**实验三报告**

**实验名称：Return-to-libc Attack Lab**

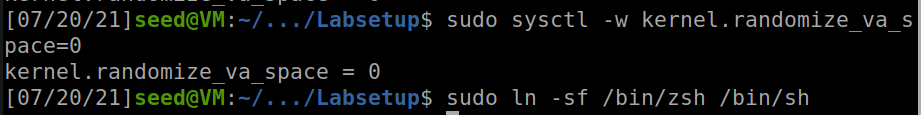
**基本信息：**

姓名：马运聪 学号：57119115 完成日期：2021.7.17

**实验内容：**

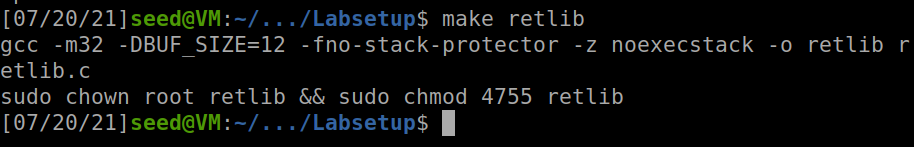
**环境配置**

关闭地址随机化和stackguard保护机制，同时将sh软链接到zsh

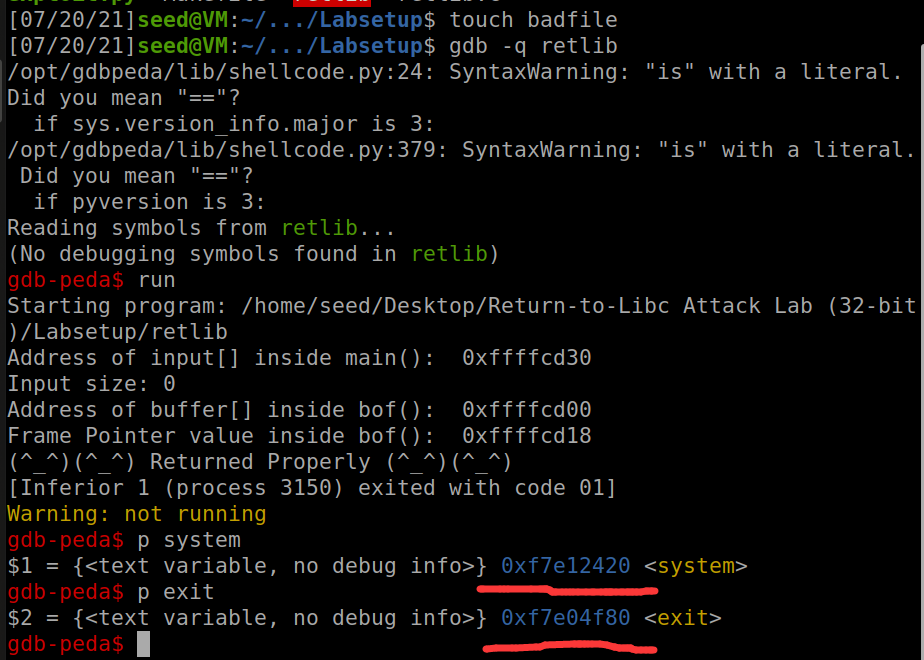


**Task1 Finding out the Addresses of libc Functions**

**编译retlib.c**

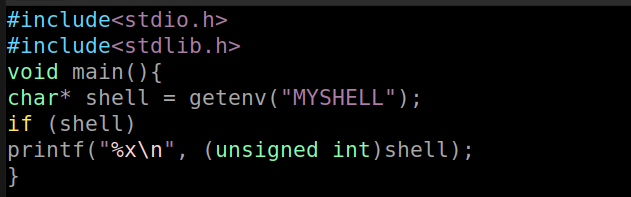


**使用gdb调试获得system和exit函数地址**

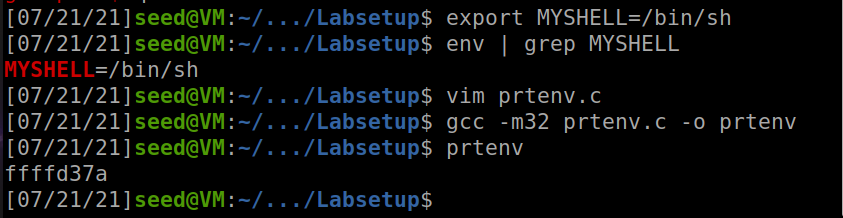


**Task2 Putting the shell string in the memory**

**创建并编译文件prtenv.c**

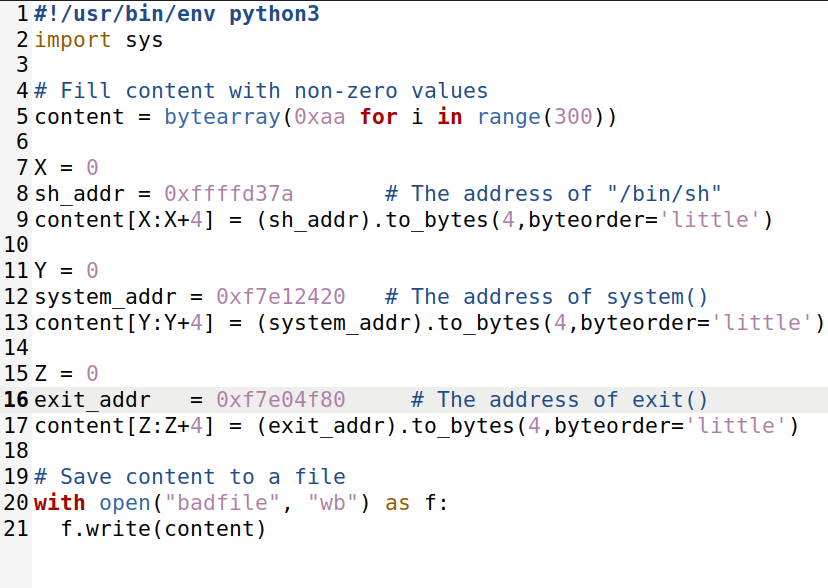


**设置环境变量MYSHELL，利用该环境变量获得/bin/sh的地址**

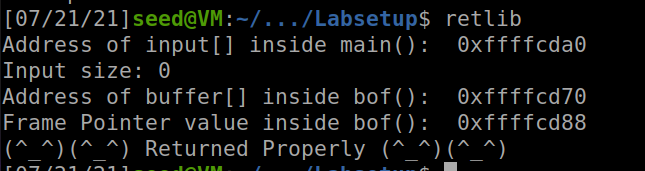


**Task3 Launching the Attack**

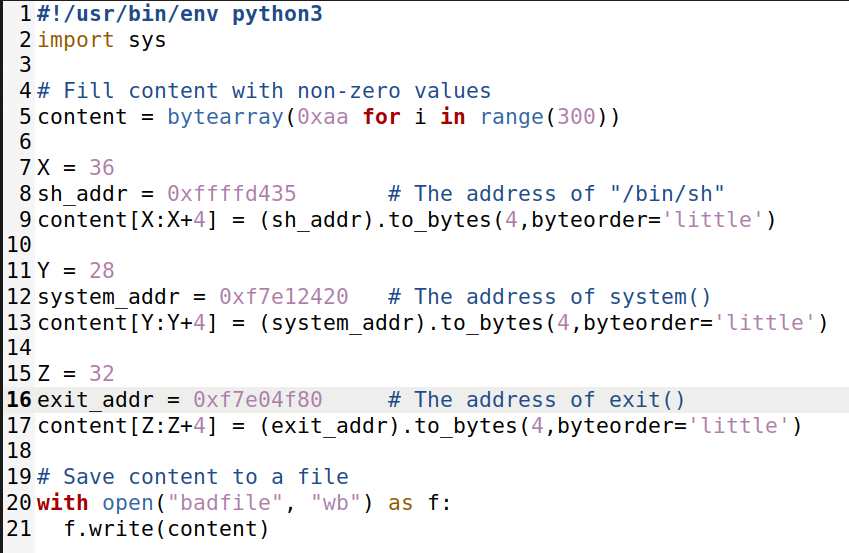
**修改exploit.py文件**

**e**

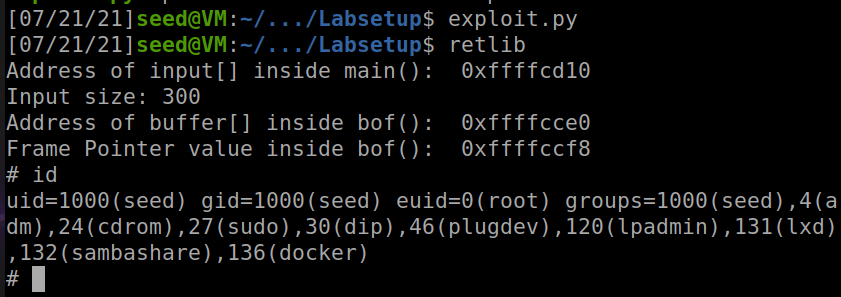
**运行retlib，查看ebp到bof（）函数中bufuffer的距离是24字**



**再次更改exploit.py中X、Y、Z的值**



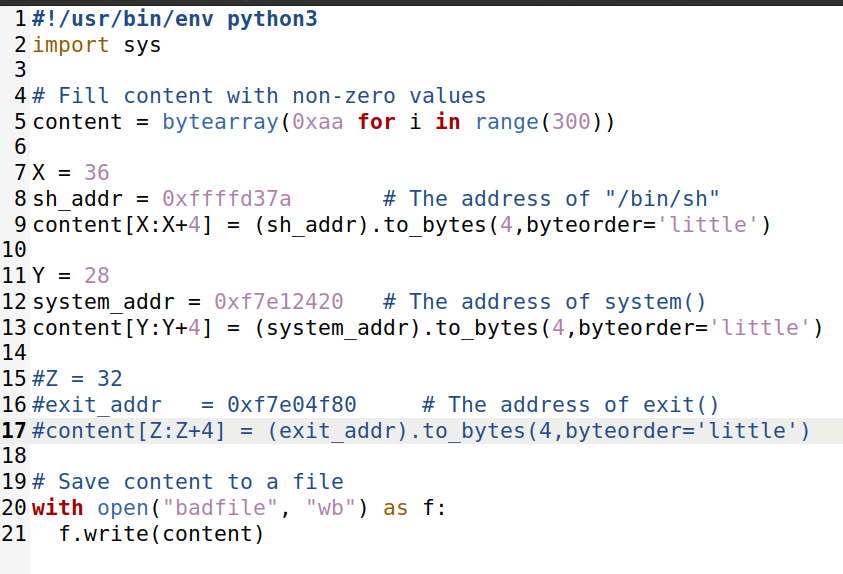
依次运行exploit.py和retlib

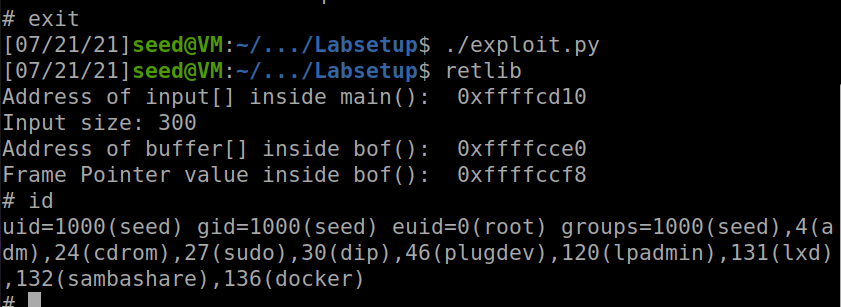


结果显示，有效uid为root，已获得root权限的shell，说明攻击成功

**Attack variation 1：**

更改exploit.py，去掉exith函数地址，再次攻击

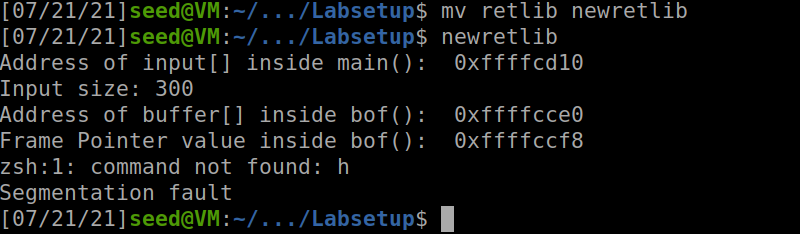




同样攻击成功

**Attack variation 2:**

更改了程序的名字，改为长度更长的newretlib，再次攻击



发现显示找不到命令，攻击失败。说明更改了程序名长度后，MYSHELL环境变量的地址发生了变化，所以使得原来指向/bin/sh字符串的地址指向了其他命令。

**实验总结：**

本次实验与lab2的缓冲区溢出漏洞有相似和不同指出，相同则是都是利用缓冲区溢出的漏洞来实现攻击，而lab2是将恶意代码放到栈中，本次实验则是通过改变返回地址，使目标程序跳转到栈外的代码中（内存中的libc库中system()）。当攻击成功时，程序会执行/bin/sh/产生一个root权限的shell。

而此次攻击实验的主要步骤为：

1. 关闭栈地址随机化和shell的保护机制
2. 找到syste（）函数和exit ()函数的地址；
3. 找到“/bin/sh“字符串的地址；
4. 把“/bin/sh“字符串的地址作为参数放到system()函数中，即把地址放到system()函数存放参数的地址中。