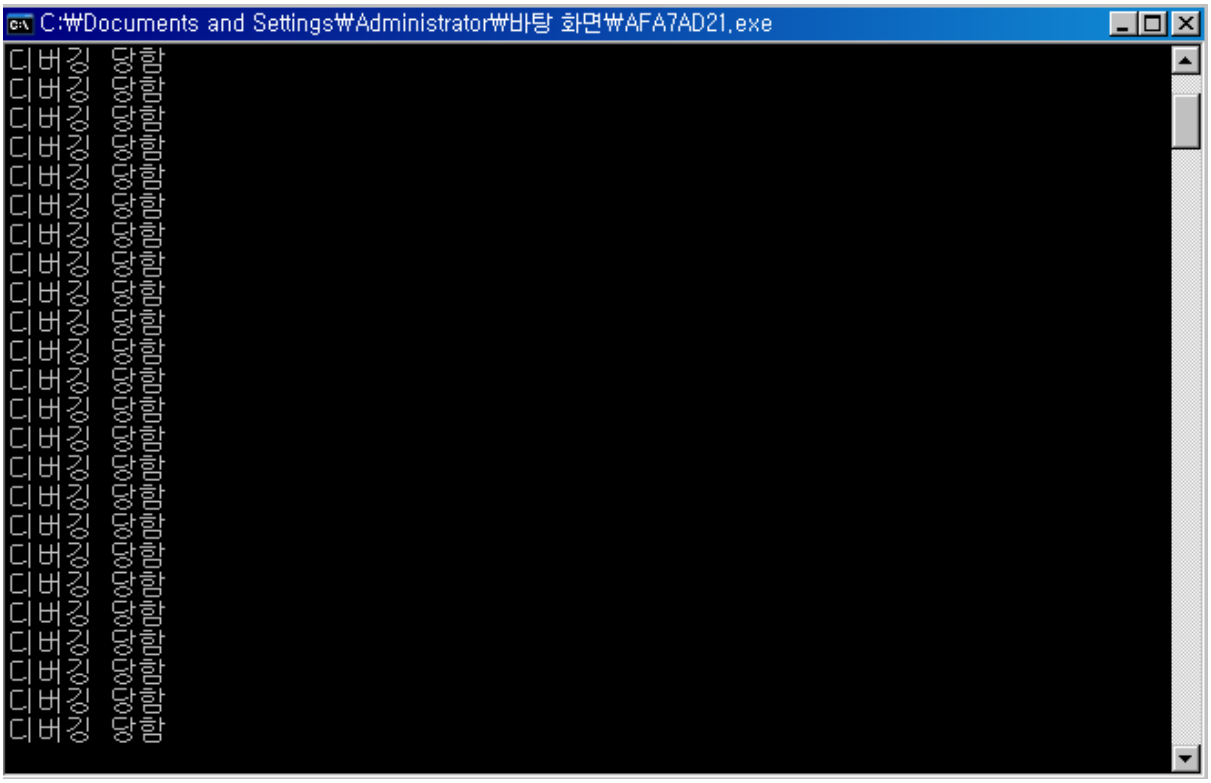
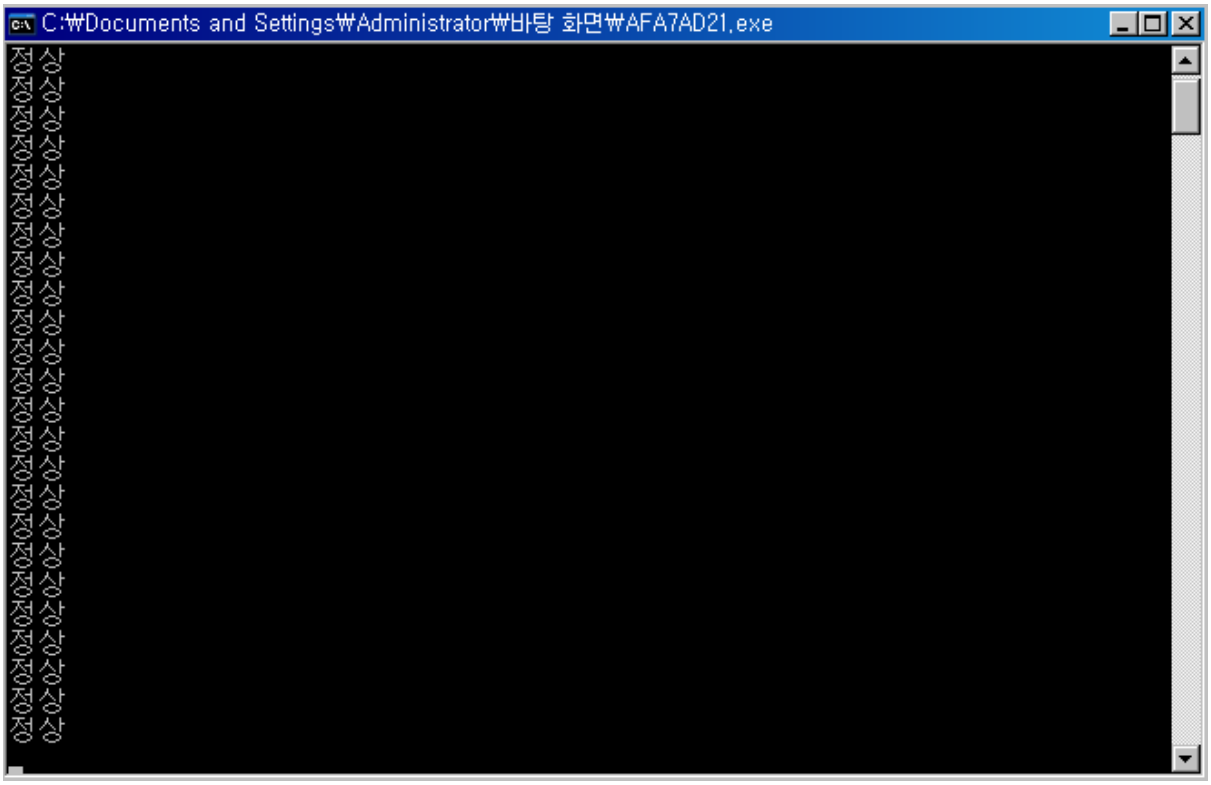
Basic RCE 04

작성자	koromoon (koromoon@naver.com)
작성일	2011-08
문제	이 프로그램은 디버거 프로그램을 탐지하는 기능을 갖고 있다. 디버거를 탐지하는 함수의 이름은 무엇인가?
정답	IsDebuggerPresent

(1) 설명



[그림 - 디버거에서 실행시]



[그림 - 그냥 실행시]

디버거를 실행시와 그냥 실행시 각각 반응하는게 다름.
해당 프로그램이 디버거를 탐지하는 듯 함.

(2) 문제 프로그램의 예상 코드

```
#define _WIND32_WINNT 0x0501
#include <iostream>
#include <windows.h>
void main()
{
    for(;;) {
        Sleep(1000);
        if (IsDebuggerPresent())
            printf("디버깅당함₩n");
        else
            printf("정상₩n");
    }
}
```

Minimum system required	Minimum value for _WIN32_WINNT and WINVER
Windows 7	_WIN32_WINNT_WIN7 (0x0601)
Windows Server 2008	_WIN32_WINNT_WS08 (0x0600)
Windows Vista	_WIN32_WINNT_VISTA (0x0600)
Windows Server 2003 with SP1, Windows XP with SP2	_WIN32_WINNT_WS03 (0x0502)
Windows Server 2003, Windows XP	_WIN32_WINNT_WINXP (0x0501)
Windows 2000	_WIN32_WINNT_WIN2K (0x0500)

#define _WIND32_WINNT의 경우 운영체제마다 다름.
위 표를 참조하여, 각 운영체제 맞게 코딩하여 컴파일하면 됨.

(3) IsDebuggerPresent 함수

정의

Anti-Reversing 기법 중에서 가장 기초적인 방법 중의 하나이며 **현재 실행 중인 프로세스에 디버거가 붙은 있는지, 즉 현재 디버거에 의해서 디버깅 되고 있는지에 대한 결과를 TRUE/FALSE로 리턴해 주는 함수임.**

해당 프로세스가 디버깅을 당하고 있는지 여부를 PEB 구조체의 디버깅 상태값을 확인하여 디버깅을 당하고 있다면 1을 리턴하게 되고, 아닐 경우 0을 리턴하게 됨.
이 함수로는 커널모드 디버거는 탐지하지 못하며 유저모드 디버거만 탐지가 가능함.
kernel32.dll에서 exports되는 함수이며 보통 프로그램의 보호 차원에서 쓰이지만 우회할 수 있는 방법은 많음.

문법

```
BOOL WINAPI IsDebuggerPresent(void);
```

참고 사이트 : http://msdn.microsoft.com/en-us/library/ms680345(VS.85).aspx

IsDebuggerPresent()함수가 수행하는 코드

```
mov eax, dword ptr fs:[18]      // TEB 구조체 시작주소
mov eax, dword ptr ds:[eax+30]  // PEB 구조체(TEB+30) 시작주소
movzx eax, byte ptr ds:[eax+2]  // BeingDebugged flag 값(PEB+2)
retn                            // BeingDebugged flag 값 리턴
```

여기서 BeingDebugged flag 값은 디버깅 중인지에 대한 값임.
TEB는 Thread Environment Bloack의 약자로써 Thread에 대한 정보가 저장되어 있는 구

조체이며 PEB는 Process Environment Bloack의 약자로써 Process에 대한 정보가 저장되어 있는 구조체임. (PEB와 TEB에 대한 정보는 구글링으로...)

여기서 BeingDebugged 필드는 IsDebuggerPresent 함수가 디버깅 중인지 아닌지를 이 필드를 통해 참조함.

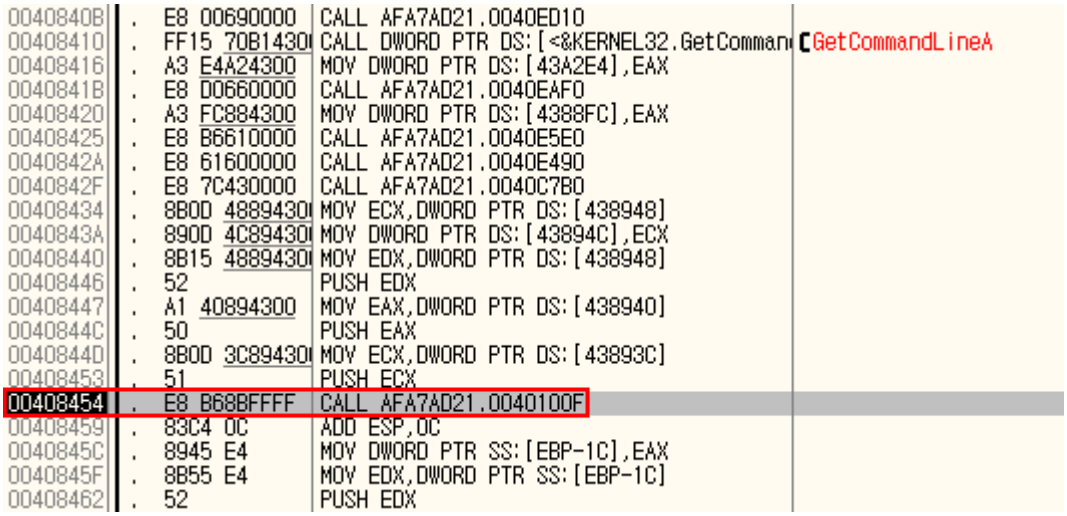
inline 어셈블리 코드로 구현한 IsDebuggerPresent()함수

```
BOOL AntiHackIsDebuggerPresent (void)
{
    BOOL bRet = TRUE;
    __asm
    {
            MOV     EAX, FS:[00000018H]
            MOV     EAX, DWORD PTR [EAX+030H]
            MOVZX   EAX, BYTE PTR [EAX+002H]
            MOV     bRet, EAX
    }
    return bRet;
}
```

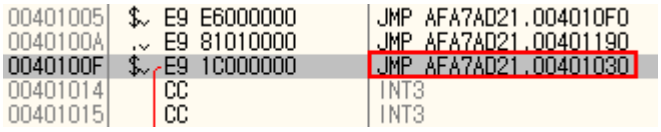
위 코드는 Debugging Applications for Microsoft .NET and Microsoft Windows by John Robbins책을 인용함.

Native 응용 프로그램에서 IsDebuggerPresent API 를 사용하는 경우는 아마도 Debugger를 통한 로직 검사를 막기 위해서일 것임. 하지만, 이 API 를 호출한다고 해서 100% 막을 수는 없음. 해당 책에서 위와 같은 inline 어셈블리 코드를 사용할 것을 제안함. (어느 정도 효과는 볼 수 있음. ^^)

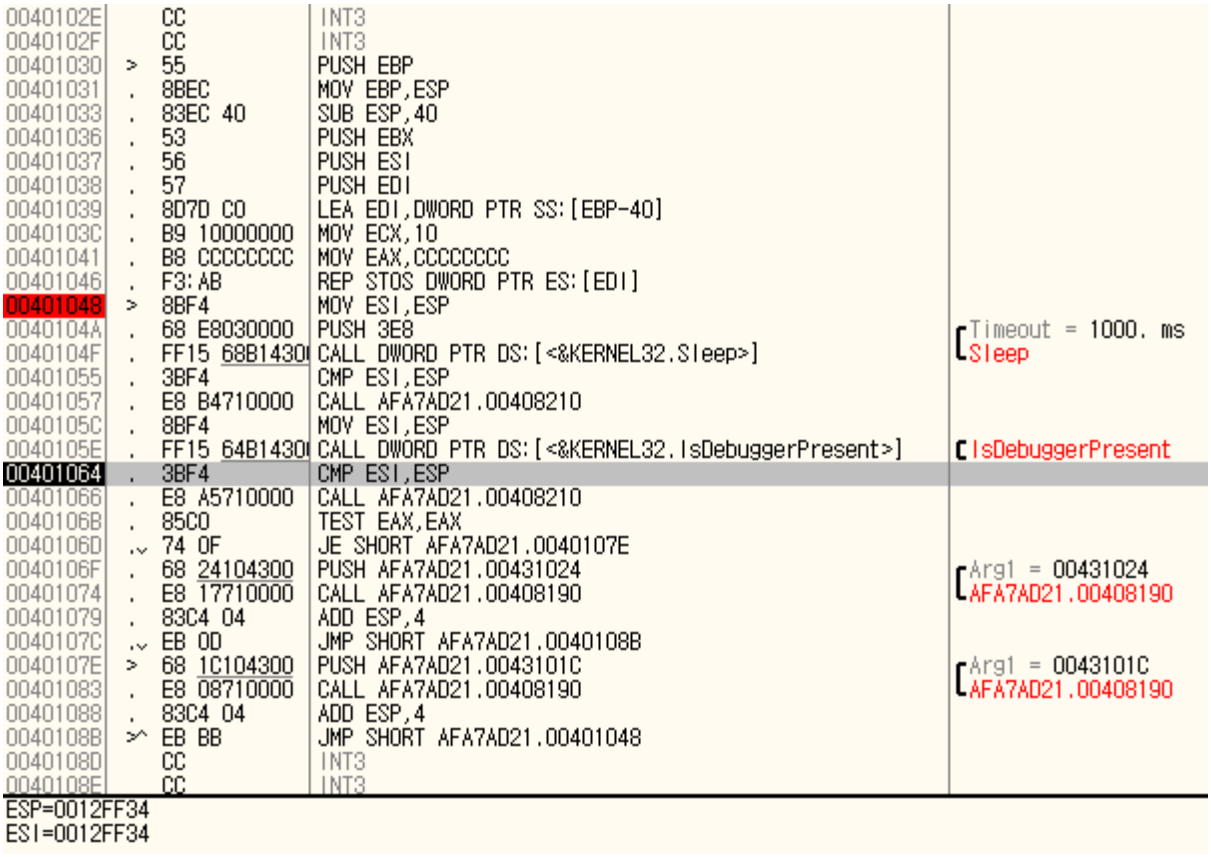
(4) 크랙 파일을 저장할 수 있는 크랙 방법



F8을 계속 눌러서 실행하다보면 어디선가 멈춤. 그리고 F8을 눌러도 더 이상 변하지 않는 점으로 보아 루프를 돌고 있을 것으로 추측을 하게 됨.

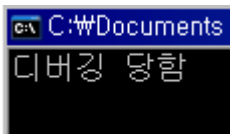


해당 0040100F에 가보면 다시 00301030으로 점프하게 됨.

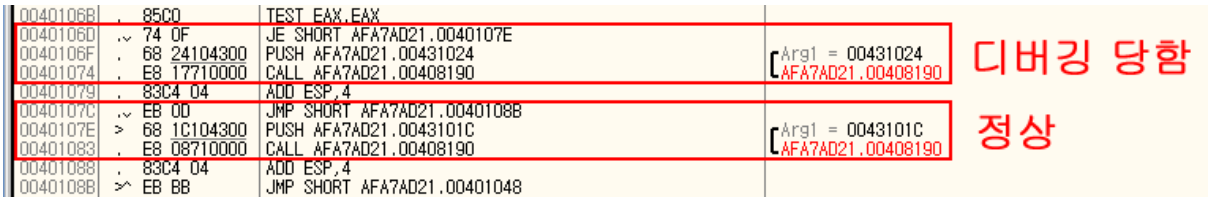


00401048에 Breakpoint(F2)를 검. 그리고 F8를 계속해서 눌러보면 0040105E에 위치한

IsDebuggerPresent 함수를 지나고 00401064에서 EAX값을 보면 1로 변함.
(참고로 현재 디버깅의 사용중이므로 IsDebuggerPresent 함수의 리턴값이 1임. 그래서 EAX가 1로 셋팅됨. 아래 IsDebuggerPresent 함수를 참고 바람.)

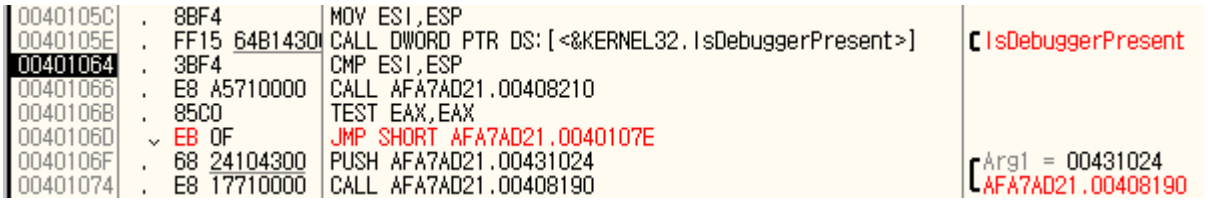


다시 F8를 계속해서 눌러보면 00401074의 CALL문를 지나면 CMD창의 디버깅 당함이라는 문구가 생김. 이런 현상으로 계속 루프를 돌게 됨.(00401048 ~ 0040108B)

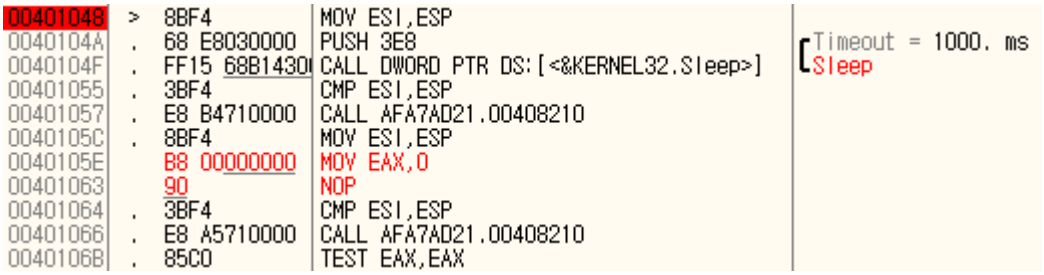


00401060 ~ 00401074 부분이 "디버깅 당함" 출력 코드라 추측되고 0040107C ~ 00401083 부분이 "정상" 출력 코드라 추측됨.

2가지 방법으로 크랙을 하면 됨. (크랙한 파일을 저장 가능함.)



첫 번째로 0040106D의 코드문에서 JE 대신 JMP로 바꾸어서 "정상" 출력 코드 부분으로 무조건 분기시키는 방법임.

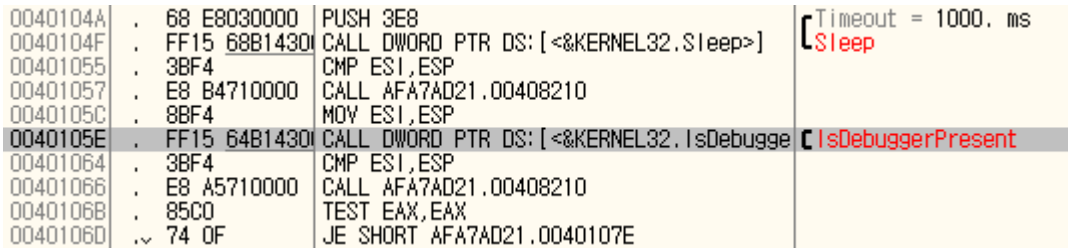


두 번째로 0040105E의 IsDebuggerPresent 함수 코드문을 위와 같이 "MOV EAX, 0", "NOP" 로 바꿔서 EAX가 0이 되므로 "정상" 출력 코드 부분으로 분기됨.

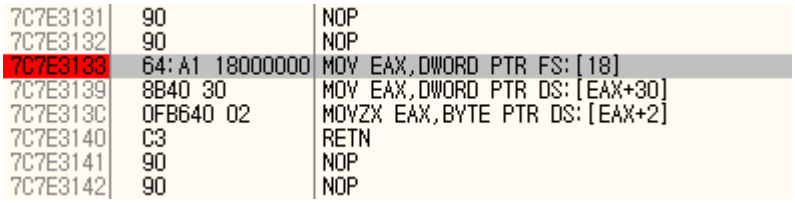
두 가지 방법을 통해서 "정상" 메시지가 출력되는 걸 볼 수가 있음.

(5) 크랙 파일을 저장할 수 없는 크랙 방법

IsDebuggerPresent()함수가 수행하는 코드 부분을 수정하면 "정상" 메시지가 출력할 수 있는 방법이 있으나 크랙된 파일을 저장할 수 없음. OllyDbg는 유저 모드 디버거이기 때문임.

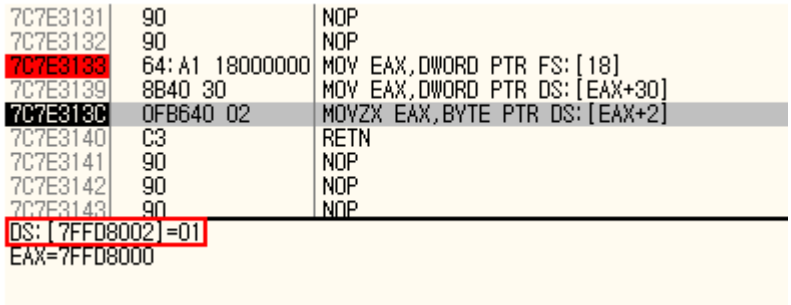


0040105E의 IsDebuggerPresent 함수 코드문에서 엔터를 누름.

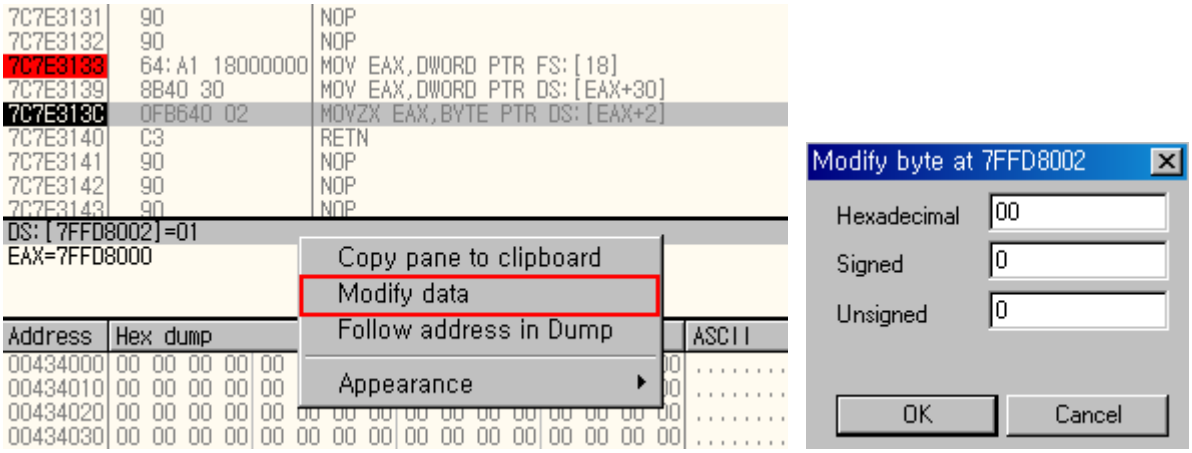


7C7E3133에서 Breakpoint(F2)를 검.

F8를 계속 눌러서 해당 IsDebuggerPresent 함수 코드문 부분에서의 변화를 살펴보자!



7C7E313C의 IsDebuggerPresent 함수의 리턴값이 디버깅 중임을 나타내는 1로 되어 있는 걸 볼 수가 있음.



DS: [7FFD8002]=01 부분에서 오른쪽 마우스를 눌러서 Modify data 를 클릭 00으로 바꿔주자! (0은 디버깅 아님을 나타냄.)

그리고 "정상" 메시지가 출력되는 걸 볼 수가 있음.