**操作系统实验二实验报告**

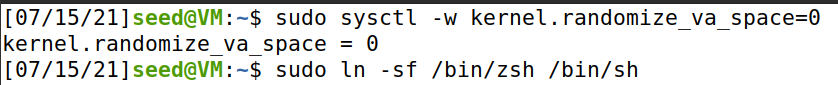
**基本信息：**

完成人姓名：顾琰 学号：57119117 完成日期：2021 年 7 月 15 日

**实验内容：**

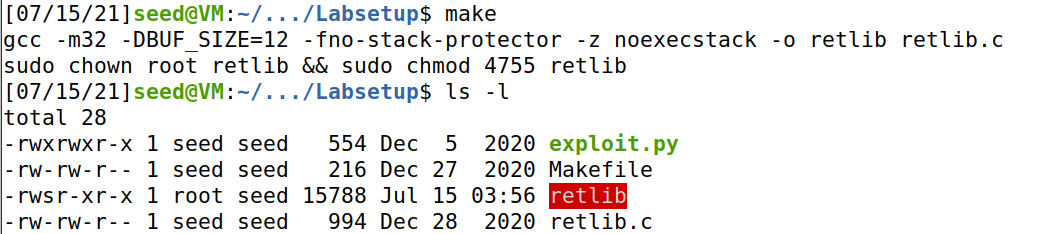
**环境配置：**

关闭ASLR 并将sh软链接到zsh：

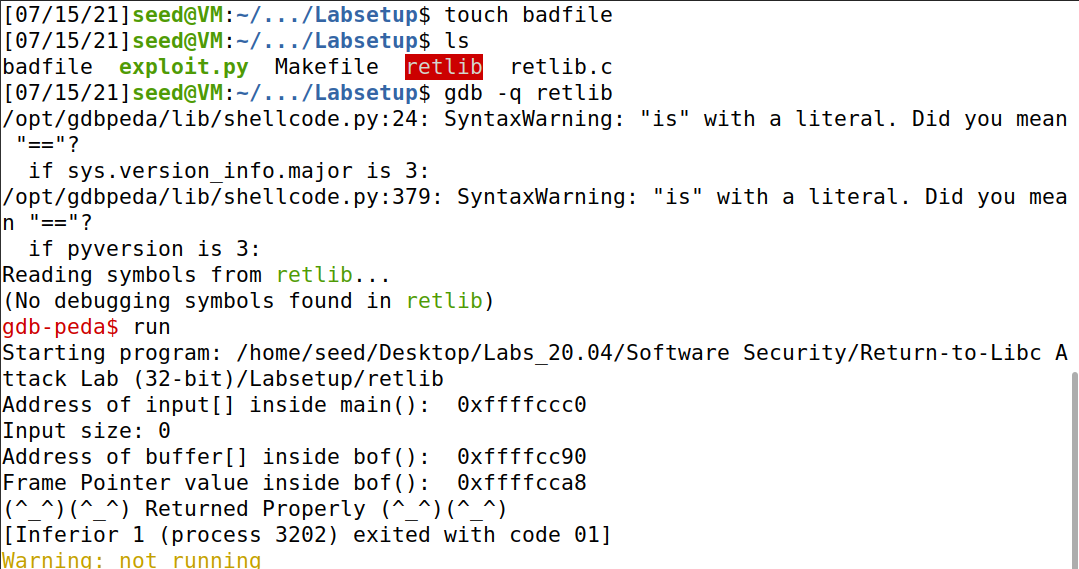


**Task 1: 找到system（）函数的地址**

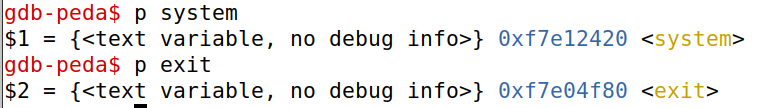
使用 makefile 编译具有漏洞的程序：



通过 gdb 调试获得 system 和 exit 函数地址：



得到system（）和exit（）的地址:



**Task 2: 将 shell字符串放入内存中**

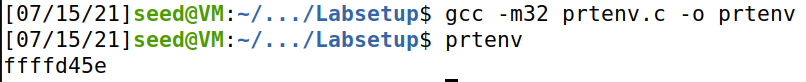
添加环境变量:



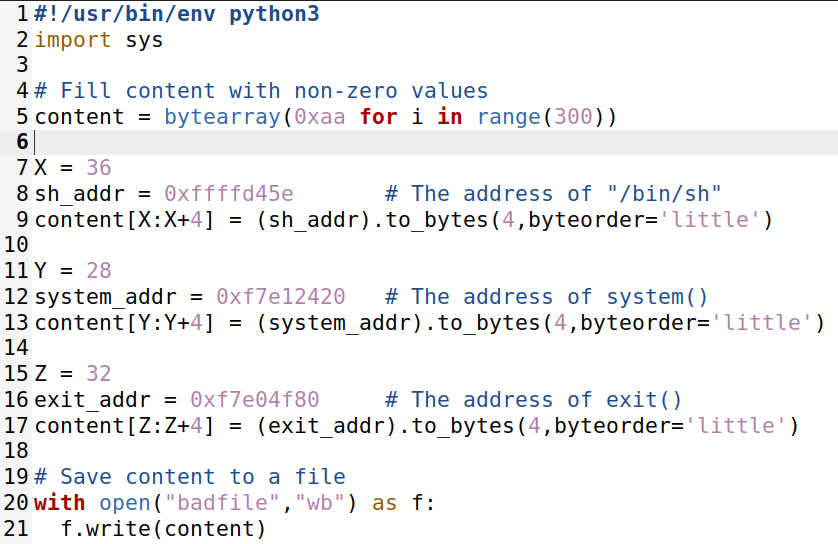
利用下述程序获得上述环境变量的地址：

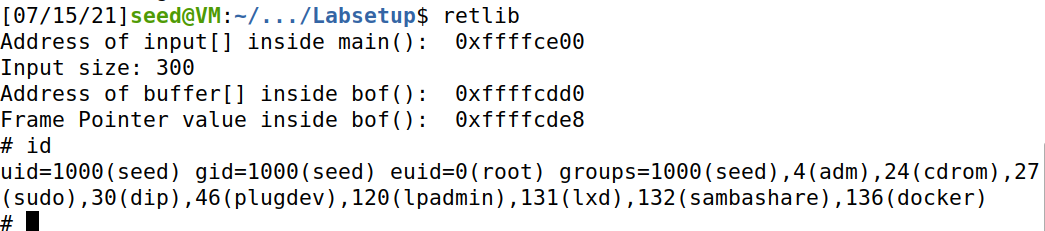


得到”/bin/sh”的地址:



**Task 3: 发起攻击**

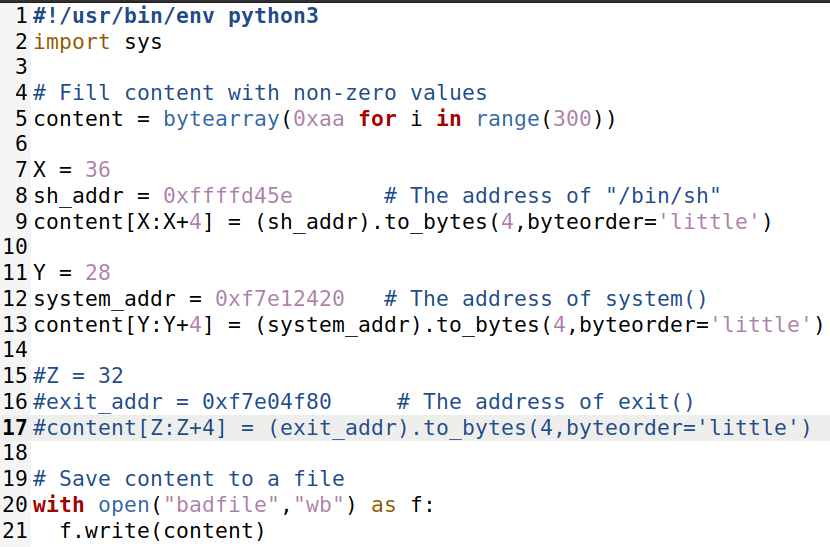




上面的 shell 用户符表示我们攻击成功，已经获得了 root 权限的 shell，攻击成功。

**Attack variation 1：**

去掉 exit 函数地址后运行攻击程序：

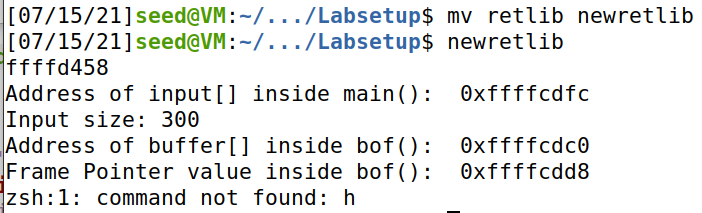


上面的 shell 用户符表示我们攻击成功，已经获得了 root 权限的 shell。

即去掉 exit 函数地址后，我们的攻击仍然有效！只是在后续退出时程序会崩溃报错。

**Attack variation 2：**

修改漏洞程序的名字使其与之前不相同时：



攻击失败，我们可以看到其打印出的关于/bin/sh 的我们之前设置的 MYSHELL 的地址

已经发生了变化，所以我们填入的攻击程序的地址现在是错误的了。

**Task 4: 击败 Shell 的对策**

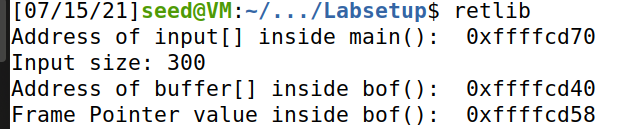
将/bin/sh 链接回/bin/dash：

IMG_256

找到 execv 函数的地址：

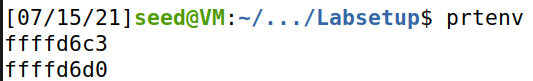
IMG_256

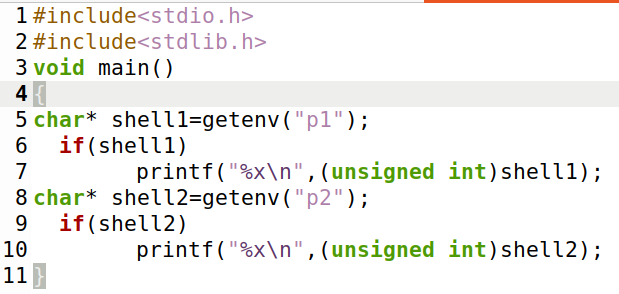
获取 main 函数内的输入的数据地址：

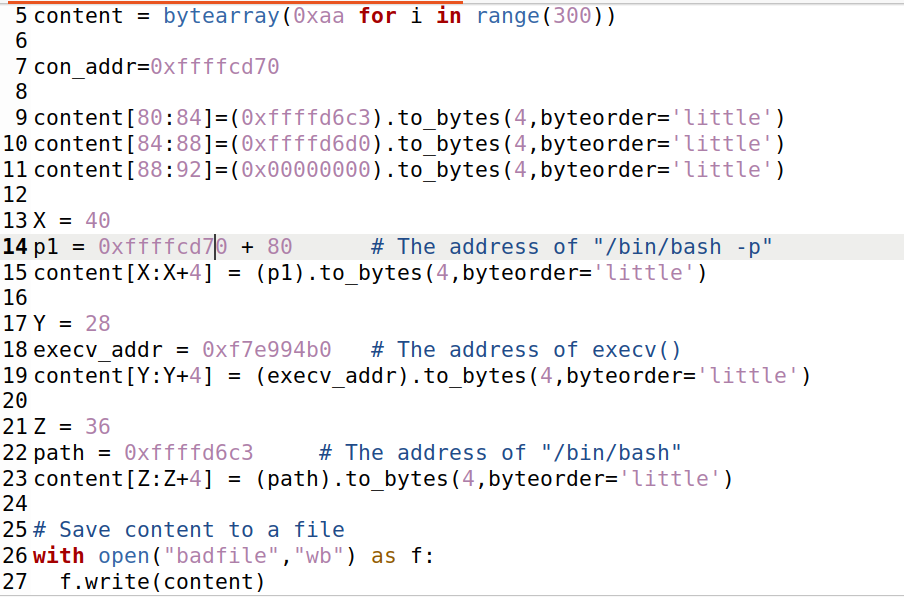


在攻击代码内通过环境变量的方式构造参数”/bin/bash”以及 char\*数组”/bin/bash”、”-p”：

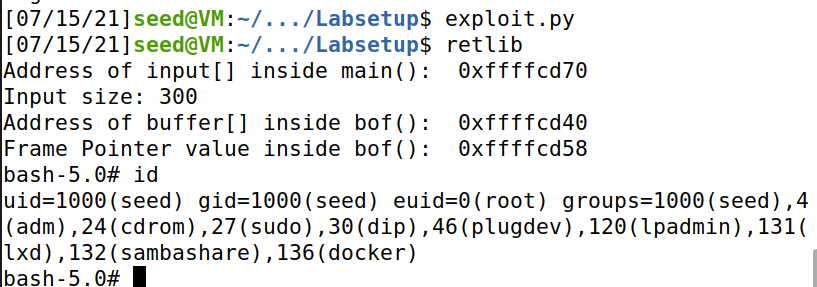
IMG_256







编译攻击：



攻击成功。

**实验体会：**

本次实验是我们的第三次实验，经过本次实验，我总结了如下的知识点:

① 通过 gdb 调试，我们可以获得内存中的 libc 库中的函数地址，这样我们可以结合缓冲区溢出攻击，将目标 setuid 程序引导到我们希望的一个 libc 库函数中(并以 root 权限运行)。

②在return-to-libc攻击中，通过改变返回地址，攻击者能够使目标程序跳转到已经被加载到内存中的libc库中的函数。