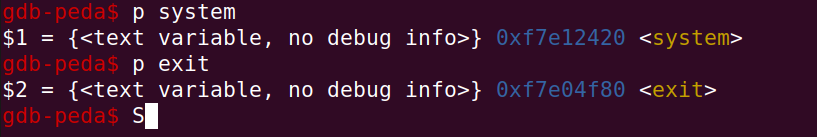
**lab3-report**

**57119122 刘恒睿**

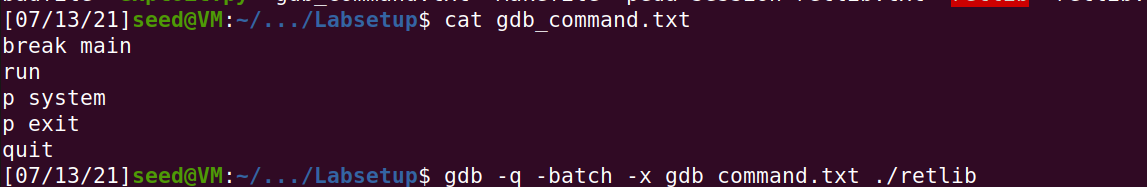
**Task1:Finding out the Addresses of libc Functions**

正常模式：

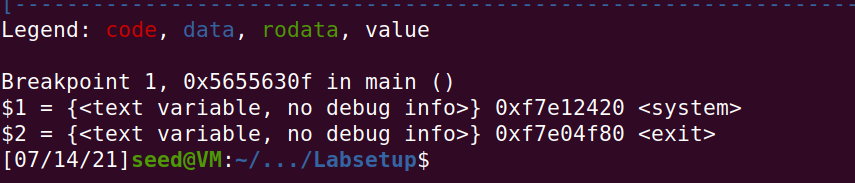
****

批处理模式：

文本内容

****

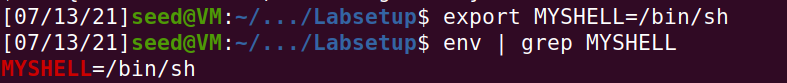
执行结果



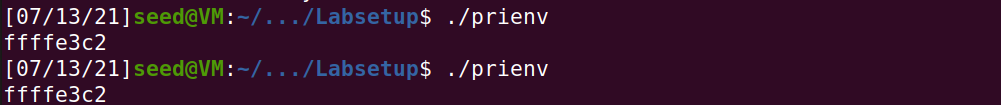
所以，system()的地址为0xf7e12420,exit()的地址为0xf7e04f80

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

插入环境变量MYSHELL

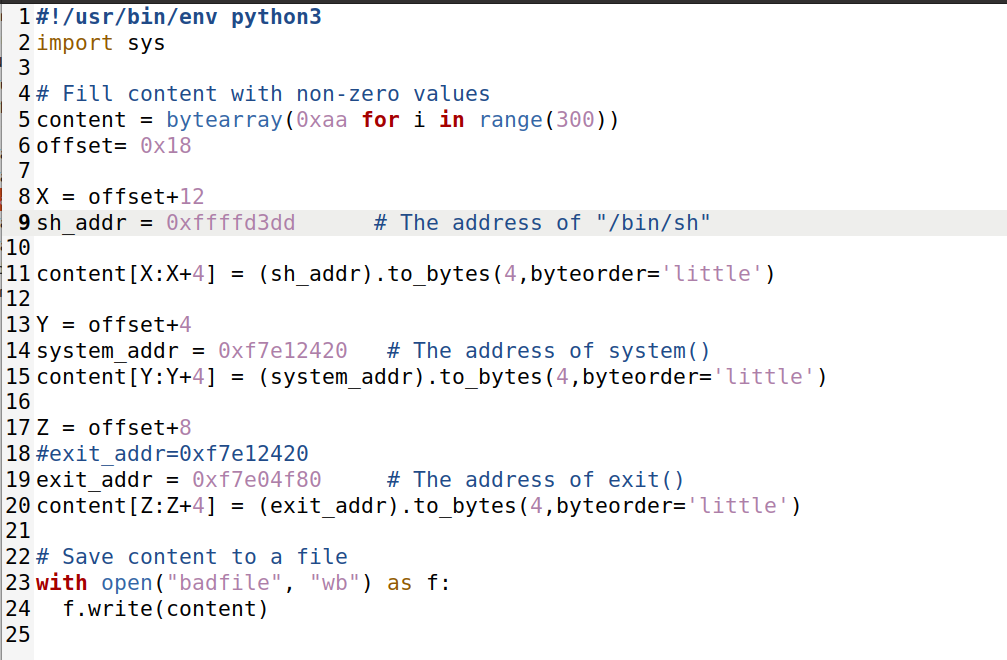
****

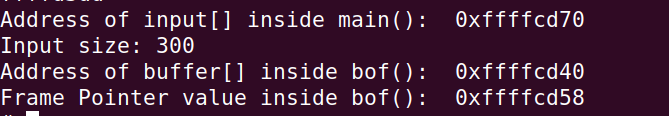
执行结果

****

由上图可以看出，因为关闭了地址随机化，所以执行两次prienv的输出相同。所以”/bin/sh”的地址为ffffe3c2。

**Task3：Launching the Attack**

攻击程序设计：  


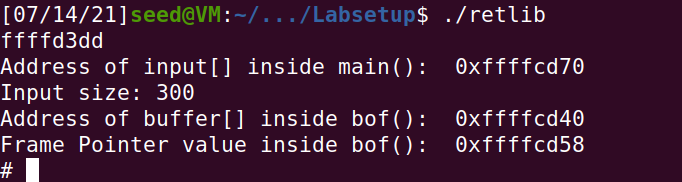
程序解释：  
首先执行retlib，得到如下输出：  


所以，offfset=0xffffcd58-0xffffcd40。

根据栈的进出规则，分别将X，Y，Z设置为offset+12,offset+4,offset+8，使程序能执行system(“/bin/sh”)，执行完成后又能够执行exit()。

将system\_addr和exit\_addr设置为task1中获取的值。

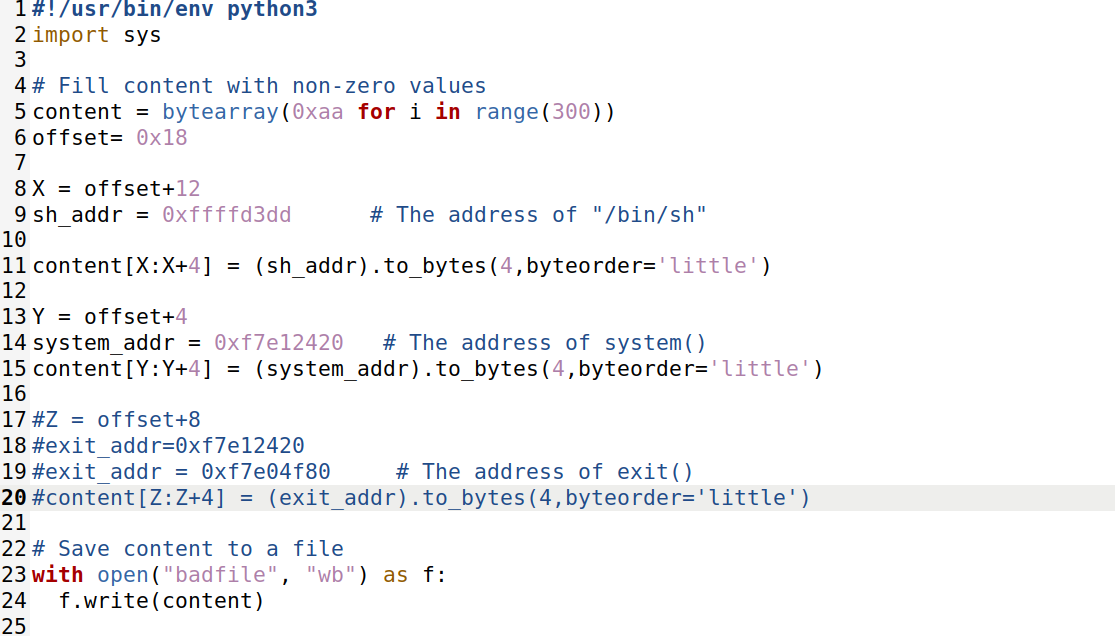
在实验过程中发现”/bin/sh”的真实值为0xffffd3dd，与task2中获取的值不同，所以将sh\_addr置为0xffffd3dd。

攻击结果：  


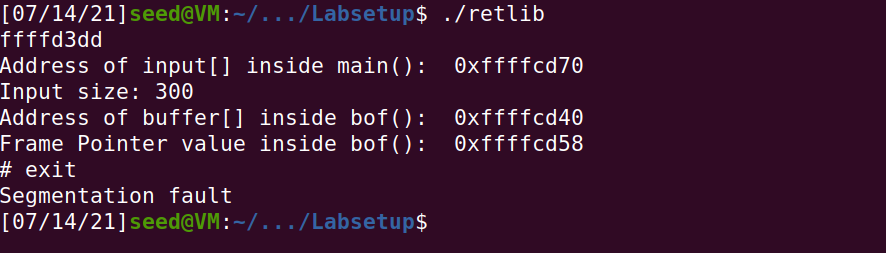
获得具有root权限的shell，攻击成功。

**Attack variation 1：**

修改后攻击程序：



程序解释：  
将17-20行注释掉，保证exit()的地址不会写入到badfile内。

攻击结果：  


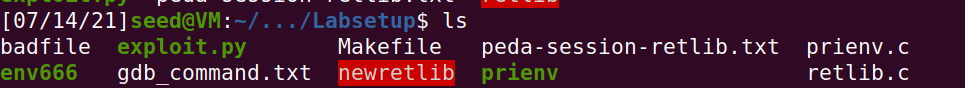
攻击依然成功，但在退出shell后，程序报错。

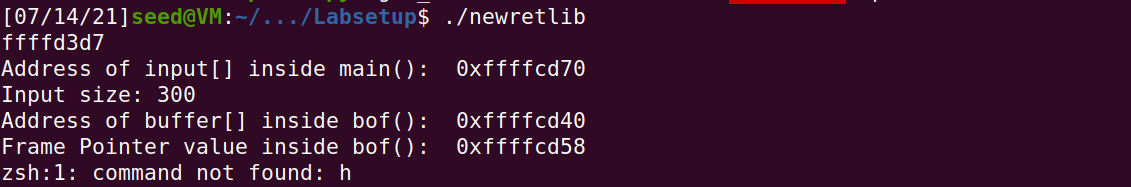
结果解释：

由于未插入exit()的地址，所以程序在执行完system()后，会从内存中未知的位置开始执行，使程序崩溃。

**Attack variation 2:**

将retlib改名为newretlib:



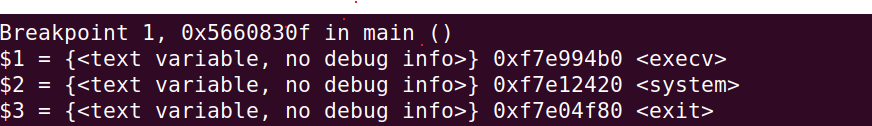
攻击结果：  


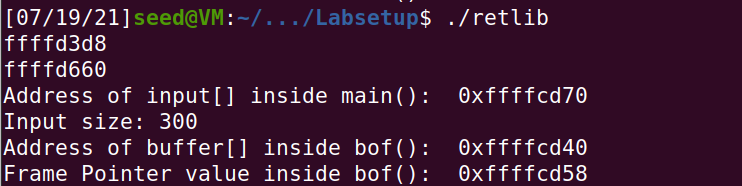
攻击失败。

结果解释：  
改名后”/bin/sh”的地址改变，所以无法传入system()的参数列表内，system()执行未知命令。

**Task4:**

首先获取execv的地址。



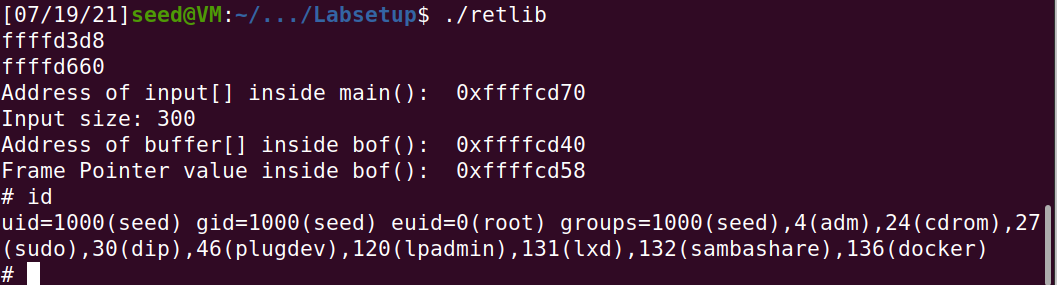
地址信息：  


攻击代码：



代码解释：  
offSET=0xffffcd70-0xffffcd40=0x30，为input到buffer的地址偏移。

Sh\_addr为/bin/sh的地址，A\_addr为-p的地址，C为空参数，因为execv的参数读取为从右到左，所以以上三个参数从高到低放置到栈内。

攻击结果：  


攻击成功。