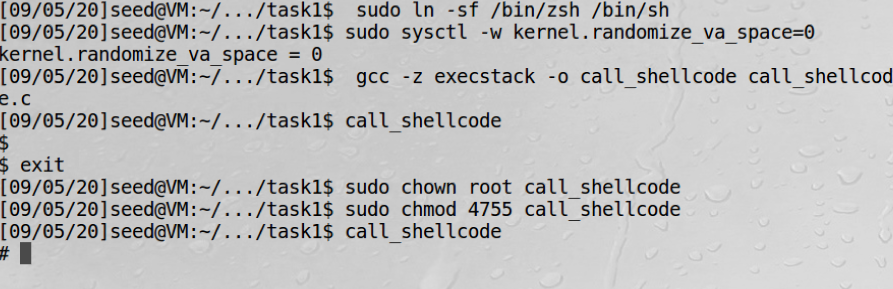
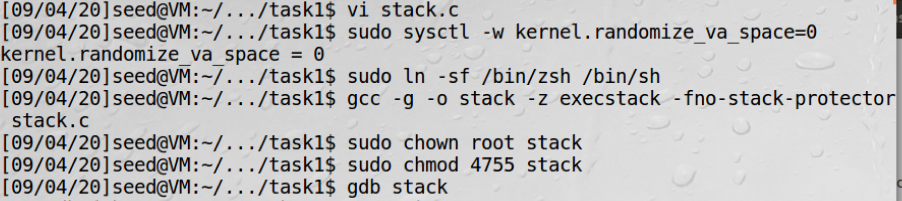
Lab2：

一、task1:runing shellcode

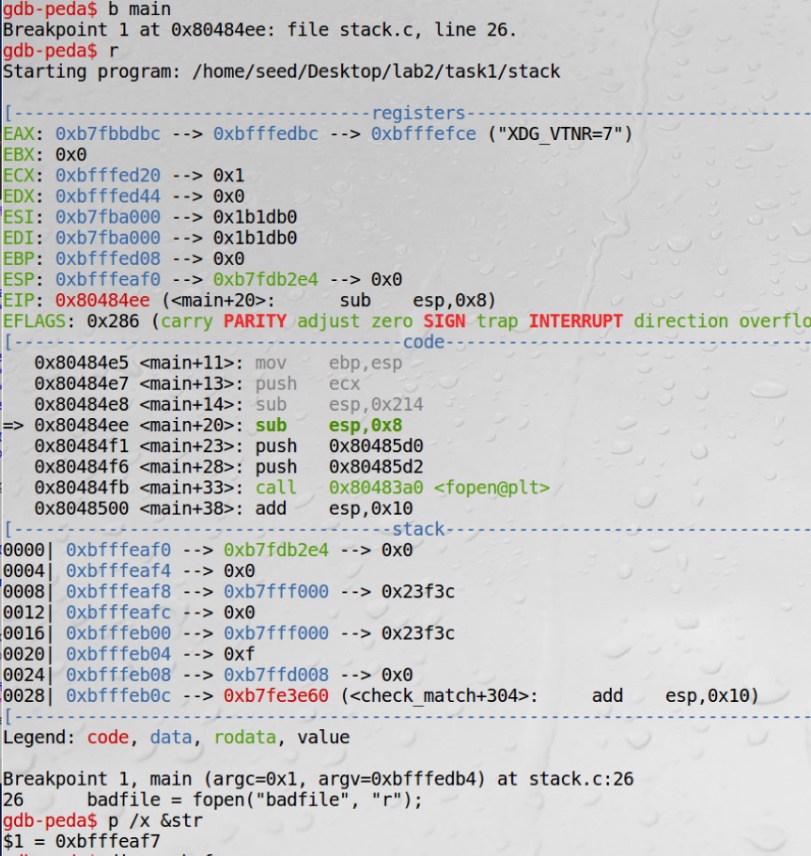


二、task2：exploiting the vulnerability

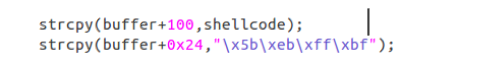
1、按照实验指导步骤对stack进行编译



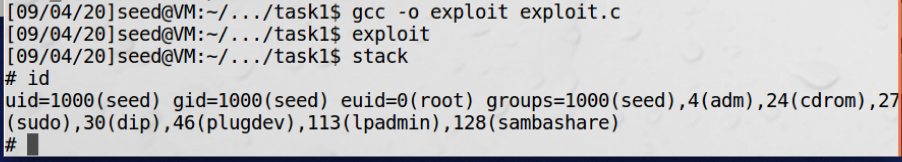
2、用gdb调试找出str地址为0xfffeaf7，则shellcode地址为0xfffeaf7加上偏移量0x64，即0xfffeb5b



3、向exploit.c中添加代码。



4、编译并运行exploit，生成badfile，利用stack的漏洞进行攻击，攻击成功。

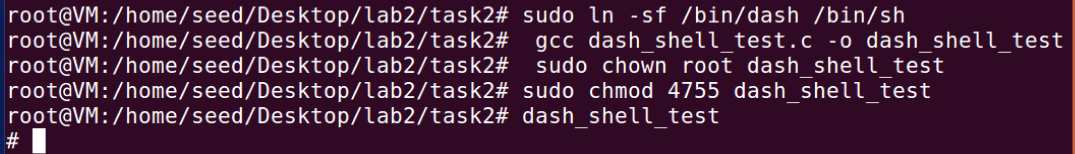


二、task3：Defeating dash’s Countermeasure

Setuid（0）被注释后，sudo ln -sf /bin/dash /bin/sh之后运行，不能获得root权限

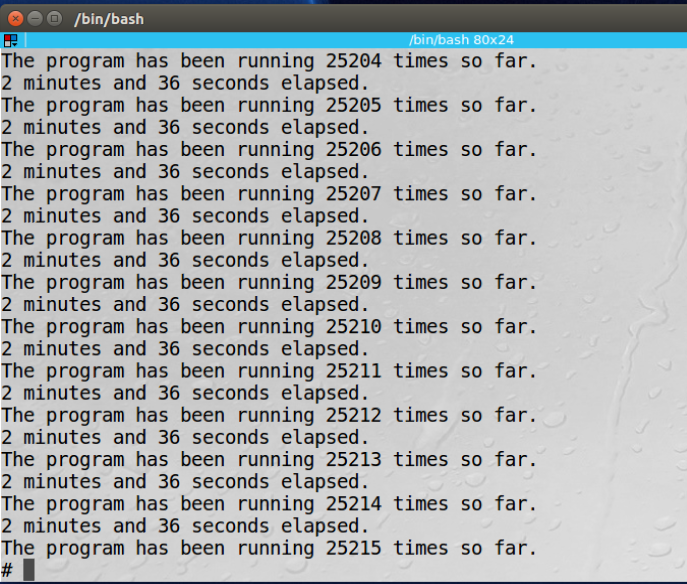


Setuid（0）被取消注释后，运行能够获得root权限

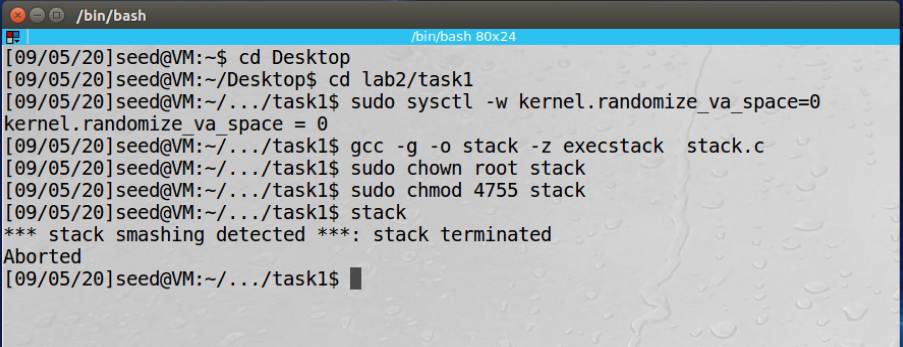


三、task4：Defeating Address Randomization

将禁止随机化关闭后，运行脚本，25215次尝试后成功获得root权限。

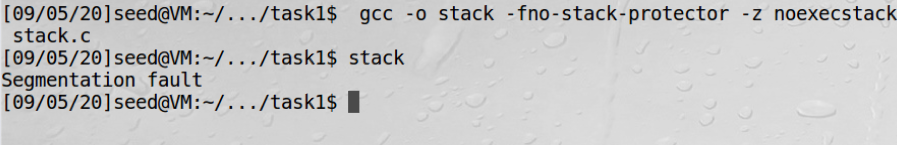


四、task5：Turn on the StackGuard Protection



报出stack smashing detected的错误

五、task6：Turn on the Non-executable Stack Protection

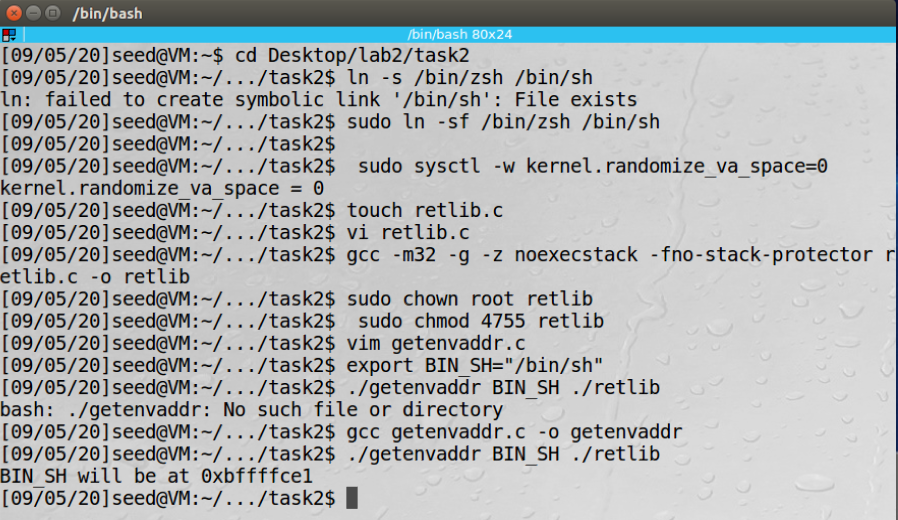


报出segmentation fault

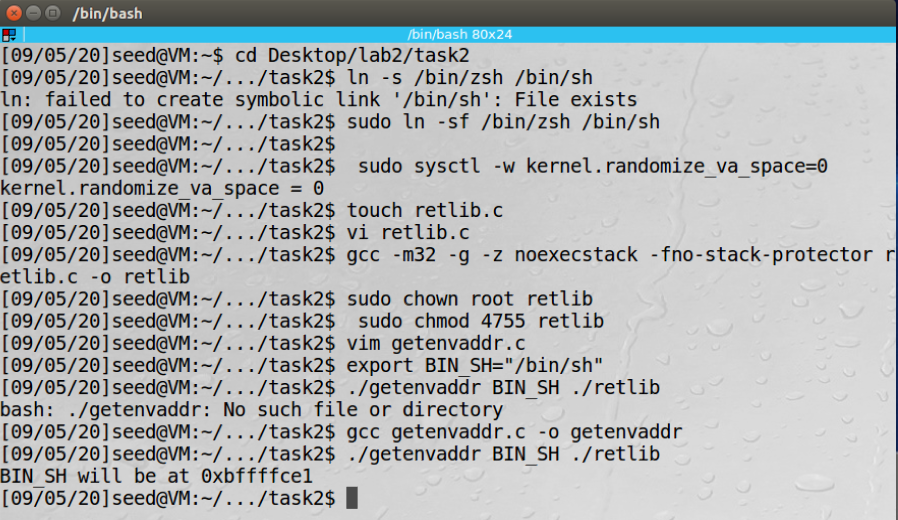
lab3

一、task1、2、3：Exploiting the buffer-overflow vulnerability

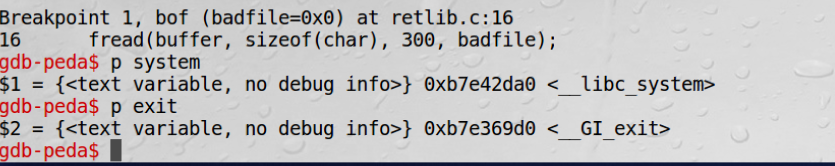
1、按照实验手册要求创建并编译retlib



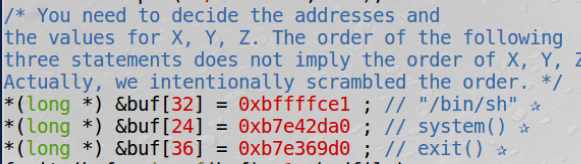
2、获取bin/sh地址



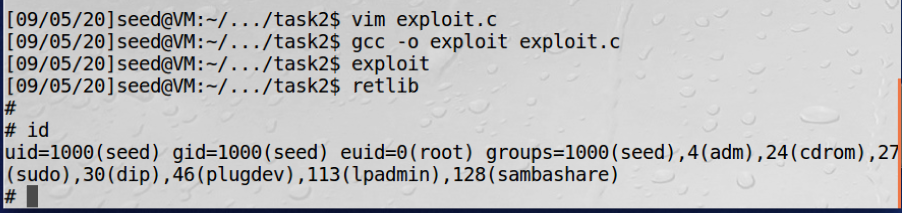
3、获取system和exit地址



4、将地址添加进入exploit.c



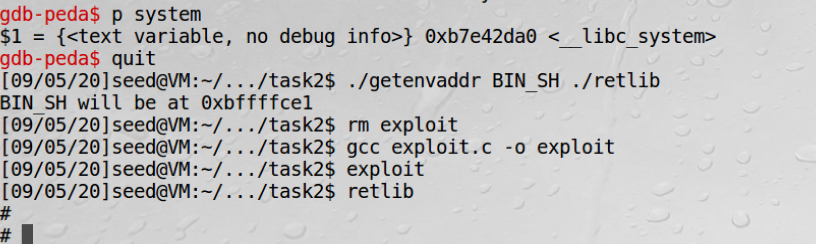
5、编译并运行exploit，生成badfile，利用retlib的漏洞进行攻击，获取root权限，攻击成功。



二、Attack variation

1、Attack variation 1: Is the exit() function really necessary? Please try your attack without including the address of this function in badfile. Run your attack again, report and explain your observations.

去掉exit地址后，攻击仍然成功，获得root权限

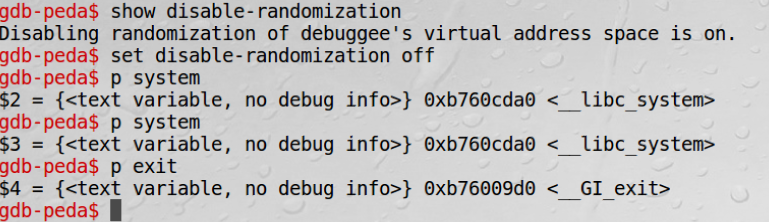


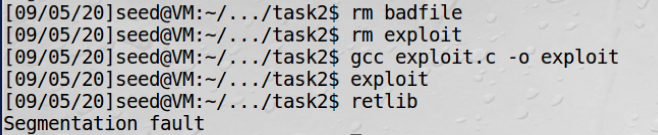
2、Attack variation 2: After your attack is successful, change the file name of retlib to a different name, making sure that the length of the new file name is different. For example, you can change it to newretlib. Repeat the attack (without changing the content of badfile). Will your attack succeed or not? If it does not succeed, explain why.

改变文件名称后，攻击不能成功。

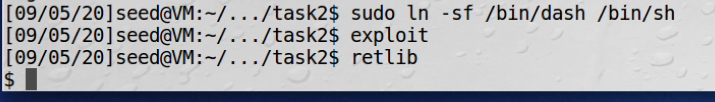
二、task4：Turning on address randomization

关闭禁止随机化后，可以看到地址已经改变





三、task5：Defeat Shell’s countermeasure



可以看到sudo ln -sf /bin/dash /bin/sh后不能获取root权限了

为此需要加入dash\_shell\_test.c的代码，利用setuid（0），可以看到重新获得root权限

