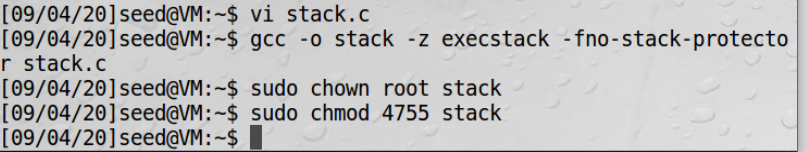
Lab 2实验报告 57118110 杨紫瑄

**Task 1**



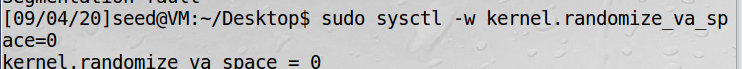
将代码编译保存在shellcode.c文件中并运行，启动shell



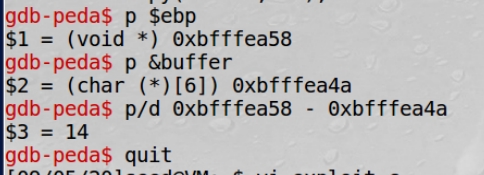
将程序的所有权更改为root并授予set-uid权限

**Task 2**

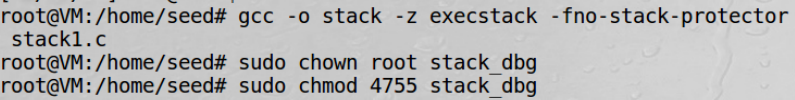
将BUF\_SIZE 改为6，以便更容易溢出



关闭地址随机化



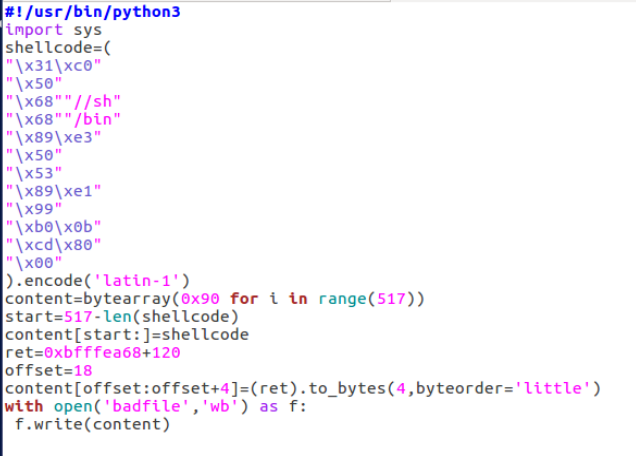
得到ebp和buffer的地址，求出两地址差，得到buffer和return address之间的距离检验是否为BUF\_SIZE+8



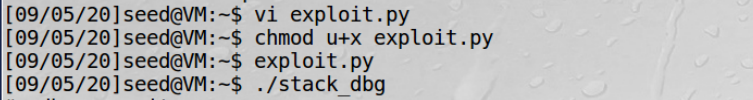
在root下编译stack1.c并提升set-uid权限

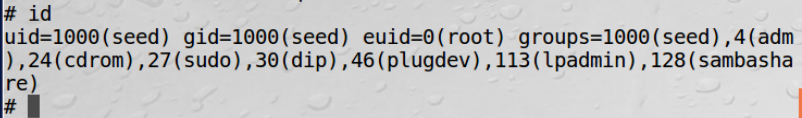


获得shellcode 的起始地址 0xbfffea68



上为exploit.py代码





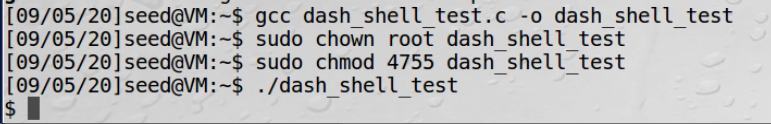
编译成功获得root权限

**Task 3**



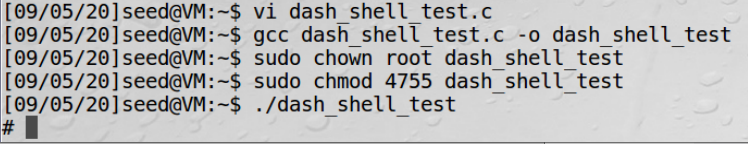
将/bin/sh符号链接指回/bin/dash

未取消注释时：



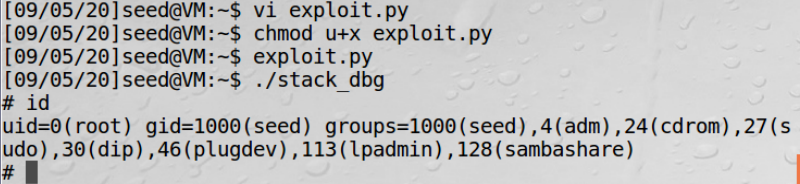
获取的是普通用户权限的shell

取消注释后：



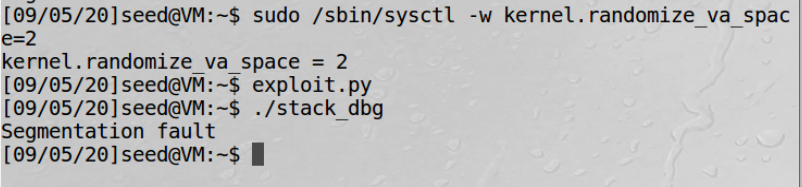
利用setuid(0),获取shell的root权限

在exploit.py中添加代码后再次进行Task 2操作：

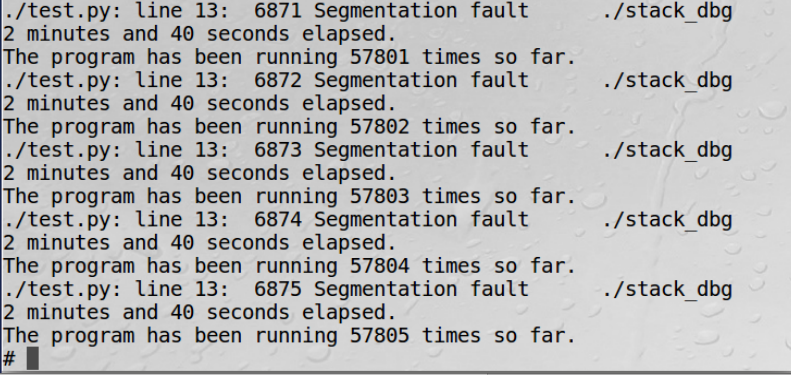


发现进入了特权模式，获得了root权限。因为在执行shell命令之前，特权用户仍然有root权限，setuid(0)设置真实用户为0，绕过dash的防御措施。

**Task 4**

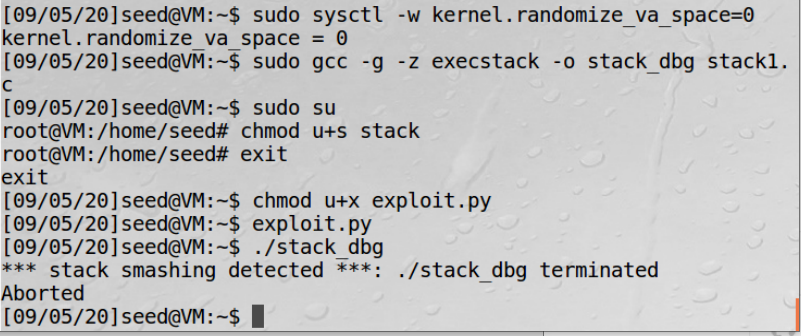


开启地址随机化，再次编译Task 2，发现出现段错误



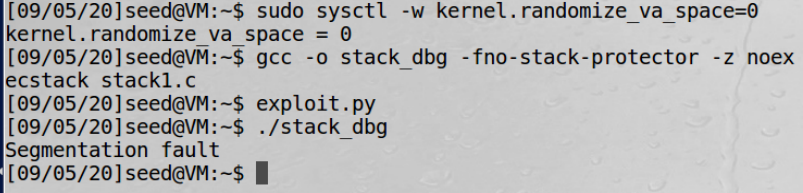
地址随机化后，编写测试代码，跑了2分钟40秒，执行了5万七千次左右命中地址，获得特权

**Task 5**



关闭地址随机化后，再次编译Task 2并启用StackGuard保护机制，发现不会进入特权模式，并出现了缓存溢出

**Task 6**



关闭地址随机化后，再次编译Task 2无法得到一个shell，显示段错误

实验总结：此次试验我在Task2中遇到了一些问题，buffer一直显示return properly，没有实验溢出，之后我将BUFFER\_SIZE改小后问题得以解决。通过此次试验我对缓冲区溢出漏洞的相关知识有了更深的了解，也对linux各代码的更加熟悉。