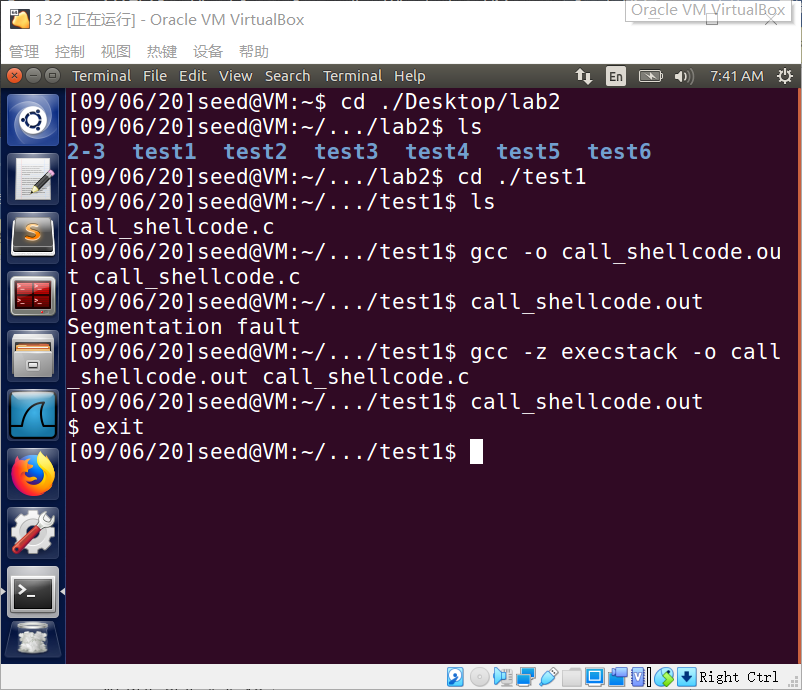
**Lab2实验报告**

57118213 陈洪杰

**Task 1: Running Shellcode**

将所给代码保存为call\_shellcode.c，用一般方式编译后，输出会显示Segmentation fault，用如图方式编译后，能够成功调用shell。



**Task 2: Exploiting the Vulnerability**

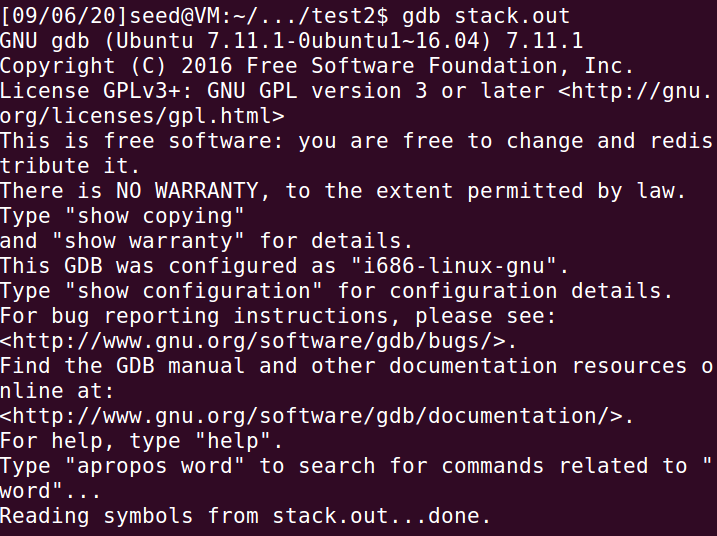
将2.3中的代码保存为stack.c，进行编译并且将其改为root权限的set-uid程序



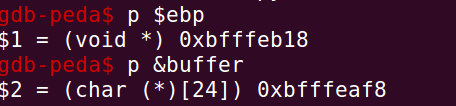




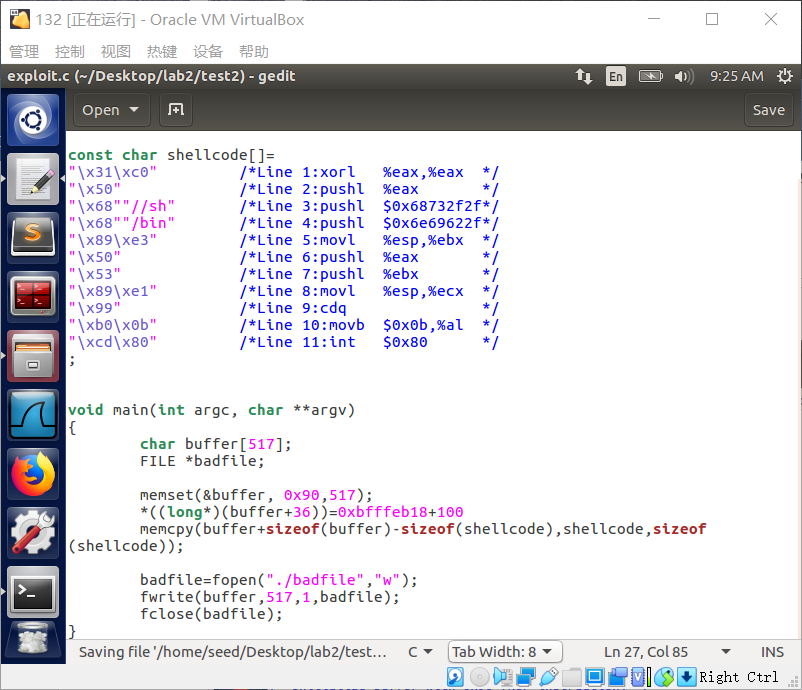
对stack.out文件gdb



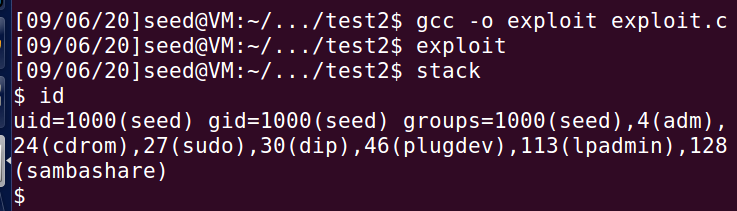
找到ebp的值是0xbfffeb18，buffer的为0xbfffeaf8，两者相减加4就能得到buffer与bof返回地址的距离



然后修改exploit.c，



最后编译运行exploit和stack，得到root的id

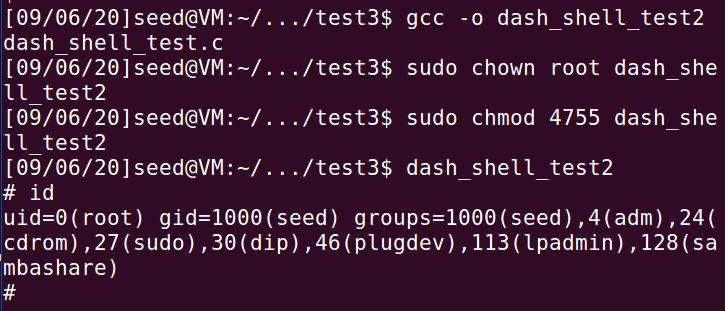


**Task 3: Defeating dash’s Countermeasure**

将所给代码编译运行，查看id



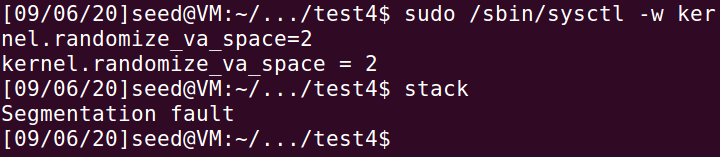
发现得到的是普通用户的id，接着取消掉注释，再运行；



发现进入了root用户的shell。

**Task 4: Defeating Address Randomization**

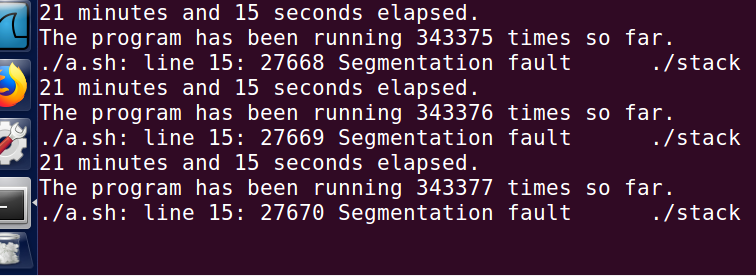
关闭地址随机化后用task2的方式攻击，发现攻击失败



编写所给的代码，存为a.sh

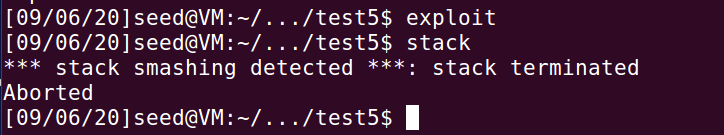


发现在运行了21min15sec后仍然为攻击成功，可能是随机性太大。



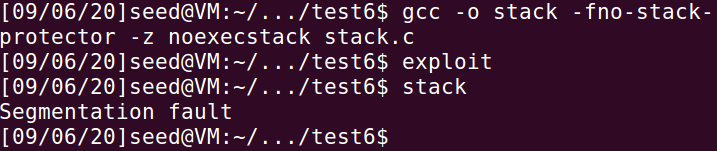
**Task 5: Turn on the StackGuard Protection**

将kernel.randomize\_va\_space再次置0，用下面的命令重新编译stack，再次运行task2中的程序，检测到栈溢出：



**Task 6: Turn on the Non-executable Stack Protection**

用noexecstack重新编译stack.c，运行后出现错误segmentation fault，



查看错误原因，发现在0xbfffeb7c出现问题，这是正在nop上

