

Code Engn Basic 19

4.Z320

elttzero@gmail.com

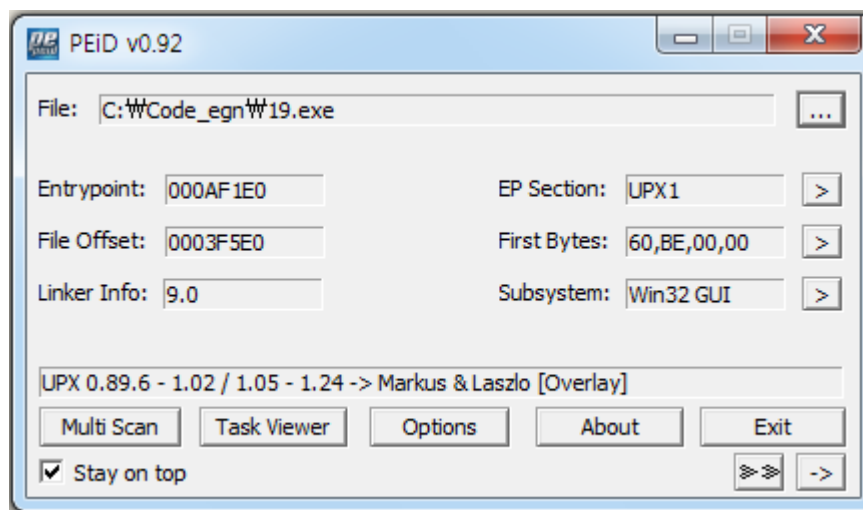
Challenges : Basic 19

Author : CodeEngn

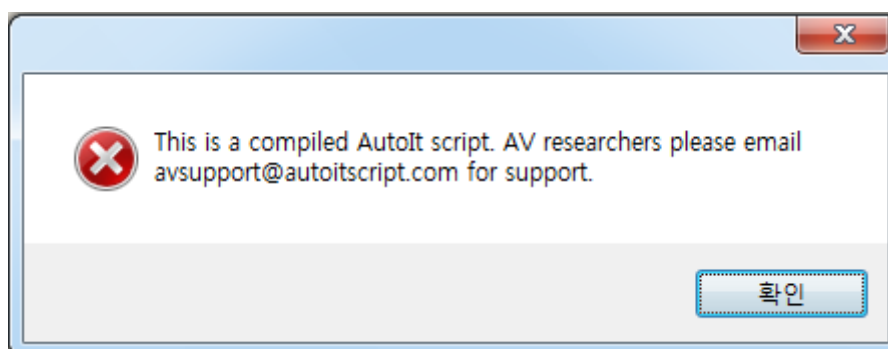
KO : 이 프로그램은 몇 밀리세컨드 후에 종료 되는가

EN : How many milliseconds does it take for this program to terminate

프로그램이 종료되는 시간을 찾아내는 문제입니다.



프로그램을 확인하면 UPX 로 패킹되어 있음을 알 수 있습니다. UPX 로 언팩해 줍시다.



OllyDbg 로 실행하다 보면 정상실행이 되지 않고 이렇게 에러가 뜨는것을 확인할 수 있습니다.

안티디버깅이 의심되므로 우선 IsDebugPresent 부터 찾아봅니다.

0040E950	. 68 04010000	PUSH 104	BufSize = 104 (260.)
0040E955	. FF15 24D3470	CALL DWORD PTR DS:[&KERNEL32.GetCurrentDirectoryW]	GetCurrentDirectoryW
0040E95B	. 57	PUSH EDI	
0040E95C	. E8 1FDFFFFF	CALL 19_0040C880	
0040E961	. FF15 20D3470	CALL DWORD PTR DS:[&KERNEL32.IsDebuggerPresent]	IsDebuggerPresent
0040E967	. 85C0	TEST EAX,EAX	
0040E969	. 0F85 6F4F020	JNZ 19_004338DE	
0040E96F	. 8B4424 0F	MOV BYTE PTR SS:[ESP+F],AL	
0040E973	. BE 30044A00	MOV ESI,19_004A0430	
0040E978	. 3905 3CF4490	CMP DWORD PTR DS:[49F43C],EAX	
0040E97F	. 0F84 734F020	JE 19_004338F7	

찾아낸 IsDebuggerPresent 는 TEST EAX EAX 를 실행 후 JNZ 를 통하여 프로그램을 종료시킵니다.

0040E95B	. 57	PUSH EDI	
0040E95C	. E8 1FDFFFFF	CALL 19_0040C880	
0040E961	. FF15 20D3470	CALL DWORD PTR DS:[&KERNEL32.IsDebuggerPresent]	IsDebuggerPresent
0040E967	. 33C0	XOR EAX,EAX	
0040E969	. 0F85 6F4F020	JNZ 19_004338DE	

이를 회피하기 위해서는 JNZ 부분을 NOP 으로 처리하여도 되지만

수정하는 부분을 최소화 시키기 위해 TEST 를 XOR 로 바꾸어 회피하도록 하겠습니다.

0044409F	. 83C4 18	ADD ESP,18	
004440A2	. 8BF0	MOV ESI,EAX	
004440A4	. 8B5424 20	MOV EDX,DWORD PTR SS:[ESP+20]	
004440A8	. 8B4424 1C	MOV EAX,DWORD PTR SS:[ESP+1C]	
004440AC	. 8B4C24 18	MOV ECX,DWORD PTR SS:[ESP+18]	
004440B0	. 52	PUSH EDX	
004440B1	. 8B5424 18	MOV EDX,DWORD PTR SS:[ESP+18]	
004440B5	. 50	PUSH EAX	
004440B6	. 51	PUSH ECX	
004440B7	. 52	PUSH EDX	
004440B8	. FF15 9CD6470	CALL DWORD PTR DS:[&USER32.MessageBoxW]	MessageBoxW
004440BE	. 8BF8	MOV EDI,EAX	
004440C0	. 85F6	TEST ESI,ESI	
004440C2	. 74 17	JE SHORT 19_004440DB	
004440C4	. 6A FF	PUSH -1	
004440C6	. 56	PUSH ESI	
004440C7	. C605 D3E8480	MOV BYTE PTR DS:[48E8D3],0	
004440CE	. FF15 D4D2470	CALL DWORD PTR DS:[&KERNEL32.WaitForSingleObject]	WaitForSingleObject
004440D4	. 56	PUSH ESI	
004440D5	. FF15 E4D2470	CALL DWORD PTR DS:[&KERNEL32.CloseHandle]	CloseHandle
004440DB	. 8B4C24 18	MOV ECX,DWORD PTR SS:[ESP+18]	

메세지 박스를 만드는 구간입니다만 딱히 시간을 가지고 카운트 하는 부분이 없습니다.

이를 찾기 위하여 함수 리스트에서 시간을 구하거나 Sleep 등을 찾아 모두 Break Point 를 걸어봅니다.

004443DC	CALL DWORD PTR DS:[&KERNEL32.TerminateThread]	kernel32.TerminateThread
0040B350	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
0040E6CA	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
004301F3	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
004305BC	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
0043197F	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00431D70	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00431EED	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00444C44	CALL EDI	WINMM.timeGetTime
00451B82	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00451B9E	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00456F68	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
0046FBC1	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
0046FB0C	CALL DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
004183AA	CALL DWORD PTR DS:[&KERNEL32.TlsAlloc]	kernel32.TlsAlloc

```

00437B9A CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0040B341 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00412571 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004140EE CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0041AE3A CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0041AE88 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0041AEDA CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004298FE CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0042EADA CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0043010B CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004305A8 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004306C7 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00430AC7 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00431967 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00431D5C CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00431E72 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00431EDA CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00437C31 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00437C64 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00437CC6 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004392FE CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00439333 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00439397 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0043A04A CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0043A063 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004460EC CALL EDI kernel32.Sleep
00446E1D CALL EDI kernel32.Sleep
0044D106 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0044D114 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0044D1FA CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00453395 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004568F0 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
004568FE CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00463EB1 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0046F42C CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0046FC65 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00472F94 CALL DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
0045D6E5 CALL <JMP.&WSOCK32.#23> WS2_32.socket
0045D7A1 CALL <JMP.&WSOCK32.#23> WS2_32.socket

```

그리고 실행을 하게 되면

```

00444C3H . 53 PUSH EBX
00444C3B . 55 PUSH EBP
00444C3C . 56 PUSH ESI
00444C3D . 57 PUSH EDI
00444C3E . 8B3D 58D7470 MOV EDI,DWORD PTR DS:[<%%WINMM.timeGetTime>] WINMM.timeGetTime
00444C44 . FFD7 CALL EDI <%%WINMM.timeGetTime>
00444C46 . 803D D3E8480 CMP BYTE PTR DS:[48E8D3],0
00444C4D . 8BF0 MOV ESI,EAX
00444C4F . 0F84 FF000000 JE 19____.00444D54
00444C55 . 8B5C24 14 MOV EBX,DWORD PTR SS:[ESP+14]
00444C59 . 8B2D 58D1470 MOV EBP,DWORD PTR DS:[<%%KERNEL32.Sleep>] kernel32.Sleep
00444C5F . FFD7 CALL EDI
00444C61 . 3BC6 CMP EAX,ESI
00444C63 . 0F83 CF000000 JNB 19____.00444D38
00444C69 . 2BC6 SUB EAX,ESI
00444C6B . 48 DEC EAX
00444C6C . E9 C9000000 JMP 19____.00444D3A
00444C71 . 8B03 MOV EAX,DWORD PTR DS:[EBX]
00444C73 . 6A 00 PUSH 0
00444C75 . 68 FC864300 PUSH 19____.004386FC
00444C7A . 50 PUSH EAX
00444C7B . C705 28F9490 MOV DWORD PTR DS:[49F9281.0]

```

이부분에서 실행이 멈추게 됩니다.

우선 TimeGetTime 함수를 통하여 시스템의 시간을 msec 단위로 구한 뒤 이를 ESI 에 보관합니다.

그리고 한번 더 실행하고 EAX 와 ESI 를 비교, EAX 가 ESI 보다 크거나 같으면 JMP 를 하게 됩니다.

```

00444D32 . 33C0 XOR EHX,EHX
00444D34 . 5B POP EBX
00444D35 . C2 0400 RETN 4
00444D38 . 2BC6 SUB EAX,ESI
00444D3A . 3B43 04 CMP EAX,DWORD PTR DS:[EBX+4]
00444D3D . 0F83 2EFFFFFF JNB 19____.00444C71
00444D43 . 6A 0A PUSH 0A
00444D45 . FFD5 CALL EBP
00444D47 . 803D D3E8480 CMP BYTE PTR DS:[48E8D3],0
00444D4E . 0F85 0BFFFFFF JNZ 19____.00444C5F
00444D54 . 5F POP EDI
Stack DS:[008AF894]=00002B70
EAX=00000000
Jump from 00444C6C

```

그리고 이곳에 도착한 뒤 EAX 에 ESI 를 뺀 뒤 EAX 와 [EBX+4]를 비교연산 합니다.

그리고 EAX 가 [EBX+4] 즉 0x2B70 과 같아질때까지 다시 위로 올라가 CALL EDI 를 통해 EAX 값을 갱신하게 되고 이런식으로 계속 반복연산을 하다가 0x2B70 이상의 값이 되면 CMP 구문 밑의 JNB 를 통하여 프로그램을 종료하는 루틴으로 넘어가게 됩니다.

따라서 이번 문제의 정답은 0x2B70 즉 11120 초가 됩니다.