

SECCON_2016_CTF

Anti-debugger Write Up

Author : hideroot(M.O.K)

Data : 2017/12/01

이번문제는 꽤 재밌는 문제다 그리고 딱 100점 자리 문제가 맞다.

문제에서 원하는 건 " 리버싱 하면서 코드 패치 할 줄 알아요? " 를 묻는 문제다.

그래서 굉장히 간단하다. 일단 bin 파일을 구글링으로 다운 받는다. 다음에 IDA로 열어 보았더니 아래와 같다.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    FILE *v3; // eax@1
    HANDLE v4; // eax@8
    int v11; // [sp+C4h] [bp-A8h]@20
    int v12; // [sp+D0h] [bp-9Ch]@35
    DWORD v13; // [sp+D4h] [bp-98h]@11
    LPCSTR lpFileName; // [sp+D8h] [bp-94h]@17
    BOOL pbDebuggerPresent; // [sp+DCh] [bp-90h]@8
    int v16; // [sp+E0h] [bp-8Ch]@14
    int v17; // [sp+E4h] [bp-88h]@35
    int i; // [sp+E8h] [bp-84h]@11
    int v19; // [sp+ECH] [bp-80h]@35
    int v20; // [sp+F0h] [bp-7Ch]@35
    char v21[4]; // [sp+F8h] [bp-74h]@1
    int v22; // [sp+108h] [bp-64h]@1
    char v23; // [sp+10Ch] [bp-60h]@1
    char v24; // [sp+10Dh] [bp-5Fh]@1
    CPPEH_RECORD ms_exc; // [sp+154h] [bp-18h]@35

    v23 = 0;
    memset(&v24, 0, 0x3Fu);
    v22 = 1;
    printf("Input password >");
    v3 = (FILE *)sub_40223D();
    fgets(&v23, 64, v3);
    strcpy(v21, "I have a pen.");
    v22 = strncmp(&v23, v21, 0xDu);
    if ( !v22 )
    {
        puts("Your password is correct.");
        if ( IsDebuggerPresent() == 1 )
        {
            puts("But detected debugger!");
            exit(1);
        }
        if ( sub_401120() == 112 )
        {
            puts("But detected NtGlobalFlag!");
            exit(1);
        }
        v4 = GetCurrentProcess();
    }
```

앞부분만 잘라 왔다. 뒤에도 비슷하기 때문에 IDA가 너무 좋아서 패스워드를 바로 뱉어 버렸는데 다른 디버거들은 한참 걸렸을 듯 싶다. 보니까 디버거를 검사를 한다. 처음에는 I have a pen. 입력했는데 Corret 뜨고 아무 소식이 없더라.

```
Windows PowerShell
PS C:\Users\shin\Desktop\system_hacking\system_hacking\secon_2016> .\bin.exe
Input password >I have a pen.
Your password is correct.
PS C:\Users\shin\Desktop\system_hacking\system_hacking\secon_2016>
```

그래서 여기가 문제가 아니라는 걸 알수 있다. 다시 IDA에 attach시켜서 실행을 지켜보자.

```
.text:00401399 add     esp, 0Ch
.text:0040139C mov     [ebp+var_64], eax
.text:0040139F cmp     [ebp+var_64], 0
.text:004013A3 jnz     loc_40174F
EIP .text:004013A9 mov     [ebp+var_A8], 0
.text:004013B3 push    offset aYourPasswordIs ; "Your password is correct."
.text:004013B8 call    _puts
.text:004013BD add     esp, 4
.text:004013C0 call    ds:IsDebuggerPresent
.text:004013C6 mov     [ebp+var_A8], eax
.text:004013CC cmp     [ebp+var_A8], 1
.text:004013D3 jnz     short loc_4013E9
.text:004013D5 push    offset aButDetectedDeb ; "But detected debugger!"
.text:004013DA call    _puts
.text:004013DF add     esp, 4
.text:004013E2 push    1 ; int
.text:004013E4 call    _exit
.text:004013E9 ; -----
.text:004013E9 loc_4013E9: ; CODE XREF: _main+E31j
.text:004013E9 call    sub_401120
.text:004013EE mov     [ebp+var_A8], eax
.text:004013F4 cmp     [ebp+var_A8], 70h
.text:004013FB jnz     short loc_401411
.text:004013FD push    offset aButDetectedNtg ; "But detected NtGlobalFlag?"
.text:00401402 call    _puts
.text:00401407 add     esp, 4
.text:0040140A push    1 ; int
.text:0040140C call    _exit
```

브포 걸어주고 하나씩 따라가면서 보자. 0x4013D3에서 보면 위의 CMP에서 검사하고 디버거가 붙어 있는지 보고 붙어 있으면 종료 시켜버린다. 그럼 이 루틴을 하나씩 제거 하면서 가보자. 여기서 여러가지 방법이 있는데 jnz, jb 이런 OP 들을 JMP로 바꿔버리기가 있고 OP(Over Power) 기능을 사용하는 IDA를 쓰는 법이 있다. IDA로 해보자 IDA가 편하다. IDA pro 에서 Ctrl + N 을 누르면 EIP를 현재 커서로 옮겨 버릴 수가 있다.(완전 X사기 ...) 이렇게 해서 디버거 검사 루틴들을 전부다 돌아가보면

```

.text:0040164D call    _exit
.text:00401652 ; -----
.text:00401652
.text:00401652 loc_401652:                                ; CODE XREF: _main+34C↑j
.text:00401652 mov     [ebp+var_78], 0
.text:00401659 cmp     [ebp+var_78], 1
.text:0040165D jnz     loc_40174D
.text:00401663 mov     ecx, 7
EIP> .text:00401668 mov     esi, offset aAj@Jq7hbotH?u8 ; ";aj&@:JQ7HB0t[h?U8aCBk]0aI38" |
.text:0040166D lea     edi, [ebp+var_CC]
.text:00401673 rep movsd
.text:00401675 movsb
.text:00401676 xor     ecx, ecx
.text:00401678 mov     [ebp+var_AF], ecx
.text:0040167E lea     edx, [ebp+var_CC]
.text:00401684 mov     [ebp+var_D8], edx
.text:0040168A mov     [ebp+Text], 0
.text:00401691 push    7Fh ; size_t
.text:00401693 push    0 ; int
.text:00401695 lea     eax, [ebp+var_157]
.text:0040169B push    eax ; void *
.text:0040169C call    _memset
.text:004016A1 add     esp, 0Ch

```

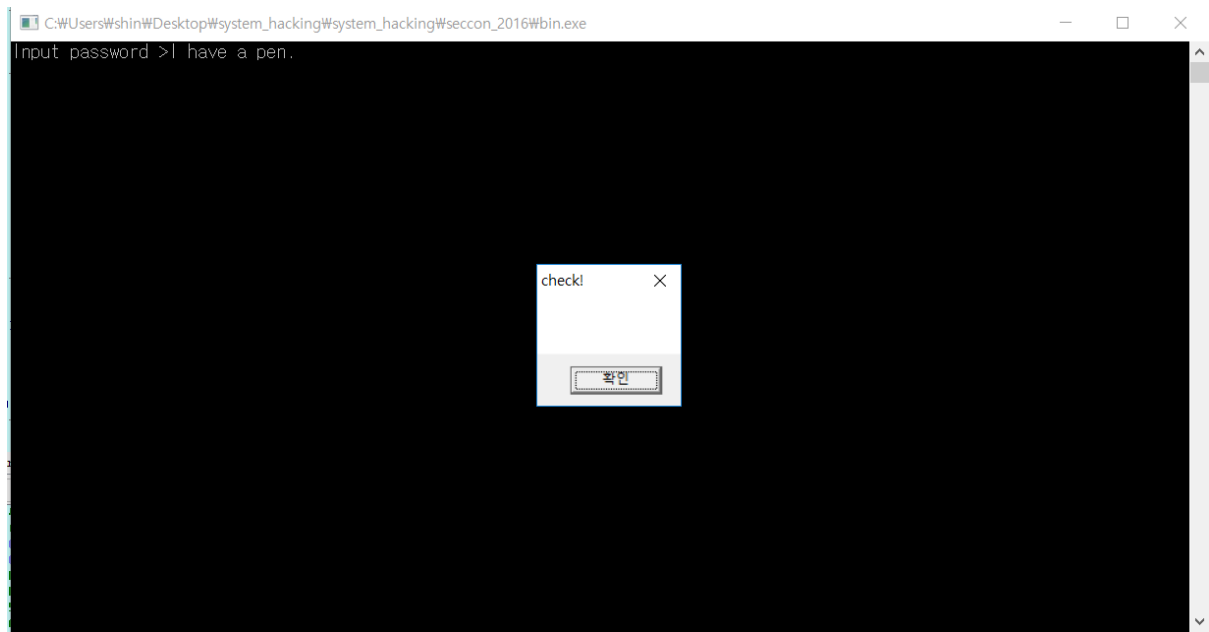
이런 부분을 만날수 있다. 딱 봐도 엄청 의심스러워 보인다.

```

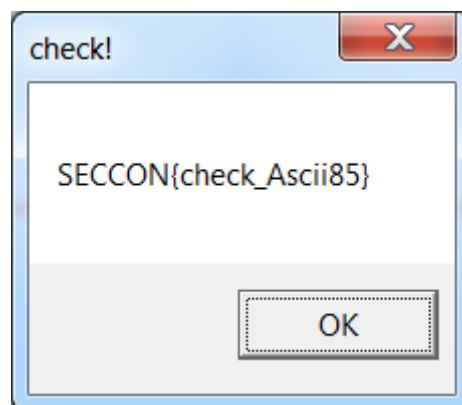
.text:00401735 jmp     snort loc_4016D1
.text:00401737 ; -----
.text:00401737
.text:00401737 loc_401737:                                ; CODE XREF: _main+41A↑j
.text:00401737 ; _main+443↑j
EIP> .text:00401737 push    0 ; uType
.text:00401739 push    offset Caption ; "check?"
.text:0040173E lea     ecx, [ebp+Text]
.text:00401744 push    ecx ; lpText
.text:00401745 push    0 ; hWnd
.text:00401747 call    ds:MessageBoxA
.text:0040174D
.text:0040174D loc_40174D:                                ; CODE XREF: _main+36D↑j
.text:0040174D jmp     short loc_40175C

```

좀더 내려가다 보면 Check! 라고하고 window 창을 띄어주는 루틴이 숨어 있었다! 아마 여기가 답인듯 싶다.



지금 이상태는 한번에 EIP를 넘겨서 중간에 문자열이나 여러 가지 처리들을 안해서 그냥 빈박스만 나온다.



실제로 하나씩 넘겨 가면서 실행하면 아까 보였던 수상한 문자열 들을 Decode 해서 정상적으로 출력해준다.

결국 문제에서 원하는 능력은 코드 패치를 하는 능력이라는 걸 알 수 있다.

실제로 뜯어서 주욱 내려보면 문자열을 바로 발견해서 아마 Encoding 시켜 놓은 것 같다. 그래서 Decoding 루틴을 하나 짜놓은 것 같다. 자세하게 분석 해보면 Decoding 루틴이 있다.