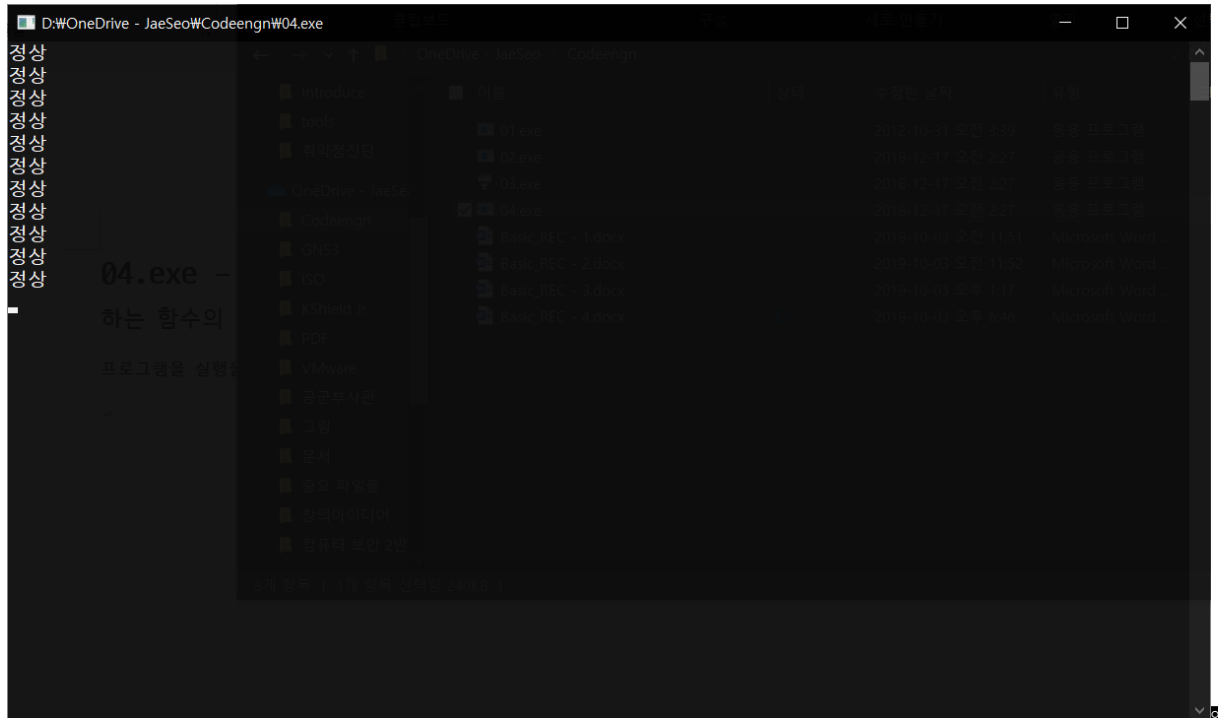


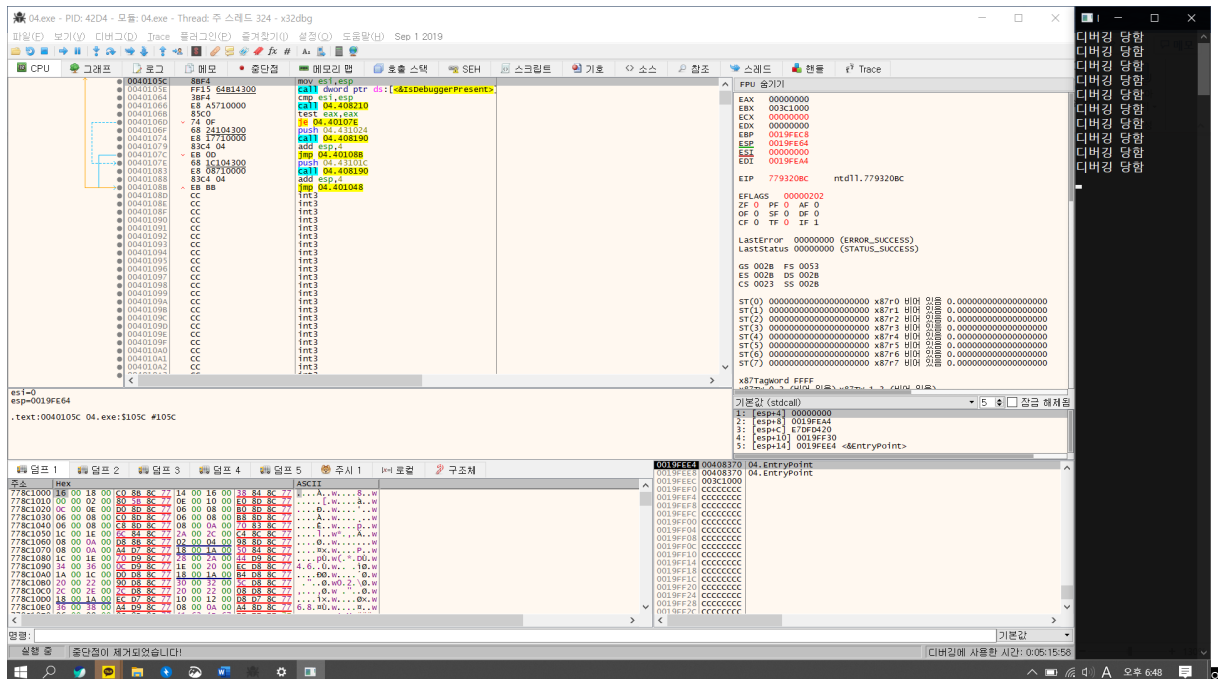
**04.exe** - 이 프로그램은 디버거 프로그램을 탐지하는 기능을 갖고 있다. 디버거를 탐지하는 함수의 이름은 무엇인가?

프로그램을 실행을 해본다.



일정 시간동안 딜레이를 가지고 정상적으로 동작 하는지에 대해 체크 하는 모습을 볼수 있다.

X32DBG에서 실행해본다.

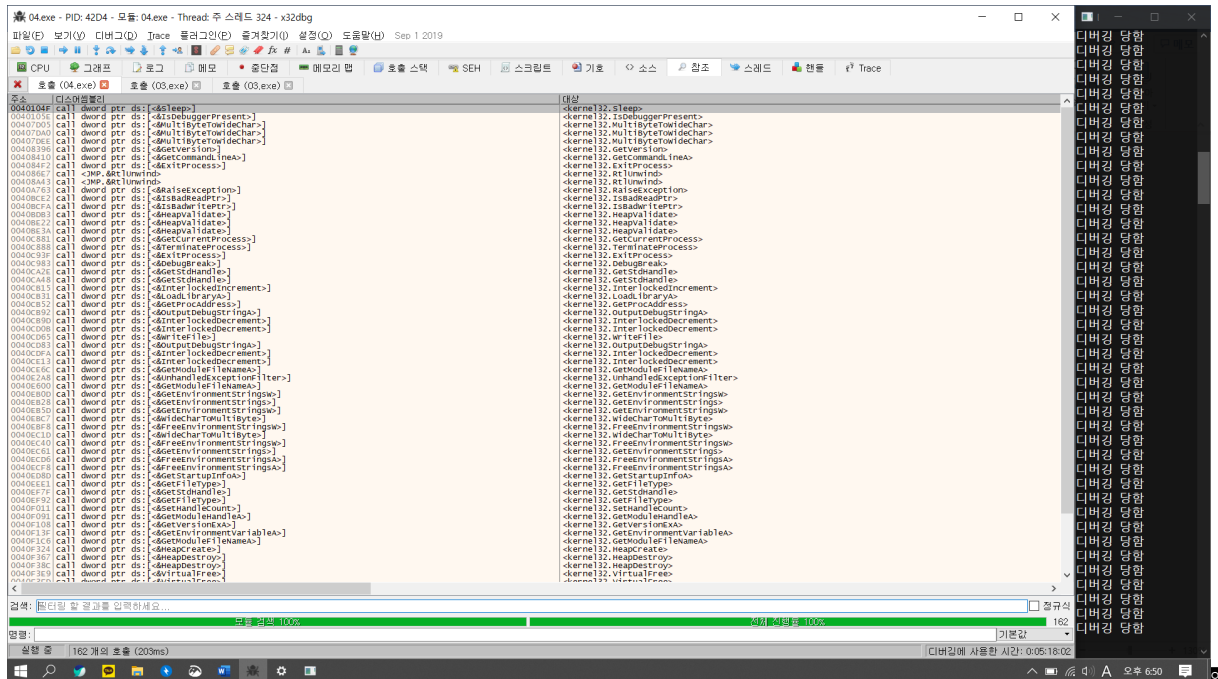


디버깅 환경에서는 “디버깅 당함”이라는 문자를 계속 출력하는 모습을 볼수 있다.

이제 프로그램을 분석 해본다.

계속 반복적으로 작동을 하면서 디버거 프로그램을 탐지하는 기능을 담당 하는 함수를 호출하는 것으로 예상된다.❶

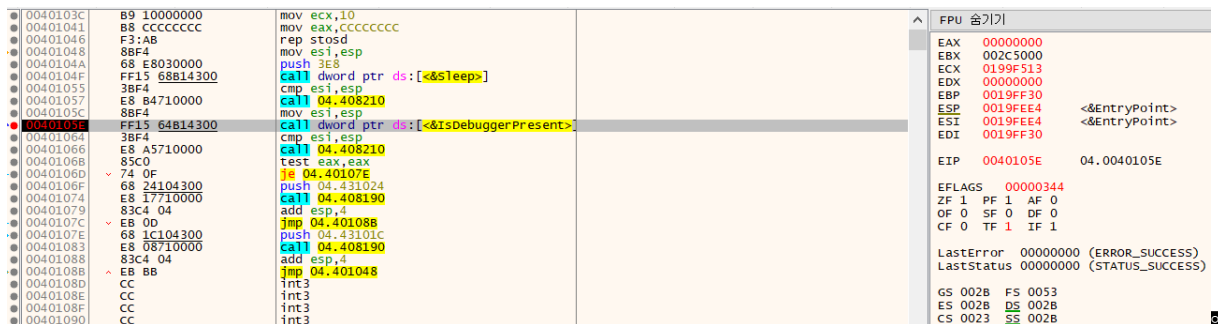
이때 인터럽트를 검색해본다.❷



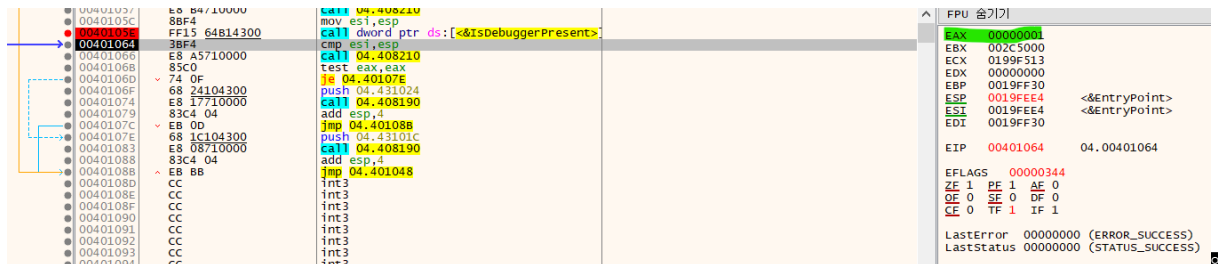
디버거를 가르키는 키워드 dbg,debug 등을 검색한다.❸

주소	디스어셈블리	대상
0040105E	call dword ptr ds:[<&IsDebuggerPresent>]	<kernel32.IsDebuggerPresent>
0040C983	call dword ptr ds:[<&DebugBreak>]	<kernel32.DebugBreak>
0040CB92	call dword ptr ds:[<&OutputDebugStringA>]	<kernel32.OutputDebugStringA>
0040CD83	call dword ptr ds:[<&OutputDebugStringA>]	<kernel32.OutputDebugStringA>

그결과 이와 같은 함수들을 발견하게 되는데 디버거를 탐지하는 함수로 의심이 되는 “IsDebuggerPresent”에 들어가 살펴 본다.❹



그리고 “IsDebuggerPresent”에서 반환하는 값을 살펴볼수 있도록 중단점을 걸어둔다. 그리고 Call를 했을때의 EAX값을 살펴본다.❺



위와 같이 EAX를 1를 반환을 받는 것을 알수 있다. 그리고 밑에 있는 어셈블리어를 해석해 본다.❻

cmp eax,esp		EBP 0019FF30
call 04.408210		ESP 0019FEE4 <&EntryPoint>
test eax,eax		ESI 0019FEE4 <&EntryPoint>
je 04.40107E		EDI 0019FF30
push 04.431024		

Esi와 esp를 cmp로 비교를 한다.

ZF 플래그는 1이 되는 것을 알수 있다.

그다음 call 04.408210를 호출 한다. 내부를 살펴본다.

→ 00401066	E8 A5710000	call 04.408210	
------------	-------------	----------------	--

00408210	75 01	jne 04.408213	
00408212	C3	ret	
00408213	55	push ebp	
00408214	88EC	mov ebp,esp	
00408216	83EC 00	sub esp,0	
00408219	50	push eax	
0040821A	52	push edx	
0040821B	53	push ebx	
0040821C	56	push esi	
0040821D	57	push edi	
0040821E	68 E8124300	push 04.4312E8	
00408223	68 C4114300	push 04.4311CC	
00408228	6A 7A	nush 7A	

EAX	00000000
EBP	0019FF30
ESP	0019FEE0 <&EntryPoint>
ESI	0019FEE4 <&EntryPoint>
EDI	0019FF30
EIP	00408210 04.00408210
EFLAGS	00000246
ZF	1 PF 1 AF 0
OF	0 SF 0 DF 0
CF	0 TF 0 IF 1

Jne 04.408213은 ZF가 1이기 때문에 작동을 안하고 return 하게 된다.

00401066	E8 A5710000	call 04.408210	
00401068	85C0	test eax,eax	
0040106B	74 0F	je 04.40107E	
0040106F	68 24104300	push 04.431024	
00401074	E8 17710000	call 04.408190	
00401079	83C4 04	add esp,4	
0040107C	EB 0D	jmp 04.40108B	
0040107E	68 1C104300	push 04.43101C	
00401083	E8 08710000	call 04.408190	
00401088	83C4 04	add esp,4	
0040108B	EB BB	jmp 04.401048	
0040108D	CC	int3	
0040108E	CC	int3	

EAX	00000001
EBX	003EB000
ECX	19D5BE32
EDX	00000000
EBP	0019FF30
ESP	0019FEE4 <&EntryPoint>
ESI	0019FEE4 <&EntryPoint>
EDI	0019FF30
EIP	0040106B 04.0040106B
EFLAGS	00000246

그다음 test eax,eax를 하게 되는데 이때 eax와 eax와 and연산을 하고 연산값의 결과가 1이기 때문에 ZF의 값은 0으로 설정되게 된다.

00401068	85C0	test eax,eax	
0040106D	74 0F	je 04.40107E	
0040106F	68 24104300	push 04.431024	
00401074	E8 17710000	call 04.408190	
00401079	83C4 04	add esp,4	
0040107C	EB 0D	jmp 04.40108B	
0040107E	68 1C104300	push 04.43101C	
00401083	E8 08710000	call 04.408190	
00401088	83C4 04	add esp,4	
0040108B	EB BB	jmp 04.401048	
0040108D	CC	int3	
0040108E	CC	int3	
0040108F	CC	int3	
00401090	CC	int3	
00401091	CC	int3	
00401092	CC	int3	

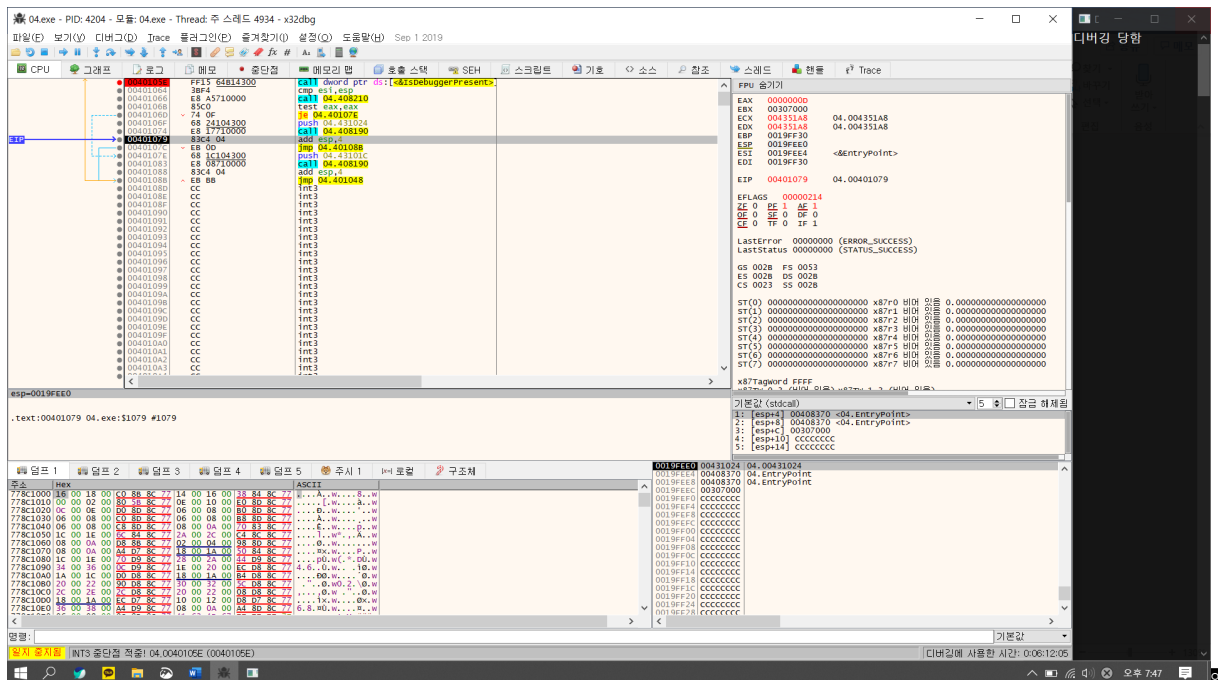
  

EAX	0019E000
EBX	003EB000
ECX	19D5BE32
EDX	00000000
EBP	0019FF30
ESP	0019FEE4 <&EntryPoint>
ESI	0019FEE4 <&EntryPoint>
EDI	0019FF30
EIP	0040106D 04.0040106D
EFLAGS	00000202
ZF	0 PF 0 AF 0
OF	0 SF 0 DF 0
CF	0 TF 0 IF 1

그리고 이때 ZF가 0이기 때문에 je 04.40107E는 작동하지 않고 실행이 된다.

Stack에 데이터를 push하고 그다음 call를 하는 것을 볼수 있다.

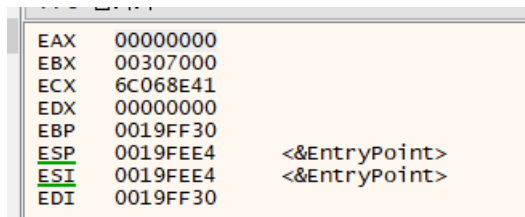
0040106D	74 0F	je 04.40107E	
0040106F	68 24104300	push 04.431024	
00401074	E8 17710000	call 04.408190	
00401079	83C4 04	add esp,4	
0040107C	EB 0D	jmp 04.40108B	
0040107E	68 1C104300	push 04.43101C	
00401083	E8 08710000	call 04.408190	
00401088	83C4 04	add esp,4	
0040108B	EB BB	jmp 04.401048	
0040108D	CC	int3	



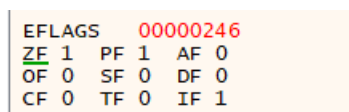
또한 이때 call를 한 다음 디버깅 다함 이라는 문자열이 출력되는 것을 보고 call 04.408190은 문자를 출력 해주는 역할 이라는 것을 알 수 있다. 2

그다음 add esp,4를 통해 가르키고 있는 변수를 그전에 가르키던 변수로 변경한다. 2

그다음부터는 jmp를 통해 다시 돌아와 반복 하게 된다. 이때 IsDebuggerPresent 함수의 리턴값을 1로 변경을 하여 분석 을 한다. 2

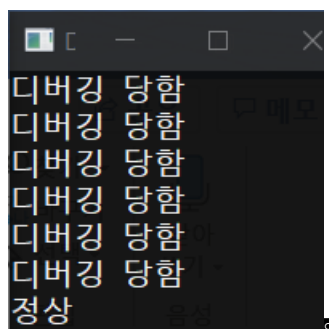


Test eax,eax를 실행하기 전은 그전과 동일하게 작동을 한다. 하지만 test eax,eax한 결과값이 ZF에 1로 저장되는 것을 알 수 있다. 2



00401048	8BF4	mov esi,esp
0040104A	68 E8030000	push 3E8
0040104F	FF15 68B14300	call dword ptr ds:[<&Sleep>]
00401055	3BF4	cmp esi,esp
00401057	E8 B4710000	call 04.408210
0040105C	8BF4	mov esi,esp
0040105E	FF15 64B14300	call dword ptr ds:[<&IsDebuggerPresent>]
00401064	3BF4	cmp esi,esp
00401066	E8 A5710000	call 04.408210
0040106B	85C0	test eax,eax
0040106D	74 0F	je 04.40107E
0040106F	68 24104300	push 04.431024
00401074	E8 17710000	call 04.408190
00401079	83C4 04	add esp,4
0040107C	EB 0D	jmp 04.40108B
0040107E	68 1C104300	push 04.43101C
00401083	E8 08710000	call 04.408190
00401088	83C4 04	add esp,4
0040108B	EB BB	jmp 04.401048

그 다음 je를 하게 되어 이번에는 그전과 다른 값을 stack에 push한다.



그다음 출력되는 결과는 “정상”으로 출력 되는 것을 알 수 있다. 이러한 분석을 통해 디버깅 체크를 하는 함수는 “IsDebuggerPresent”라는 것을 알 수 있다.

**정답: IsDebuggerPresent**