window Help

LEMTWHC / KBR ... S

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:A8	REP STOS DWORD PTR ES:[EDI]	
00401048	> 8BF4	MOV ESI,ESP	
0040104A	. 68 E8030000	PUSH 3E8	
0040104F	. FF15 68B14300	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	Timeout = 1000. ms Sleep
00401055	. 3BF4	CMP ESI,ESP	
00401057	. E8 B4710000	CALL 04.00408210	
0040105C	. 8BF4	MOV ESI,ESP	
0040105E	. FF15 64B14300	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	IsDebuggerPresent
00401064	. 3BF4	CMP ESI,ESP	
00401066	. E8 A5710000	CALL 04.00408210	
0040106B	. 85C0	TEST EAX,EAX	
0040106D	. 74 0F	JE SHORT 04.0040107E	
0040106F	. 68 24104300	PUSH 04.00431024	Arg1 = 00431024
00401074	. E8 17710000	CALL 04.00408190	04.00408190
00401079	. 83C4 04	ADD ESP,4	
0040107C	. EB 00	JMP SHORT 04.0040108B	
0040107E	> 68 1C104300	PUSH 04.0043101C	Arg1 = 0043101C
00401083	. E8 08710000	CALL 04.00408190	04.00408190
00401088	. 83C4 04	ADD ESP,4	
0040108B	> EB BB	JMP SHORT 04.00401048	
0040108D	. CC	INT3	
0040108E	. CC	INT3	
0040108F	. CC	INT3	

Registers (MMX)

EAX 00000001
ECX 36AD04C3
EDX 00000000
EBX 00324000
ESP 0019FEF4
EBP 0019FF40
ESI 0019FEF4
EDI 0019FF40
EIP 0040106B 04.0040106B

C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 327000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE)

MM0 0000 0000 0000 0000
MM1 0000 0000 0000 0000
MM2 0000 0000 0000 0000
MM3 0000 0000 0000 0000
MM4 0000 0000 0000 0000
MM5 0000 0000 0000 0000
MM6 0000 0000 0000 0000
MM7 0000 0000 0000 0000

Address ASCII dump

0019FEF4 00408370 04.<ModuleEntryPoint>
0019FEF8 00408370 04.<ModuleEntryPoint>
0019FEFC 00324000
0019FF00 CCCCCCCC
0019FF04 CCCCCCCC
0019FF08 CCCCCCCC
0019FF0C CCCCCCCC
0019FF10 CCCCCCCC

Paused

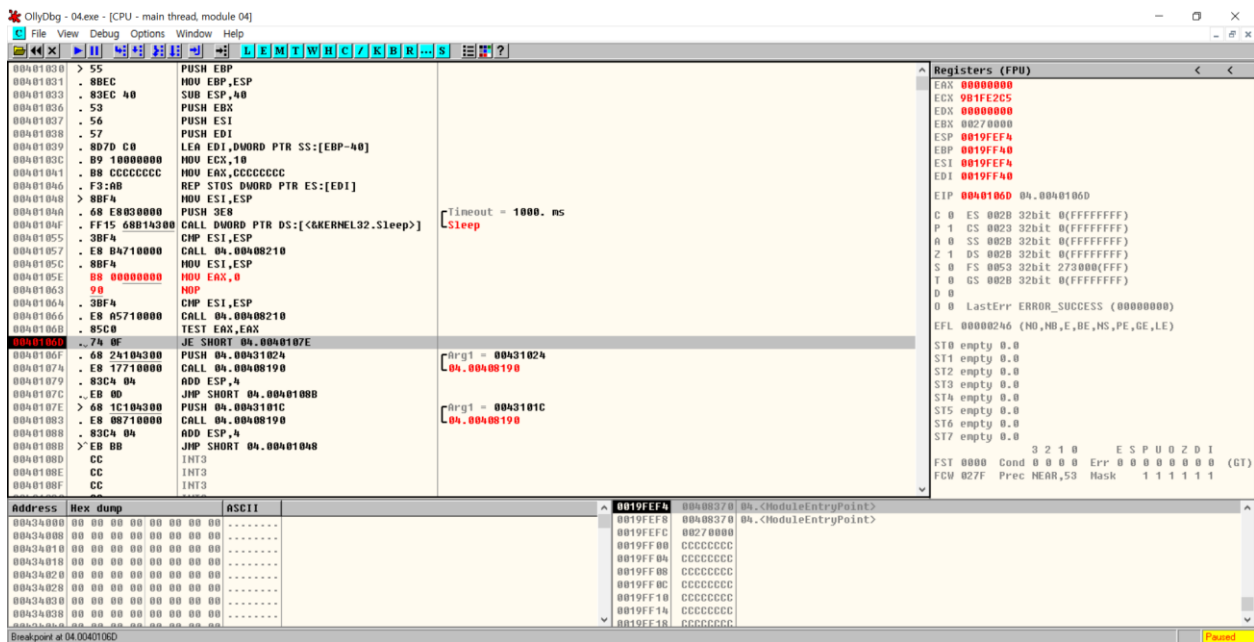
현재 디버거를 실행한 상태여서, EAX 에 1 이 들어간 것을 확인할 수 있다.

그리고 TEST EAX, EAX 가 있는데,

EAX + EAX 해서 나온 결과 값이 2 이다, 그리고 ZF 는 0 set 이 되고,

JE 조건이 맞지 않기 때문에, 40106D 로 가게 된다.

그러면, 디버깅당함이라는 문자열이 나오게 된다.

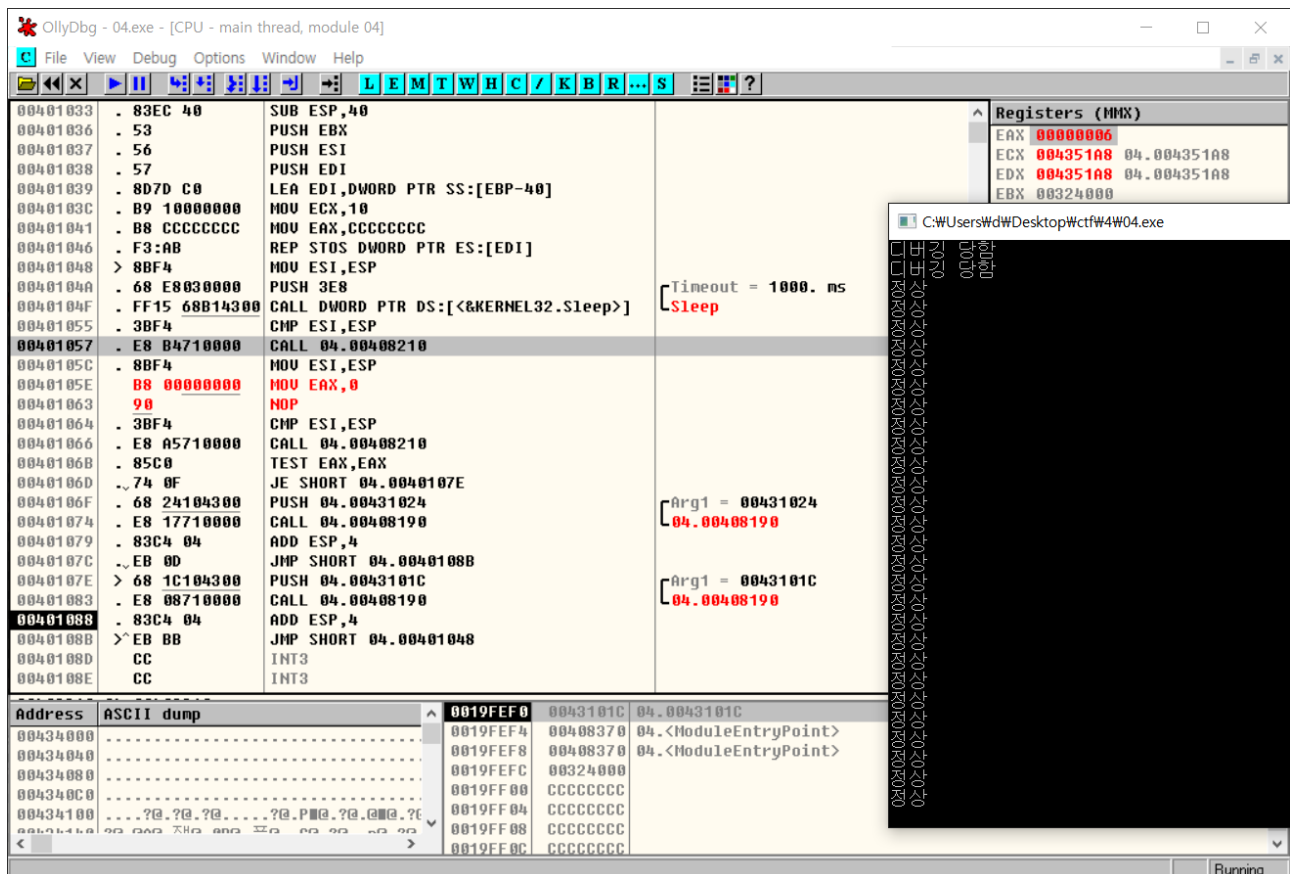


IsDebuggerPresent()에 반환값을 강제로 0 으로 설정하고 난 뒤에 결과이다.

TEST EAX, EAX 에 결과는 0 이다.

즉, ZF 가 1 set 이 되고,

때문에, 40107E 로 점프하게 되고, “정상” 이라는 문자열이 나타나게 된다.



IsDebuggerPresent() 우회 성공이다.