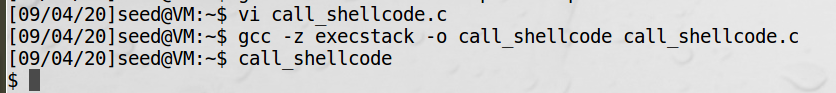
**Buffer Overflflow Vulnerability Lab实验报告**

**57117207 高晓悦**

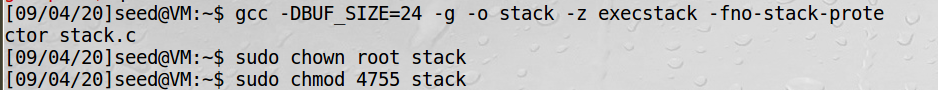
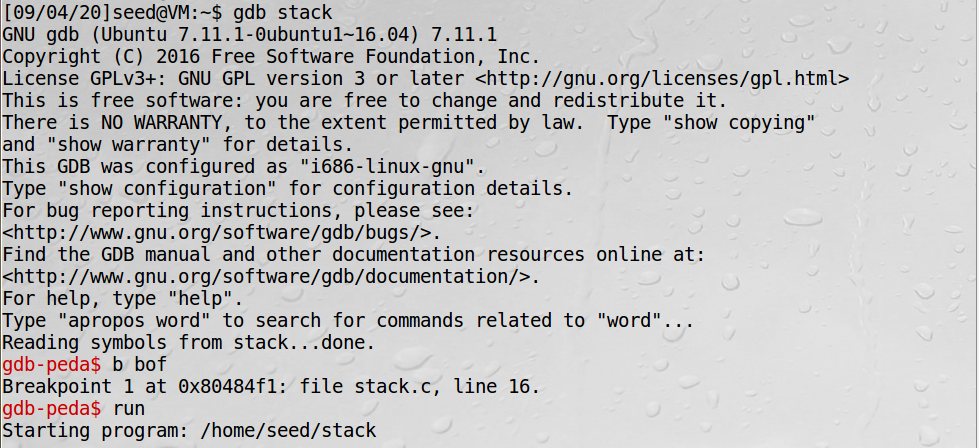
**Task 1: Running Shellcode**

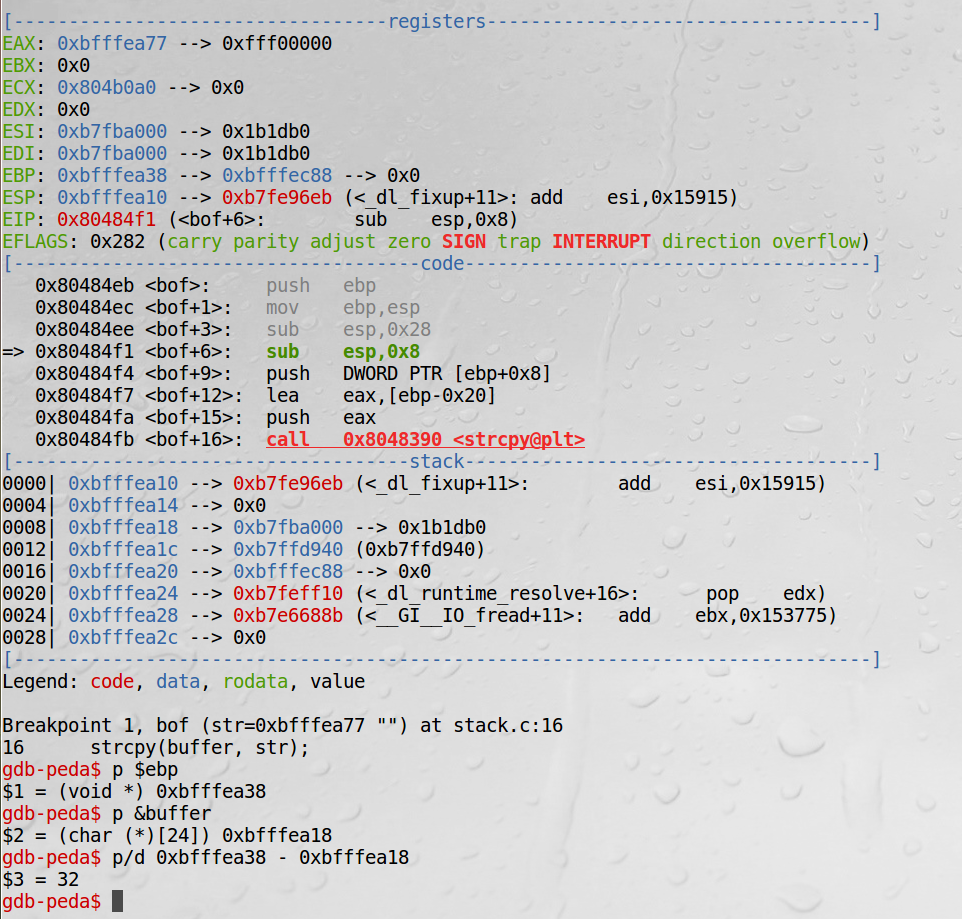
编译并运行call\_shellcode.c文件，结果成功调用shell：



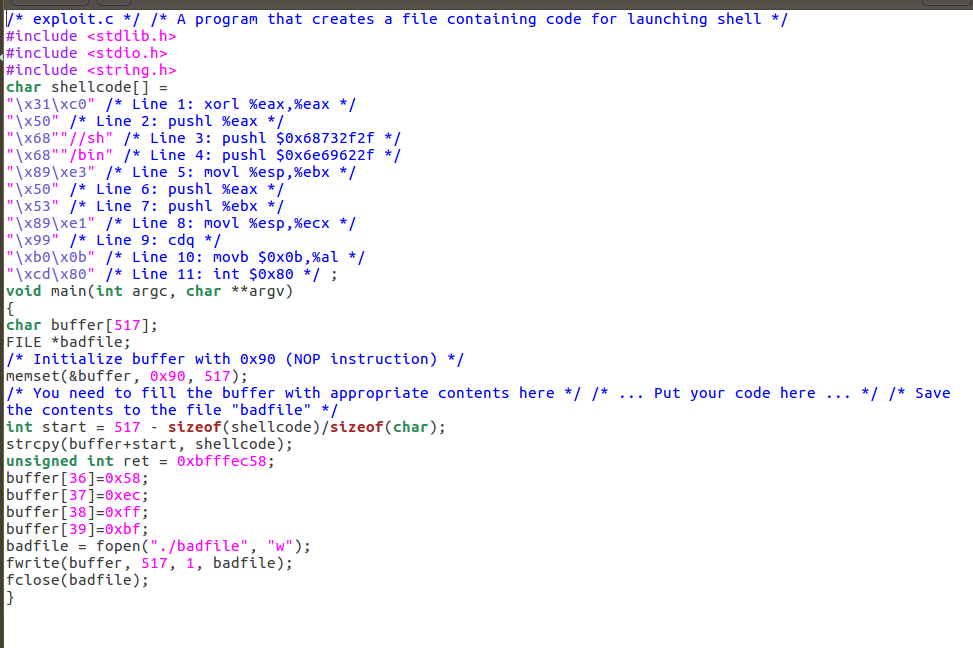
**Task 2: Exploiting the Vulnerability**

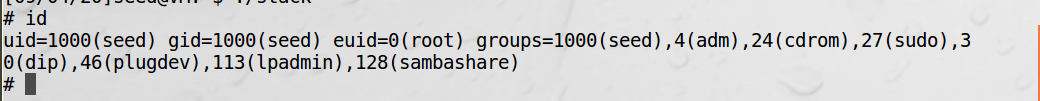
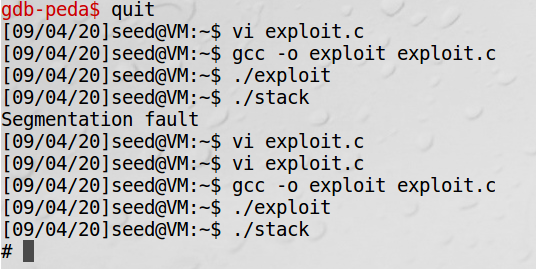
1、BUF\_SIZE设为24，编译stack.c文件并设为set-UID程序：

2、用GDB调试satck，对bof函数设断点并运行，

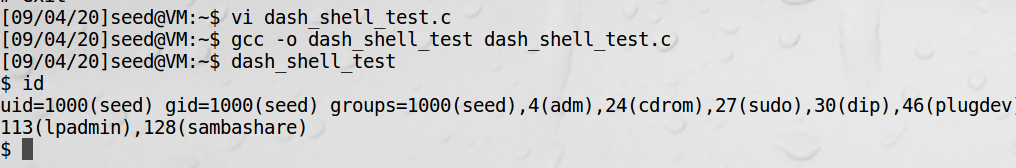
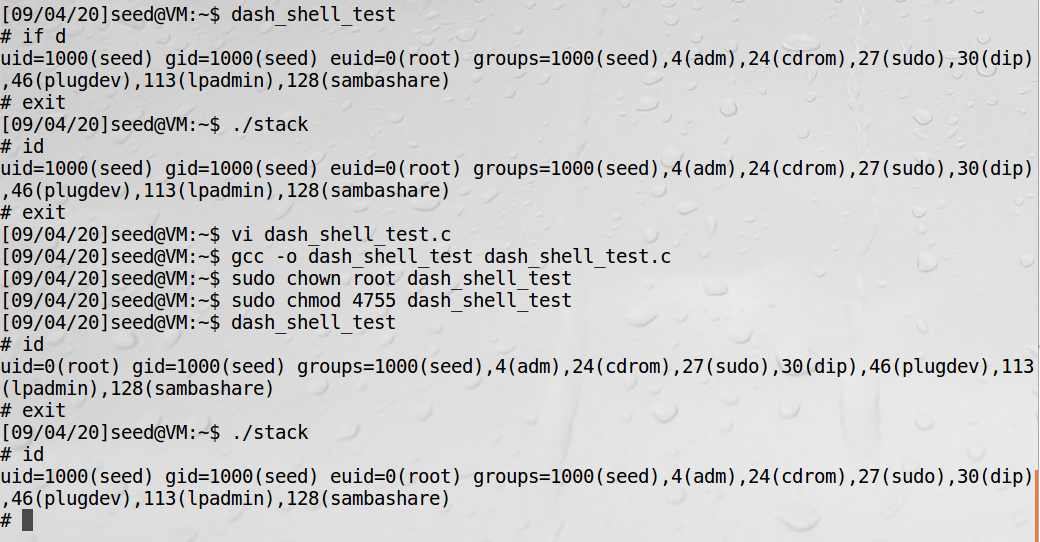
1. 查看寄存器ebp和buffer数组的地址，计算出buffer和bof返回地址间距离是36：

4、更改exploit.c文件：



5、编译后运行stack，成功得到shell，获取root用户权限：

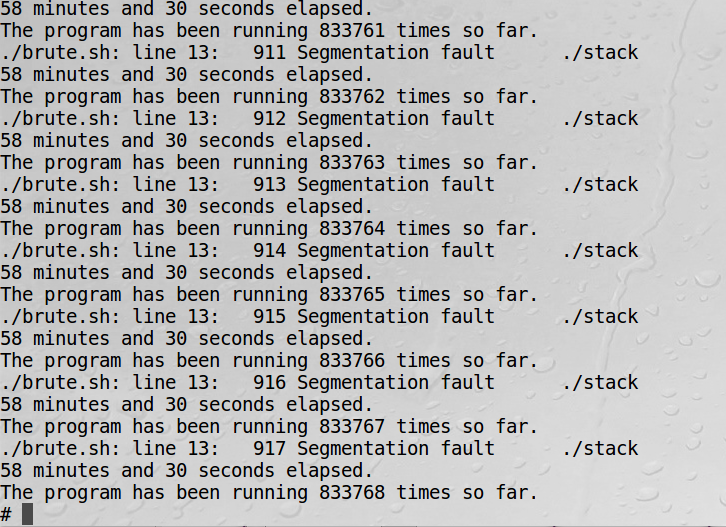
**Task 3: Defeating dash’s Countermeasure**

1. 注释掉setuid(0)，运行stack，获取到seed普通用户的shell：
2. 取消注释，运行stack，获取到root用户的shell：

把setuid(0)加到shellcode中，在dash下运行stack也可以得到root用户的shell，因为使用setuid(0)将ruid设为了root，因此便和euid一致，dash就不会降低权限。

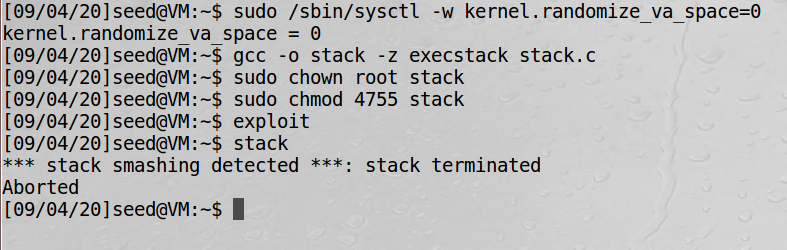
**Task 4: Defeating Address Randomization**

运行实验中提供的脚本文件，得到shell，等待时间较长：



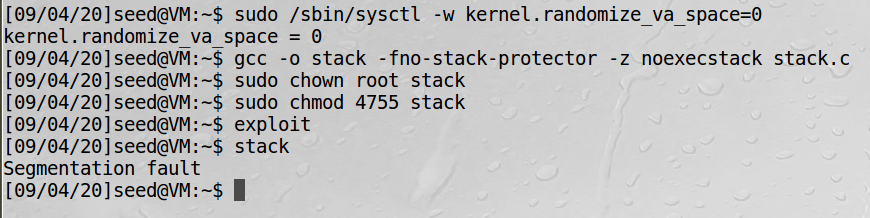
**Task 5: Turn on the StackGuard Protection**

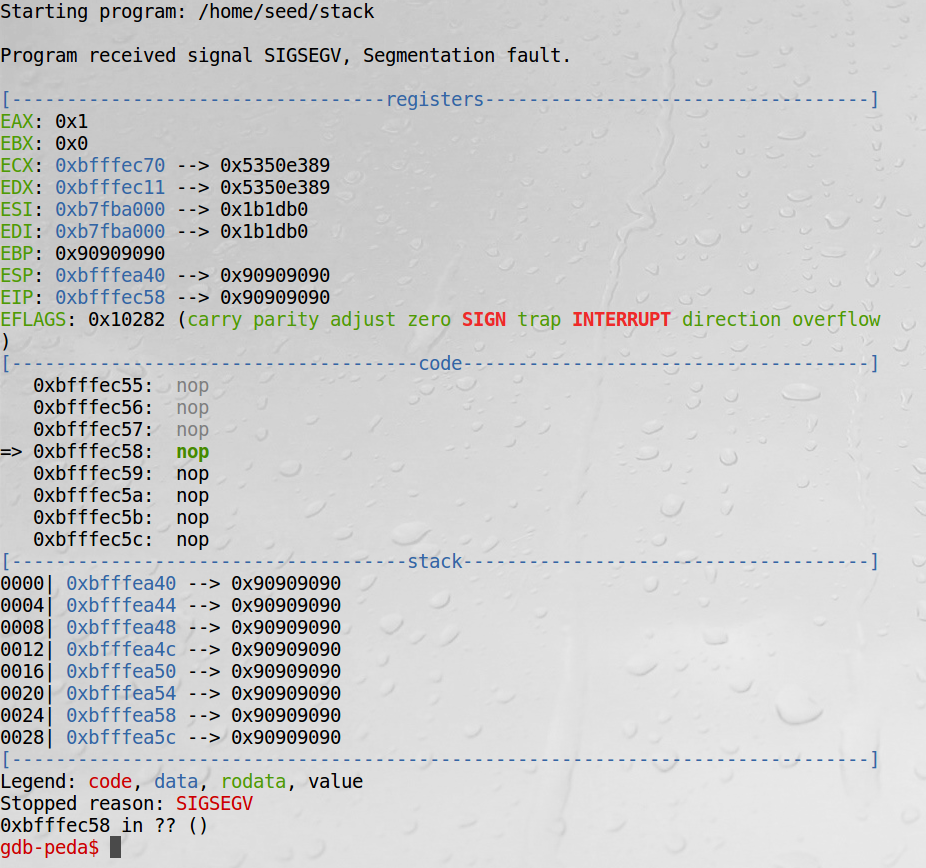
先关闭地址随机化，再禁用堆栈保护的运行stack，出现报错，因为有栈溢出攻击：



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

1、关闭地址随机化，运行stack报错：



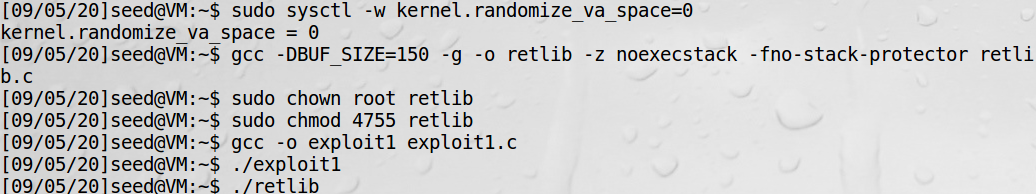
1. 用GDB进行调试，错误发生在0xbfffec58，shellcode的第一个指令，只要指令地址在栈上，保护机制就会起作用，攻击程序就会报错：

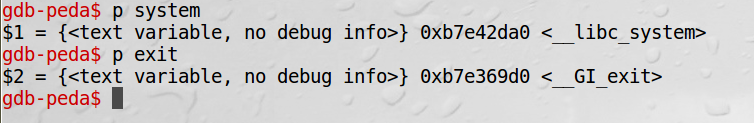
**Return-to-libc Attack Lab实验报告**

**57117207 高晓悦**

**Task 1: Finding out the addresses of libc functions**

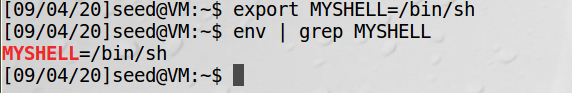
1、先关闭地址随机化，BUF\_SIZE设为150，编译并设为set-UID程序：

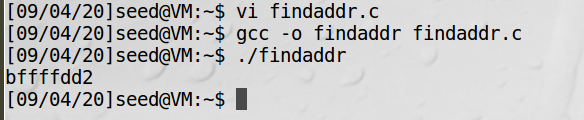


2、用GDB调试，查看system和exit的地址：

**Task 2: Putting the shell string in the memory**

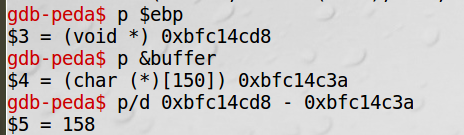
1、设置环境变量MYSHELL=/bin/sh：



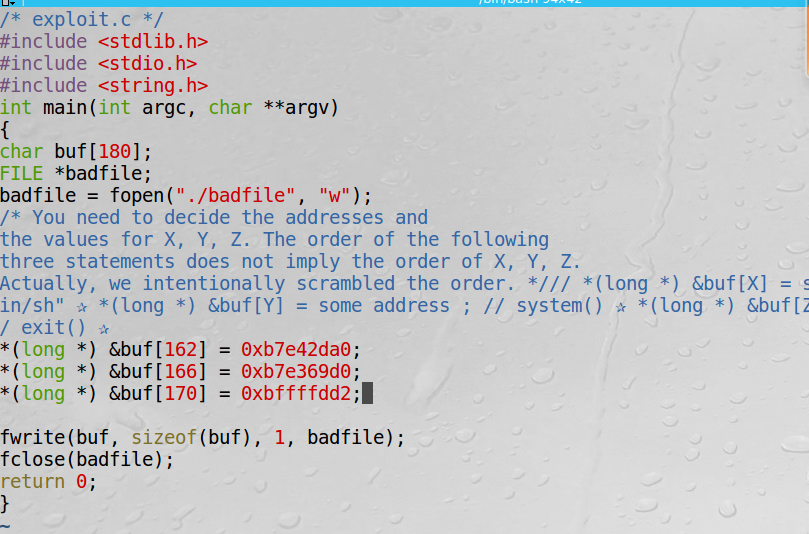
2、编译实验中给的程序代码并执行，得到/bin/sh的地址：

**Task 3: Exploiting the buffer-overflflow vulnerability**

1. 查看寄存器ebp和buffer数组的地址，计算出buffer和bof返回地址间距离是162：

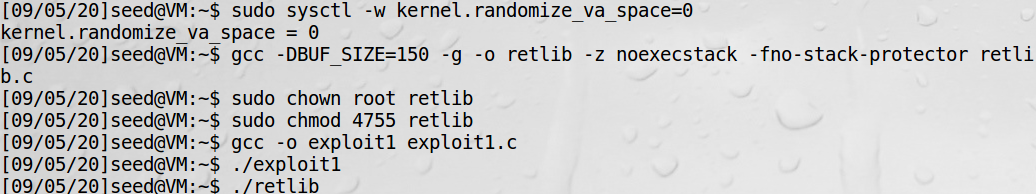


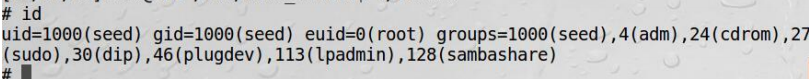
1. 更改exploit1.c文件：



其中，162-165字节为前面获得的system地址，166-169字节为前面得到的exit的地址，最后是/bin/sh的地址。

1. 编译执行exploit1.c后再运行retlib，得到root的shell：





（1）攻击1：

删除exit地址，再运行retlib，仍然可以提权，但是退出时会报错：



因为退出shell后程序会进入system的返回地址，当exit地址存在时能正常退出，删除exit地址后程序会跳转到别的地址导致出现错误。

1. 攻击2：

更改retlib的名字，会报错：



因为名字长度改变后，环境变量地址也会改变，传给system()的参数就发生了错误。

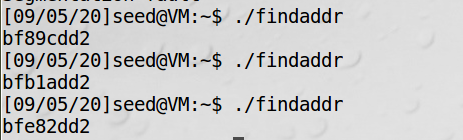
**Task 4: Turning on address randomization**

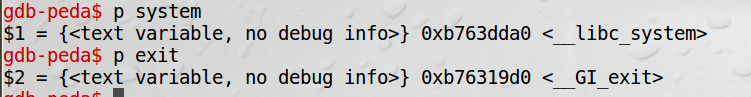
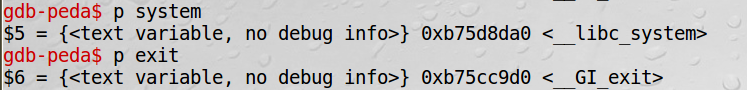
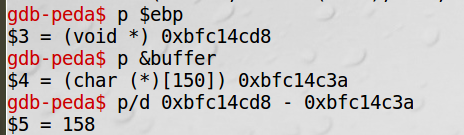
1、打开地址随机化，再次实施攻击，会报错：



因为打开地址随机化之后/bin/sh的地址不再固定。

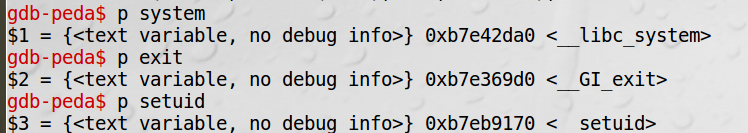
2、多次查看/bin/sh的地址，发现/bin/sh的地址确实不再固定，一直在变化：



1. 用GDB调试，set disable-randomization off，查看system和exit地址，发现system和exit地址也会发生改变：
2. 再次查看寄存器ebp和buffer数组的地址，发现ebp和buffer间的距离固定，始终为162：

**Task 5: Defeat Shell’s countermeasure**

1、查看system、exit和setuid的地址：



2、更改exploit1.c文件，可以先调用setuid(0)，然后再调用system()：



3、查看文件，发现通过这种方式retlib成功提权：

