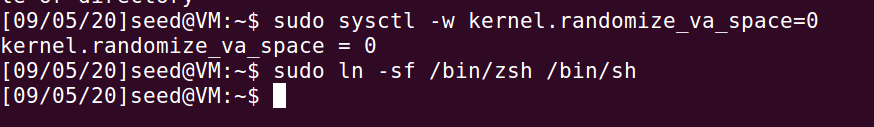
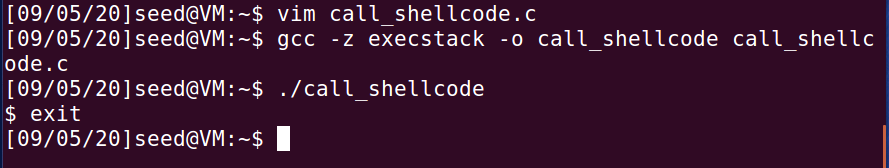
**Lab2-Report**

57118116 陈煜



Task 1: Running Shellcode

通过以下命令编译并运行call\_shellcode后，发现执行了/bin/sh，进入了zsh，并通过exit可以退出，说明代码是有效的，能打开zsh。



编写stack.c代码，通过以下方式编译，并将文件设置成Set-UID程序。





Task 2: Exploiting the Vulnerability

对stack进行gdb，在函数bof上设置断点，运行程序：

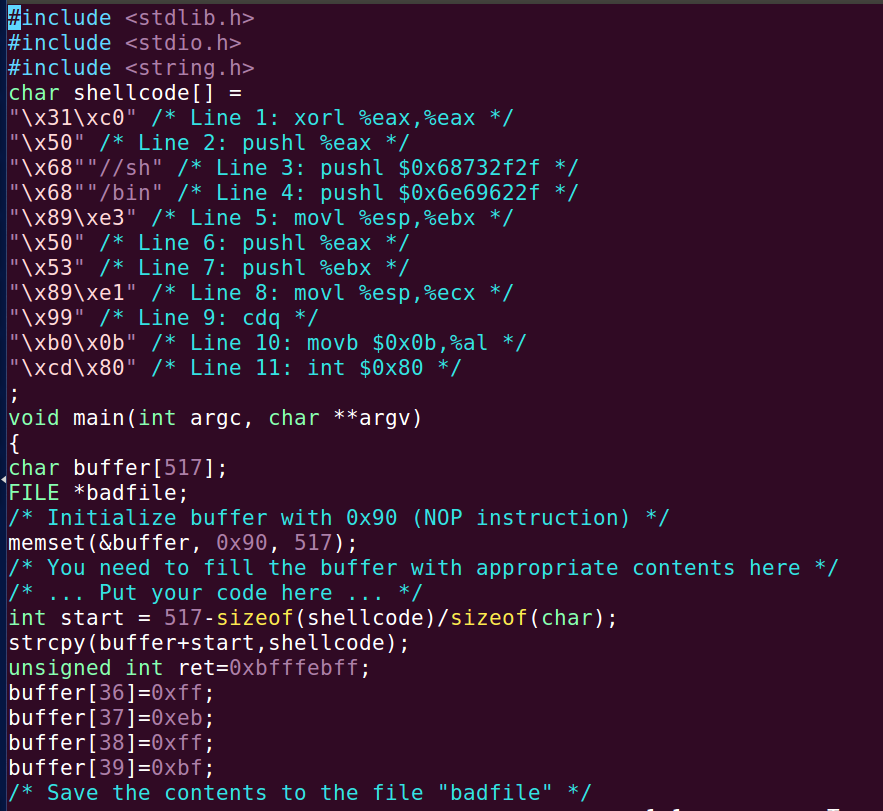


得到$ebp的值为0xbfffeb48：

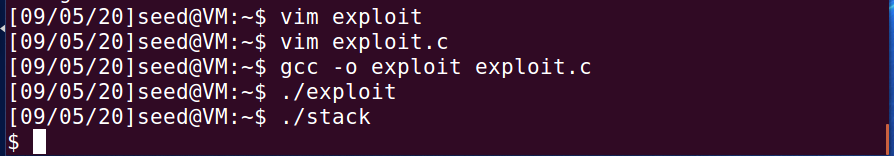


$ebp的值加8即可得到malicious code的起始地址在0xbfffeb50之后，这里取为0xbfffebff。

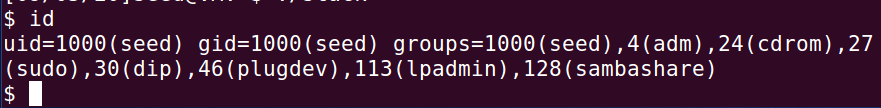
编写exploit.c代码并编译，程序内容如下：



先运行exploit程序，再运行stack程序，发现打开了zsh。



输入id，发现此时的euid为0，即当前的有效ID为root的id。

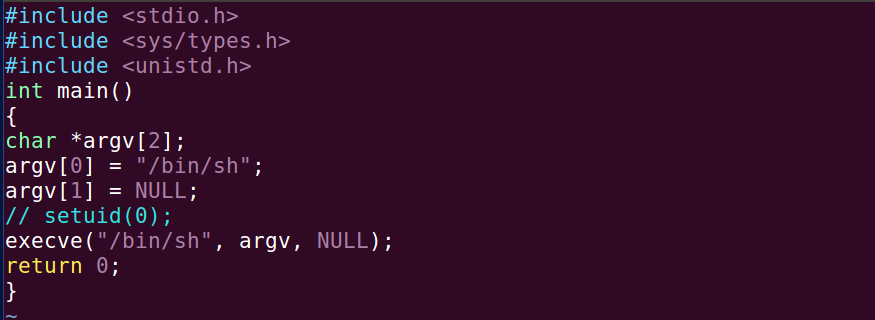


Task 3: Defeating dash’s Countermeasure

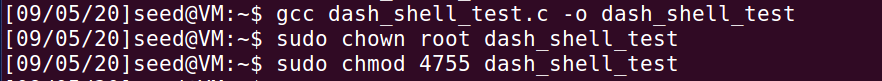
将/bin/sh转换成/bin/dash：



编译如下代码：



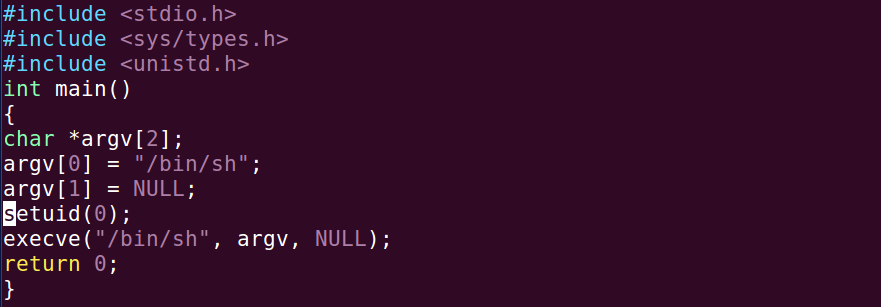
将其设置为Set-UID程序：



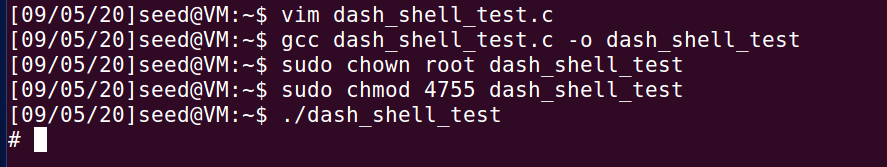
运行之后，发现进入了普通用户的shell：



修改dash\_shell\_test.c代码并重新编译：

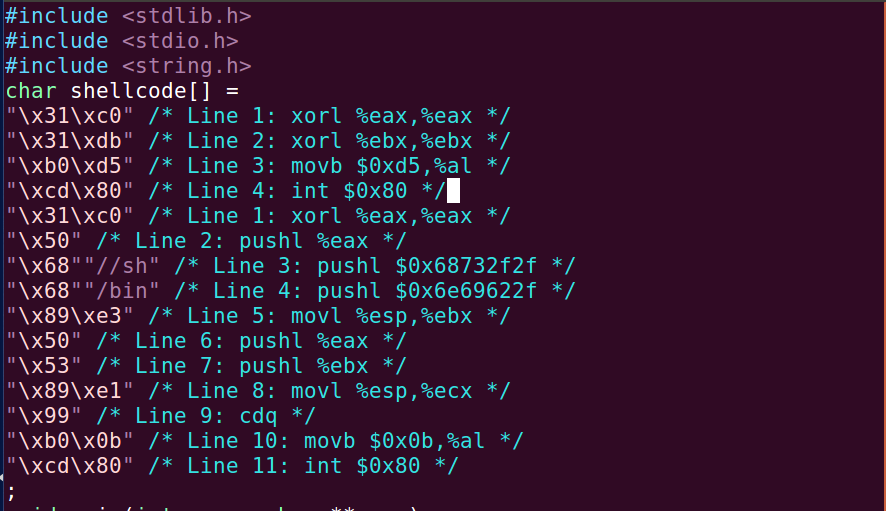


运行后发现进入了root用户的shell：

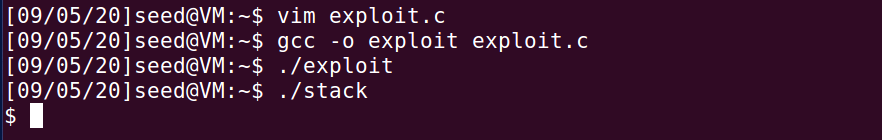


通过对比可以发现，setuid(0)能使程序进入root的dash。

更新exploit.c，在shellcode部分进行如下修改：

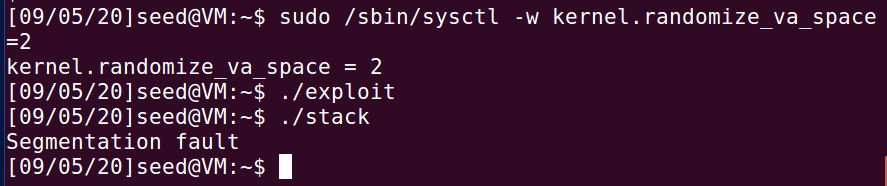


重新编译exploit并运行，发现仍能进入dash：

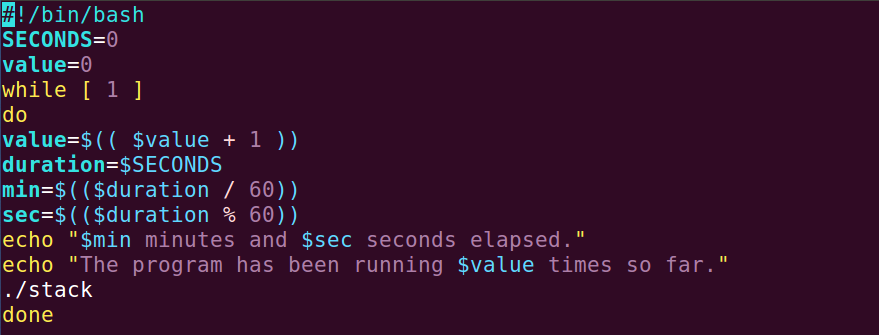


Task 4: Defeating Address Randomization

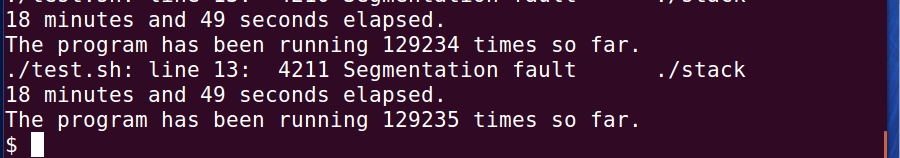
将kernel.randomize\_va\_space置为2之后，再次运行./exploit和./stack，因为堆、栈在内存中的位置不确定，攻击失败：



编写test.sh脚本，进行暴力攻击：



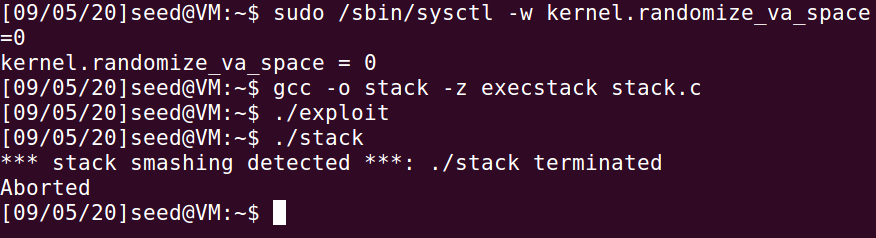
运行脚本test.sh后的结果：



尝试了18分钟49秒，一共129235次。

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

重新编译stack，输入的参数为noexecstack。运行task2的程序后发现出错，错误信息为Segmentation fault。说明Return Address不在栈上。

