****

实验报告

|  |  |
| --- | --- |
| 姓名： | 强珂阳 |
| 学号： | 57118106 |

东南大学网络空间安全学院

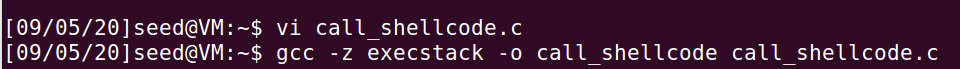
School of Cyber Science & Engineering

Southeast University

2020年9月

TASK1：Running Shellcode

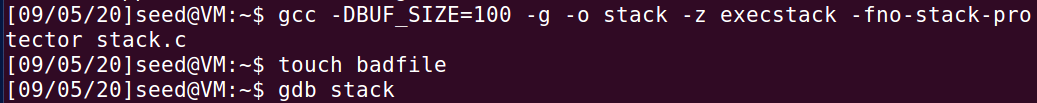
**编译并运行程序可以发现成功启动shell：**

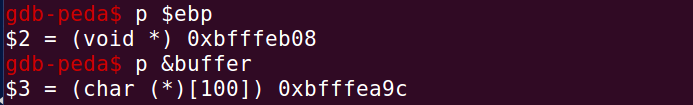




TASK2：Exploiting the Vulnerability



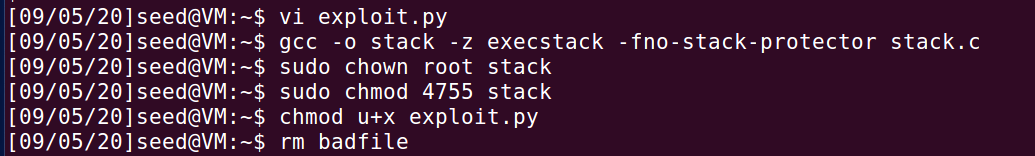




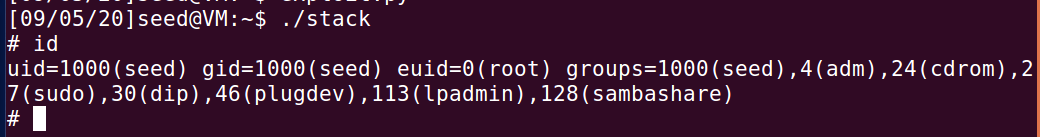


**ret=ebp+4就是return address，但由于中间填充了NOP，所以只需要选一个数值加上ebp即可有很大概率跳到恶意代码，offset为buffer[0]到ebp的距离加上4即为buffer[0]与return address的距离。**





**重新编译stack.c文件并运行，成功得到root shell：**



TASK3：Defeating dash’s Countermeasure

**注释掉setuid(0),运行得到：**

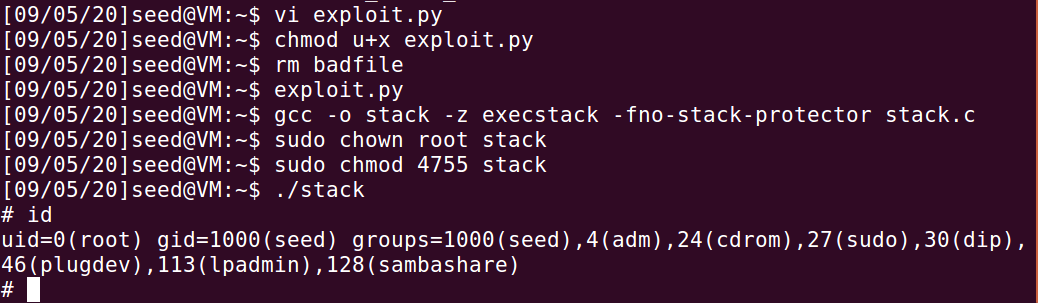


**不注释得到：**



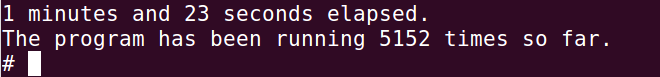
**可以发现，因为这是一个Set-UID程序，所以euid是root（0），但是ruid是seed（1000），会触发dash的防御机制，但是设置了setuid（0）后，euid=ruid=0，不会触发防御机制。**

**在修改exploit.py文件后，重新做task2，得到：**



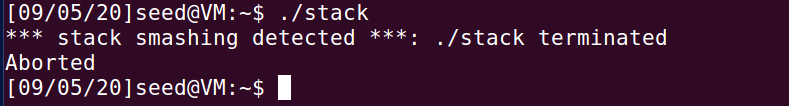
**也得到了root shell。**

TASK4：Defeating Address Randomization



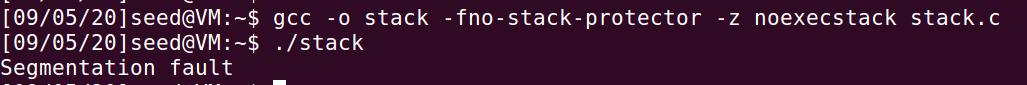
**在运行了5152次后成功运行了stack程序，获得了root shell。**

TASK5：Turn on the StackGuard Protection



**报错，运行不成功。**

TASK6：Turn on the Non-executable Stack Problem



**报错，运行不成功**

SUMMATION

**本次实验主要涉及了缓冲区溢出，比较有挑战性的是task2中补充函数，根据老师上课时的步骤，通过调用gdb来找到buffer[0]与return address，这样才能通过buffer溢出更改返回地址。**

**同时，还了解了几个保护方案。地址随机化会增加猜测地址的难度，但是在地址有限的情况下，我们仍然可以通过暴力破解来达到目的。但是non-executable stack以及stackguard会直接阻止这种攻击。针对dash对于set-uid程序调用shell的防御，我们可以通过更改调用的shell或者利用setuid()函数使得euid≠ruid来规避这个防御机制。**

**此外，由于这次实验的命令行参数多，因为漏参数等问题出现了好几次，导致实验的不顺利需要引以为戒。**