

CodeEngn Reverse Challenge

Basic RCE #4

Reverse L04 Start

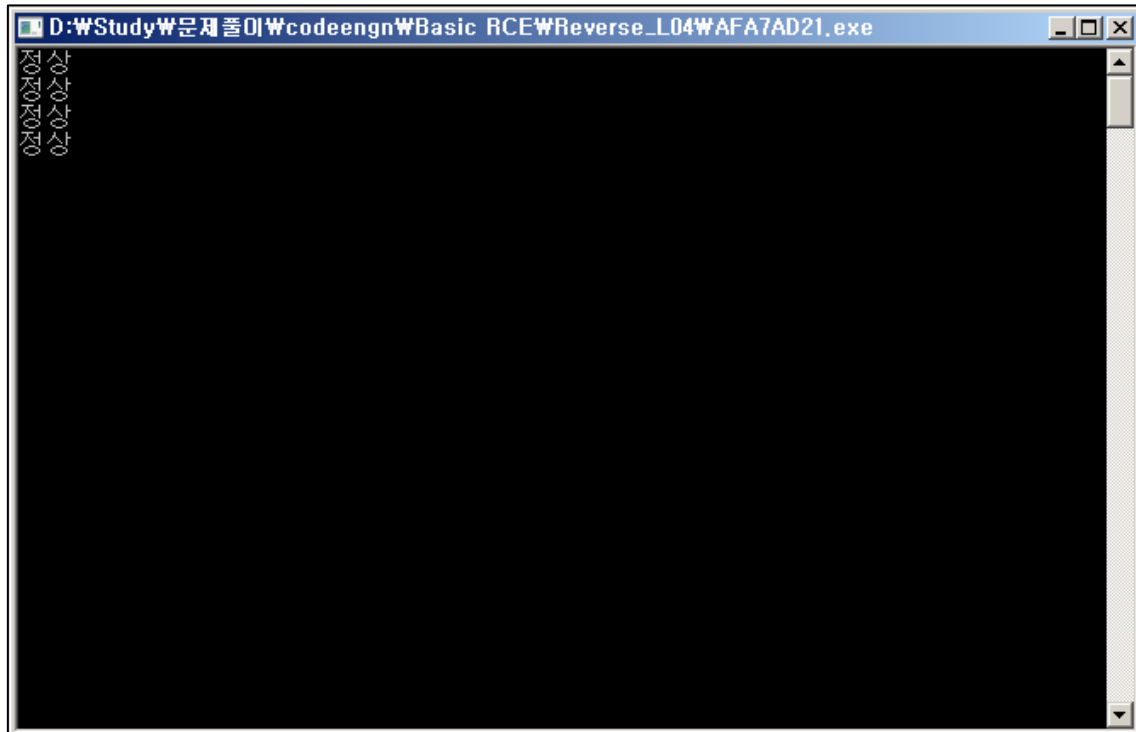
Author : CodeEngn / Lee Kang-Seok

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

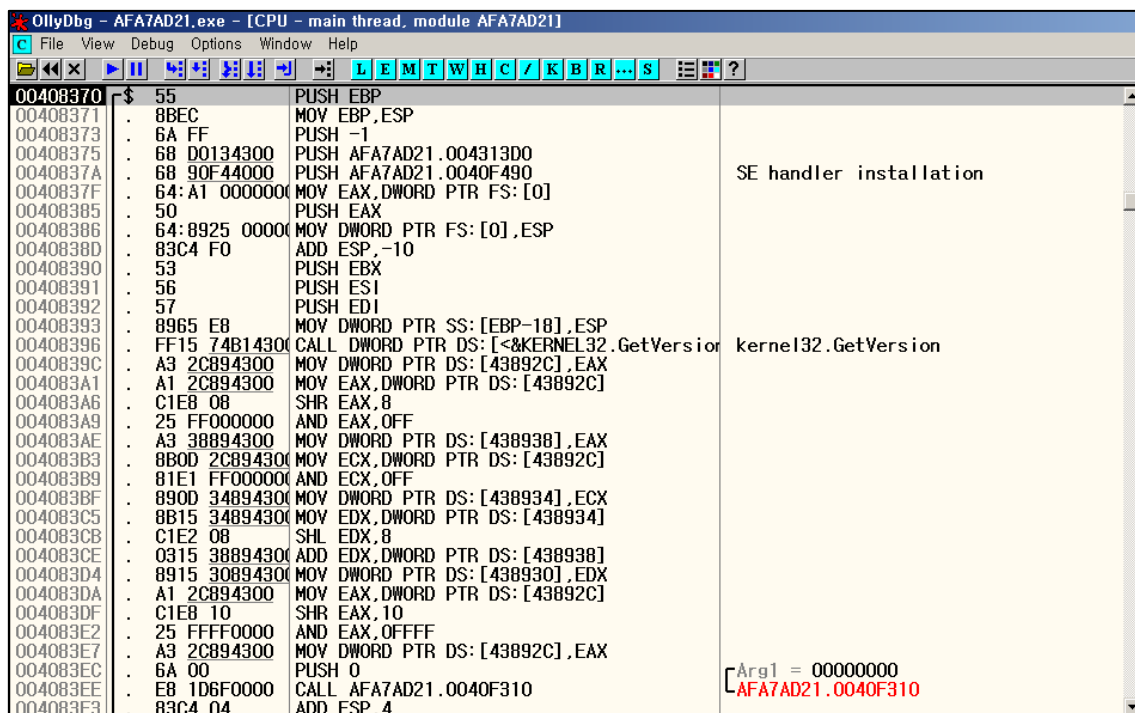
English :
This program can detect debuggers. Find out the name of the debugger detecting function the program uses.

[Down](#)

디버거 탐지 함수 이름을 묻고 있습니다. 우선 프로그램을 독립으로 실행했을 때와 올리디버거에 붙여서 실행했을 때의 결과가 달랐습니다.



디버거에 붙여서 실행하지 않으면 위와 같이 정상메시지가 디버거에 붙여서 붙여서 실행했을 시 "디버깅 당함" 이라는 메시지가 나타났습니다.



프로그램 동작이 단순한 것을 확인했기 때문에 step by step 접근으로 알아내려고 삼을 들었지만 결국 디버깅을 감지하는 함수를 찾을 수 없어 IDA 로 의사 코드를 얻었습니다.

```
void __cdecl main_0()
{
    int v0; // eax@2
    int v1; // eax@2
    int v2; // [sp+0h] [bp-4Ch]@2
    char v3; // [sp+Ch] [bp-40h]@1
    int v4; // [sp+4Ch] [bp+0h]@2

    memset(&v3, -858993460, 0x40u);
    while ( 1 )
    {
        Sleep(0x3E8u);
        _chkesp(&v2 == &v2, v0, (unsigned int)&v4);
        v1 = IsDebuggerPresent();
        if ( _chkesp(&v2 == &v2, v1, (unsigned int)&v4) )
            printf("");
        else
            printf("정");
    }
}
```

의사코드는 예상했던 데로 단순했습니다. 한글이 깨져서 메시지 출력 부분이 제대로 표시되지 않지만 "정" 이라고 표시된 부분이 "정상"을 출력하는 것으로 예상할 수 있습니다. 그리고 눈에 띄는 IsDebuggerPresent 함수를 올리디버거로 확인해 보았습니다.

Names in AFA7AD21			
Address	Section	Type	Name
0043B1B4	.idata	Import	KERNEL32.InterlockedIncrement
0043B22C	.idata	Import	KERNEL32.IsBadCodePtr
0043B188	.idata	Import	KERNEL32.IsBadReadPtr
0043B184	.idata	Import	KERNEL32.IsBadWritePtr
0043B164	.idata	Import	KERNEL32.IsDebuggerPresent
0043B240	.idata	Import	KERNEL32.LCMapStringA
0043B244	.idata	Import	KERNEL32.LCMapStringW
0043B1B0	.idata	Import	KERNEL32.LoadLibraryA
00408370	.text	Export	<ModuleEntryPoint>
0043B16C	.idata	Import	KERNEL32.MultiByteToWideChar
0043B1A8	.idata	Import	KERNEL32.OutputDebugStringA

References in AFA7AD21:.text to KERNEL32.IsDebuggerPresent	
Address	Disassembly
0040105E	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]
004013E0	JMP DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]

Ctrl + N 기능으로 심볼 이름 리스트에서 isDebuggerPresent 함수 이름을 확인 할 수 있었고 Find references to import 기능으로 심볼 이름이 참조되는 위치를 찾아 브레이크 포인트를 설정하고 분기시의 참조되는 플래그를 임의로 조작해서 "정상", "디버깅 당함" 메시지를 선택적으로 출력할 수 있었습니다.

CPU - main thread, module AFA7AD21			
00401057	. E8 B4710000	CALL AFA7AD21.00408210	
0040105C	. 8BF4	MOV ESI,ESP	
0040105E	. FF15 64B14300	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	IsDebuggerPresent
00401064	. 3BF4	CMP ESI,ESP	
00401066	. E8 A5710000	CALL AFA7AD21.00408210	
0040106B	. 85C0	TEST EAX,EAX	
0040106D	. 74 0F	JE SHORT AFA7AD21.0040107E	
0040106F	. 68 24104300	PUSH AFA7AD21.00431024	Arg1 = 00431024
00401074	. E8 17710000	CALL AFA7AD21.00408190	AFA7AD21.00408190
00401079	. 83C4 04	ADD ESP,4	
0040107C	. EB 0D	JMP SHORT AFA7AD21.0040108B	
0040107E	. 68 1C104300	PUSH AFA7AD21.0043101C	Arg1 = 0043101C
00401083	. E8 08710000	CALL AFA7AD21.00408190	AFA7AD21.00408190
00401088	. 83C4 04	ADD ESP,4	
0040108B	. EB BB	JMP SHORT AFA7AD21.00401048	

D:\Study\문제풀이\code\engn\Basic RCE\Reverse_L04\AFA7AD21.exe		
0040105E	정상	
0040106D	디버깅 당함	
0040107E	정상	
00401083	디버깅 당함	
00401088	정상	
0040108B	디버깅 당함	
00401097	CC	INT3
00401098	CC	INT3
00401099	CC	INT3
0040109A	CC	INT3

명령어 JE를 JMP로 변경 후 바이너리를 새로 생성한 후 디버거에 붙여서 실행해보니 "정상" 메시지만 확인 될 뿐 디버깅 여부를 확인하지 못했습니다. 이번 문제의 정답은 isDebuggerPresent 입니다.