**网络安全综合课程设计实验2**

57117204 姚晓

2020/9/5

**Buffer Overﬂow Vulnerability Lab：**

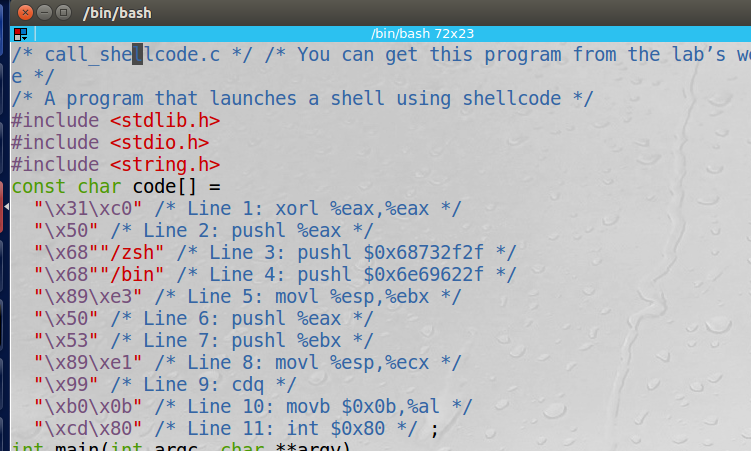
首先关闭地址空间随机化：



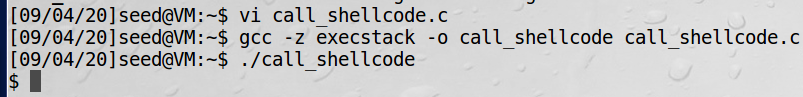
不使用stackGuard，使用可执行堆栈。

**Task1:运行shellcode**

编译并运行call\_shellcode.c代码：



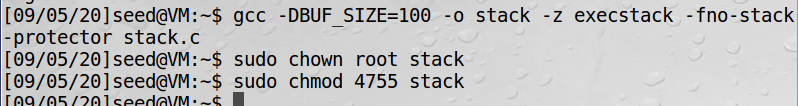
运行后发现调用了shell：



有漏洞的程序，Buffer size设置为了100：

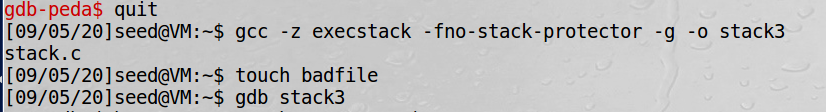


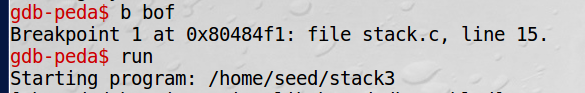
编译并将该程序设为root所有的set-UID程序：

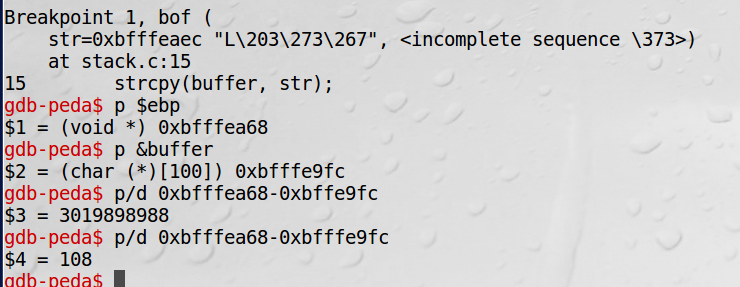


**Task2:利用漏洞**

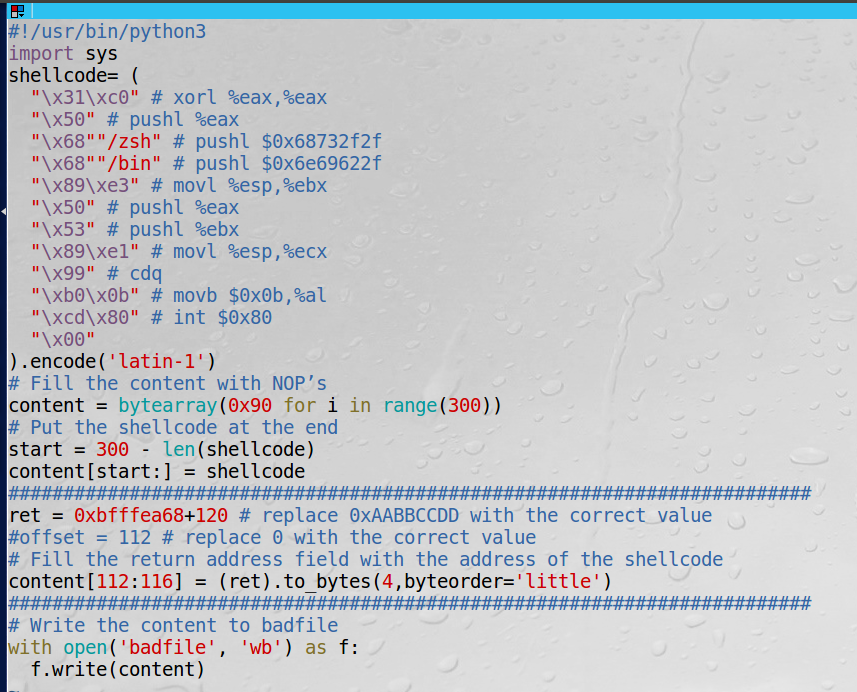
首先使用gdb找到当前指针所指地址以及返回值地址，算出offset为112：



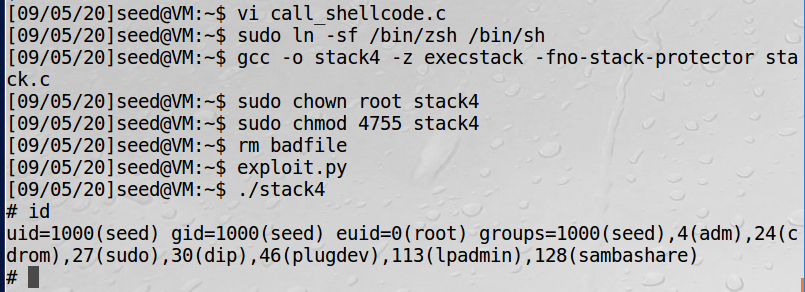




根据上面得到的地址值修改exploit.py：



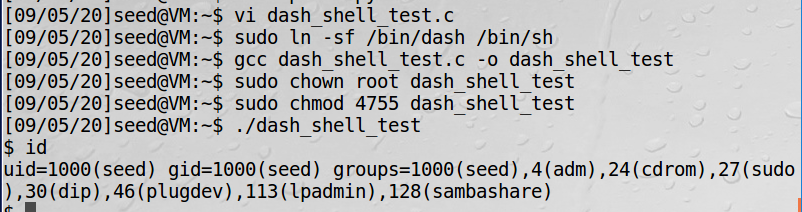
编译exploit.py并运行生成badfile文件，在运行stack可执行程序，结果如下，拿到了root权限：



**Task3：克服dash的对策**

Ubuntu 16.04中的dash shell在检测到有效UID与实际UID不相等时会放弃特权。

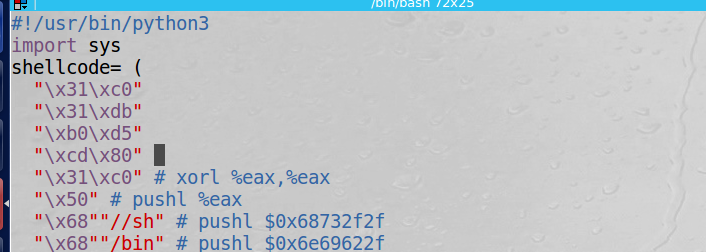
编译实验手册上的代码dash\_shell\_test.c，注释掉setuid(0)，然后把该执行文件改成root所有的Set-UID程序，执行该程序，发现不能获得root权限：

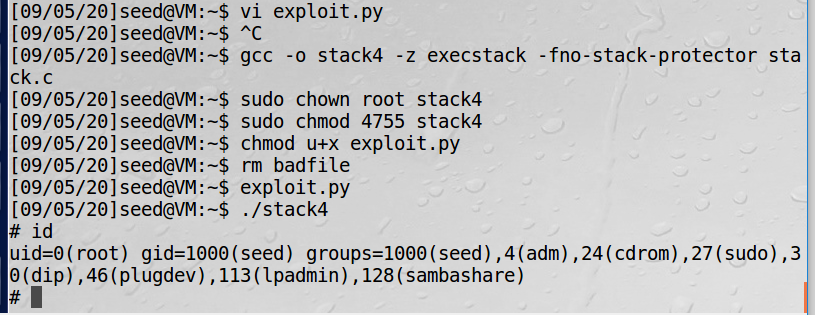


不注释掉setuid(0)，编译改过的的代码dash\_shell\_test.c，然后把该执行文件改成root所有的Set-UID程序，执行该程序，发现可以获得root权限：



利用实验手册的shellcode修改exploit.py，再次运行task2的攻击，可以获得root shell。

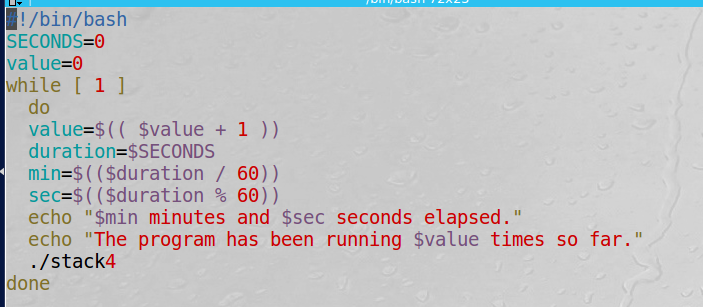




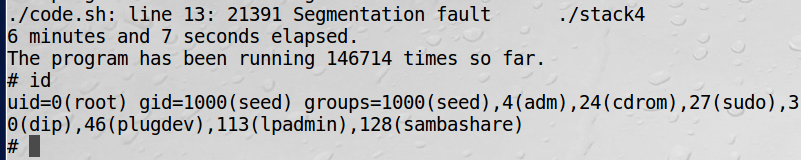
**Task4:击败地址随机化**

首先打开地址随机化，然后新建一个shell脚本code.sh，里面是实验手册的代码，运行该shell脚本，在无限循环中运行易受攻击的程序。



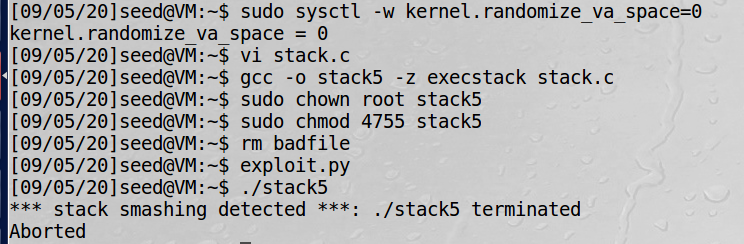


攻击成功，脚本停止，获得了root权限：



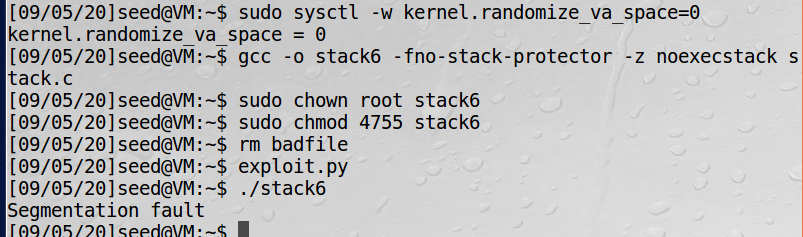
**Task5：StackGuard保护**

首先关闭地址随机化，然后使用StackGuard保护编译stack.c，再次执行任务2，发现不能获得root权限。



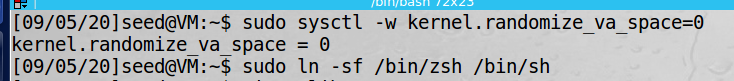
**Task6：启用不可执行的堆栈保护**

首先关闭地址随机化，然后使用noexecstack，不可执行堆栈来编译stack.c，重新执行任务2，发现不可以获得root权限：

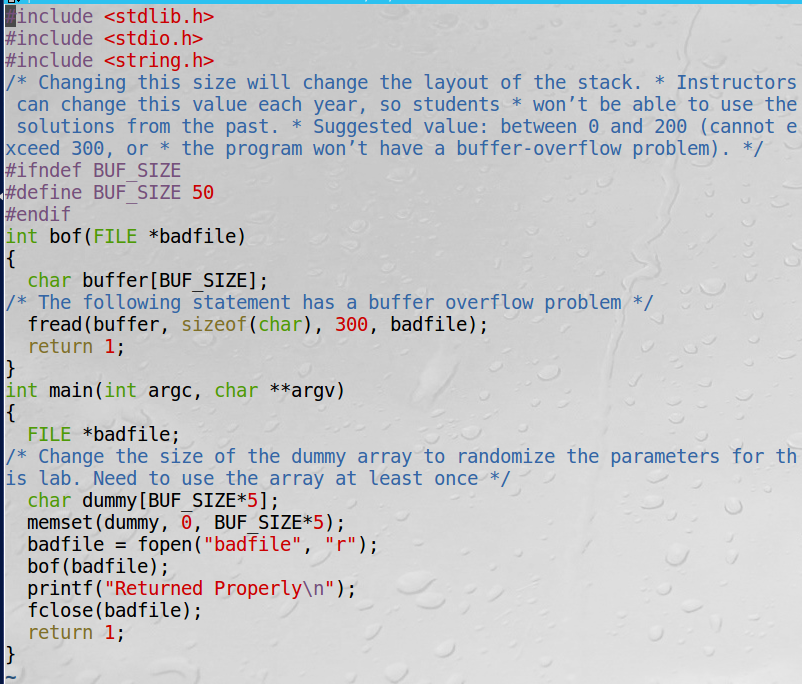


**Return-to-libc Attack Lab**

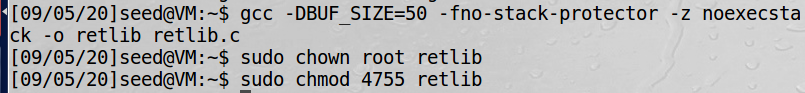
首先关闭地址空间随机化，将/bin/sh连接到zsh：



漏洞程序retlib.c，buffer size设置的50：

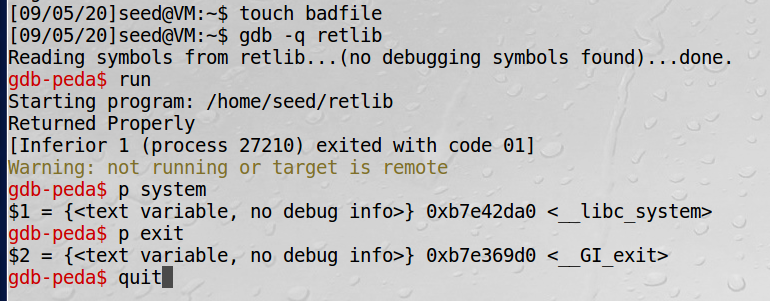


编译漏洞代码并将它设置为root所有的Set-UID程序：



**Task1：找出libc函数的地址**

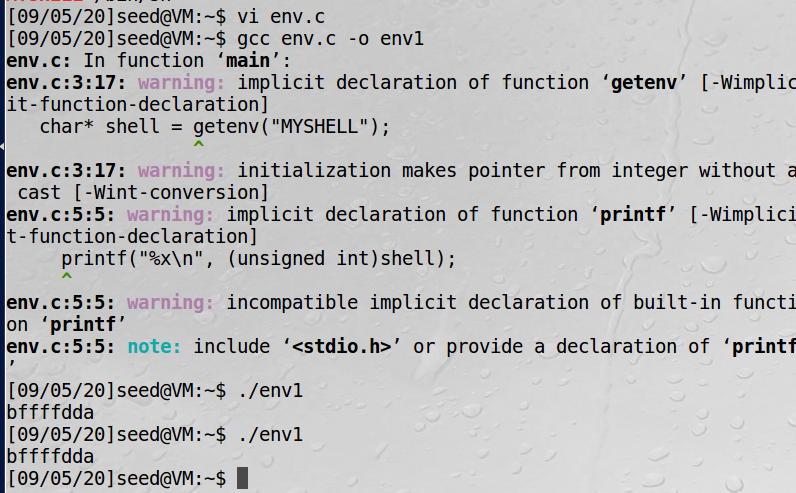
使用gdb命令，找出system（）的地址为0xb7e42da0，exit()的地址为0xb7e369d0：

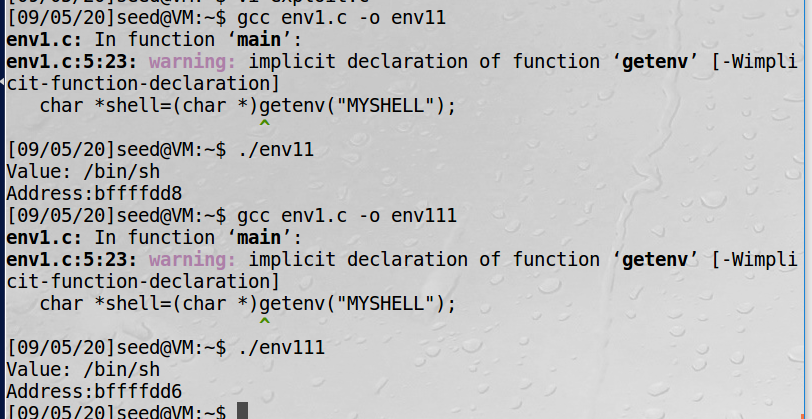


**Task2：将shell 字符串放入内存**

设置环境变量MYSHELL=/bin/sh（忘了截图）

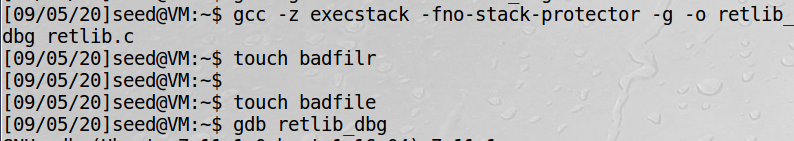
编译实验手册的代码（命名为env.c），运行可执行文件为env1时，地址为0xbffffdda，将可执行文件命名为env11时，地址为0xbffffdd8，命名为env111时可执行文件为0xbffffdd6得出规律，所以应该选择地址0xbffffdd6：

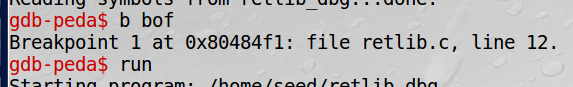


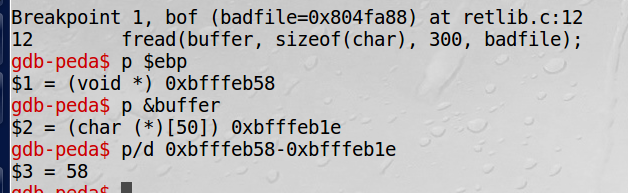


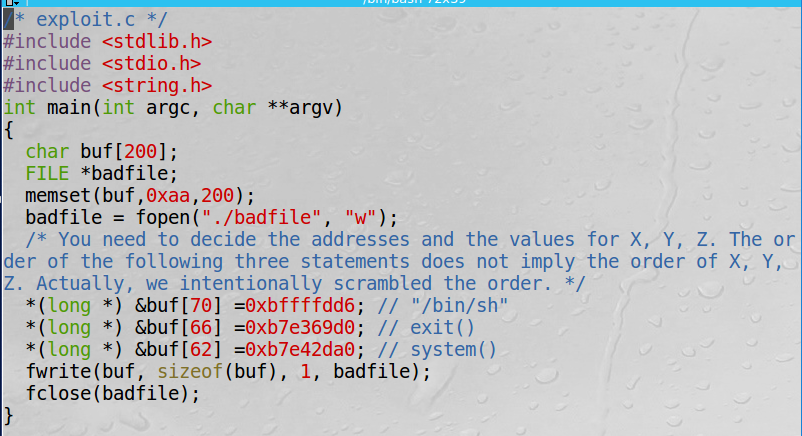
**Task3：利用缓冲区溢出漏洞**

利用gdb命令算出offset的值为58+4，所以创建badfile的代码中X，Y，Z应该分别为：62，66，70。并填入对应的地址：

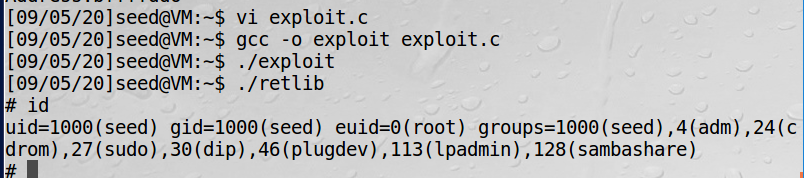






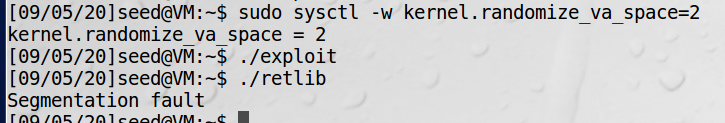


编译执行代码，可以看到结果拿到了root权限：

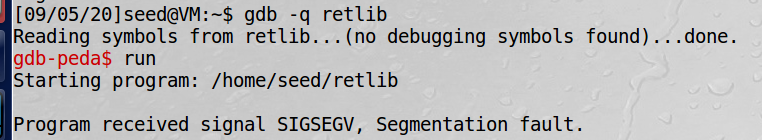


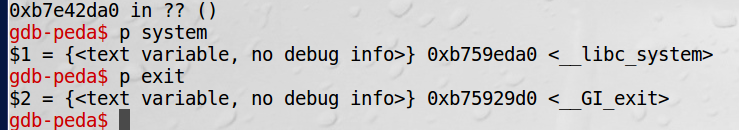
**Task4：打开地址随机化**

打开地址随机化设为2，然后再运行task3，发现不能拿到root权限：

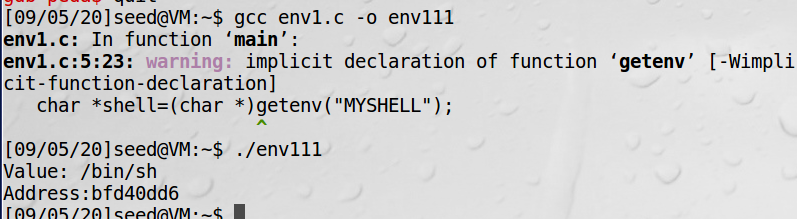


此时system（）和exit（）的地址发生了变化：

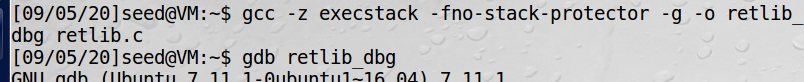


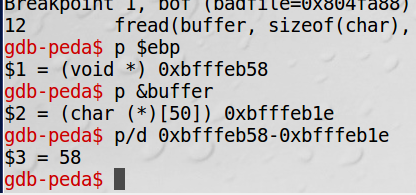


环境变量的地址并未发生变化：



X，Y，Z的值也并未发生变化：





**Task5：击败shell的对策**

关掉地址随机化，将/bin/sh重新指回/bin/dash，运行task3，发现不能拿到root权限：

