

## [ 2014 Power Of XX Qual - Brokenwindow ]

이 문제는 윈도우 포너블 문제이다.

IDA로 바이너리를 열어보면 심볼이 거의 날아가 있다. 그렇지만 취약점은 너무 대놓고 있기 때문에 찾는 것은 어렵지 않다.

다음은 메인함수에서 발견된 취약점 부분이다.

```
else
{
    if ( v3 == 3 )
    {
        puts("[ ] HELLO? : ");
        sub_40167F("%s", &v10);
        puts("[*] YOU GOT PWNEED :)");
        return 1;
    }
    if ( !v3 )
    {
        puts("[-] bye");
        return 1;
    }
    puts("[?] WTF");
}
```

심볼은 없지만 sub\_40167F()는 왠지 scanf()함수일 것 같다.

입력을 받는 v10은 [ebp-0x38]에 위치하므로 0x38+4만큼 데이터를 넣으면 ret를 덮어써 eip를 잡을 수 있을 것 같다.

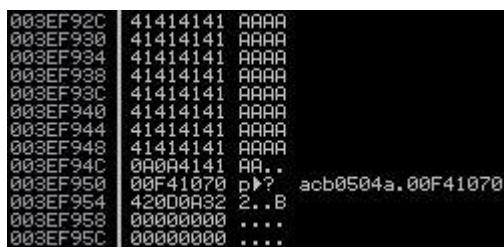
그런데 여기서 문제가 있다. 바로 stack\_security\_cookie가 존재한다는 것이다. 이것을 우회하기 위해서는 스택쿠키가 존재하는 [ebp-4]부분의 값을 leak해야 한다.

메모리를 leak 하는 부분도 쉽게 찾을 수 있다.

```
if ( v4 >= 1 )
{
    puts("[ ] Input Vulnerability Title : ");
    READ_40A587(0, &::WideCharStr, 0x12Cu);
    memmove(&v7, &::WideCharStr, 0x12Cu);
    puts("[ ] Input Details about your 0-day: ");
    READ_40A587(0, &word_4151C0, 0x1F4u);
    memmove(&v6, &word_4151C0, 0x1F4u);
    sub_40147B("[*] Title : %s\n[*] DESC : %s\n", (unsigned int)&v7);
    puts("[*] Thank you! you got 3000Won");
    ((void (__cdecl *)(void (*)(void), void (*)(void)))v8)(v8, v2);
    v5 += 3000;
    --v4;
}
```

v7은 0x12c 바이트만큼의 공간을 할당받았다. 그런데 v7에 0x12c 만큼의 데이터들을 써넣고 %s로 읽어주면 메모리가 leak 날 수밖에 없다.

이 부분에서 leak되는 메모리에는 어떤 값이 들어있는지 살펴보자.



003EF92C	41414141	AAAA
003EF930	41414141	AAAA
003EF934	41414141	AAAA
003EF938	41414141	AAAA
003EF93C	41414141	AAAA
003EF940	41414141	AAAA
003EF944	41414141	AAAA
003EF948	41414141	AAAA
003EF94C	0A0A4141	AA..
003EF950	00F41070	p? acb0504a.00F41070
003EF954	42000A32	2..B
003EF958	00000000	....
003EF95C	00000000	....

뒤에 있는 널바이트 때문에 많이 읽어오지 못하고 함수의 포인터까지는 읽어올 수 있을 것 같다.

```
char v7; // [sp+208h] [bp-16Ch]@13
void (*v8)(void); // [sp+334h] [bp-40h]@8
```

IDA에서 봐도 v7 바로 위에는 함수 포인터가 존재한다는 것을 알 수 있다.

이 함수 포인터에는 base+0x1070 의 주소가 들어있으므로 이 주소를 leak해 base의 주소를 구할 수 있다.

하지만 아직 스택쿠키를 leak하지는 못했다. 다른 곳에서는 leak이 날것같이 보이는 부분이 없으므로 스택쿠키를 다른 방법으로 우회해야 한다.

```
003EF948 41414141 AAAA
003EF94C 0A0A4141 AA..
003EF950 00F41070 p!>? acb0504a.00F41070
003EF954 420D0A32 2..B
003EF958 00000000 ....
003EF95C 00000000 ....
003EF960 00000000 ....
003EF964 00F45417 #T? acb0504a.00F45417
003EF968 003EF954 T? ASCII "201B"
003EF96C 00F457BA #? RETURN to acb0504a.00F457BA fro
003EF970 003EF9C8 #>.
003EF974 00F43890 ?? acb0504a.00F43890
003EF978 42C6CC13 !!B
003EF97C 00000000 ....
003EF980 00000000 ....
003EF984 00000000 ....
003EF988 003EF998 #>.
003EF98C 420D07FB ?..B
003EF990 003EF908 #>.
003EF994 00F4229F ?? RETURN to acb0504a.00F4229F fro
003EF998 00000001 0...
003EF99C 00477BF8 ?G.
003EF9A0 00477C78 #!G.
003EF9A4 420D07B3 ?..B
003EF9A8 00000000 ....
003EF9AC 00000000 ....
003EF9B0 7EFDE000 .#~
003EF9B4 003EF9C8 #>.
003EF9B8 00000000 ....
003EF9BC 00000000 ....
003EF9C0 003EF9A4 #!>.
003EF9C4 0000005B [...
003EF9C8 003EFA14 #?.. Pointer to next SEH record
003EF9CC 00F43890 ?? SE handler
003EF9D0 42C6CFDB #>..
```

스택을 살펴보던 중에 SEH 부분을 찾았다. 우리는 [ebp-0x38]에 원하는 만큼 데이터를 쓸 수 있으므로 SE handler까지는 충분히 덮어쓸 수 있다.

마침 nx도 걸려있지 않으니 셸코드를 넣어주고 해당 부분을 eip가 가리키게 하면 될 것 같다. 위의 메모리 leak부분을 다시 보면 굳이 데이터를 전역변수에 넣었다가 지역변수에 복사하는 것을 볼 수 있다. 메모리 leak에 쓰이지 않는 Detail 부분에 셸코드를 넣어주면 data영역에 존재하는 word\_4151C0에 셸코드가 들어가고 해당 부분으로 eip를 옮겨주면 될 것 같다.

[Exploit]

[메모리 leak + 셸코드 삽입] -> [overwrite SEH] -> exploit!

(코드 따로 첨부)