**操作系统实验二实验报告**

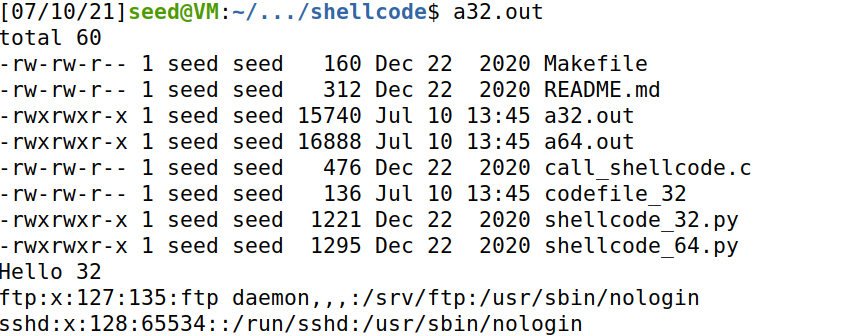
**基本信息：**

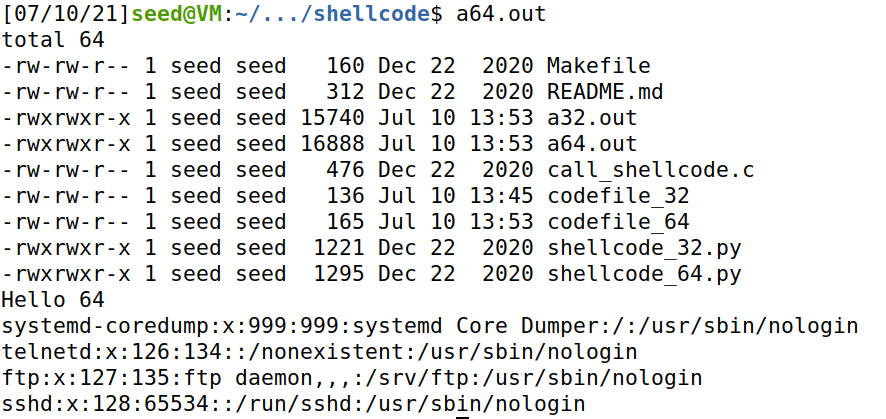
完成人姓名：顾琰 学号：57119117 完成日期：2021 年 7 月 10 日

**实验内容：**

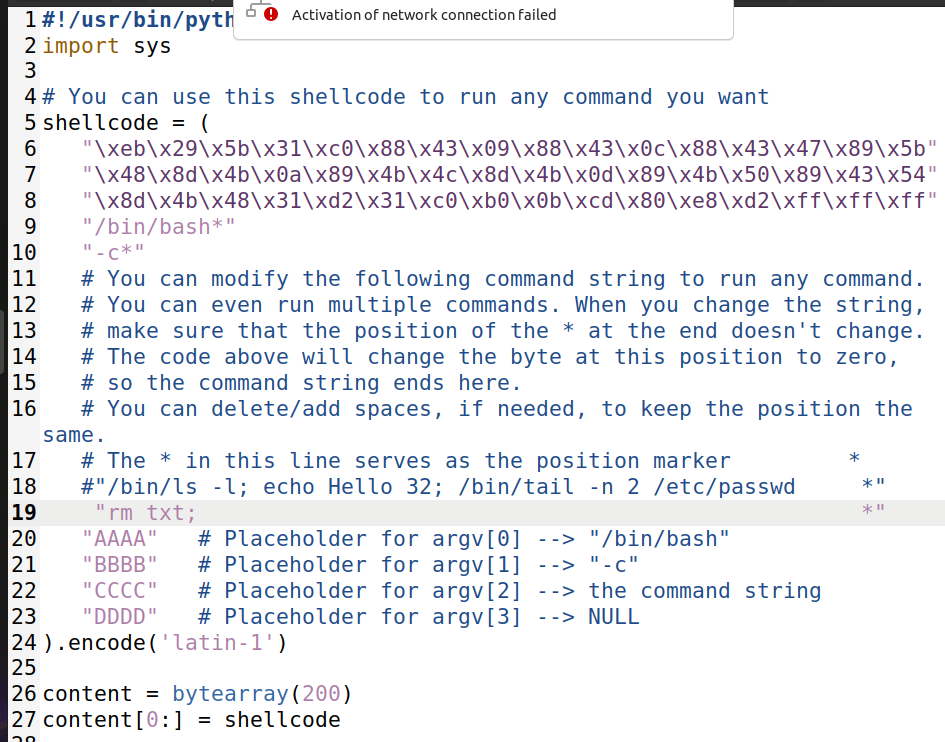
**Task 1: Get Familiar with the Shellcode**

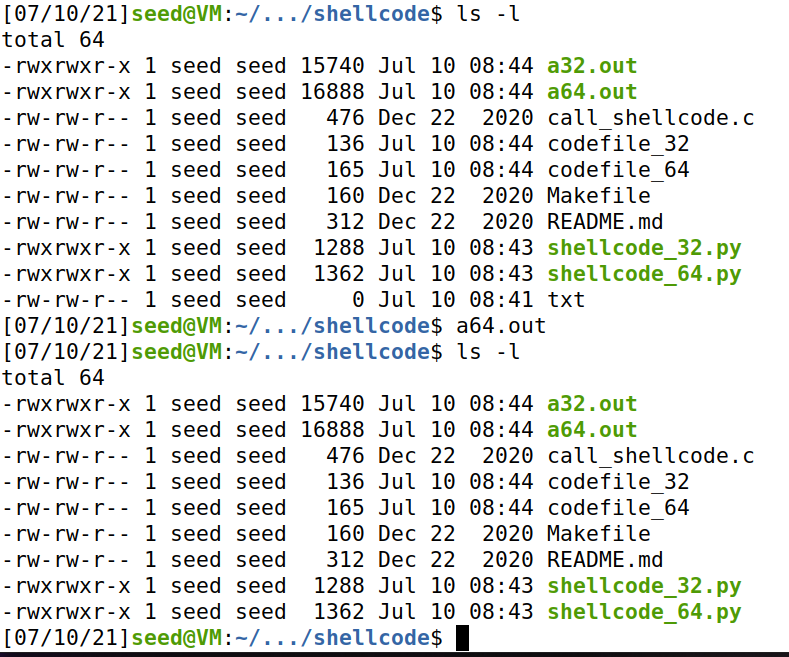
编译原始的shellcode运行测试:





讲shellcode功能改为修改某一文件:



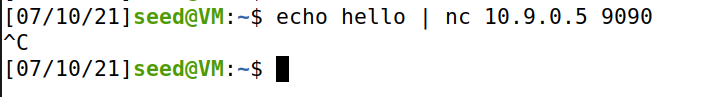


txt文件消失，删除成功。

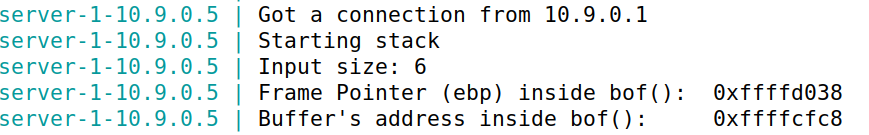
**Task 2: Level-1 Attack**

测试服务器：

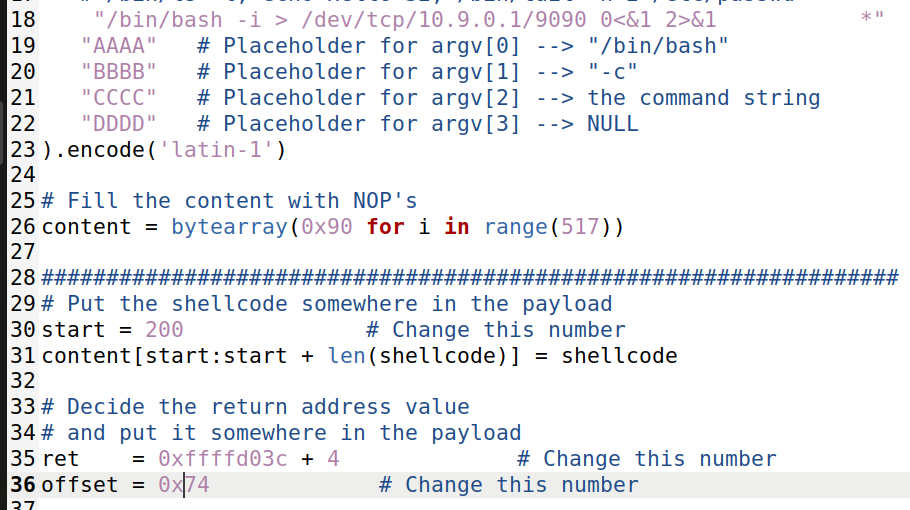
攻击端：



服务器：

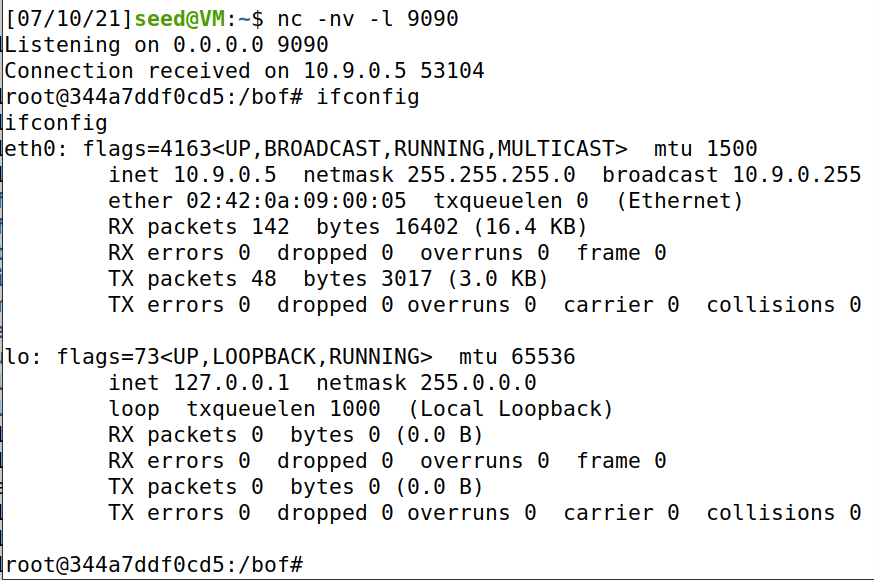


修改 exploit.py 文件：



编译并发送给 L1 服务器：

IMG_256

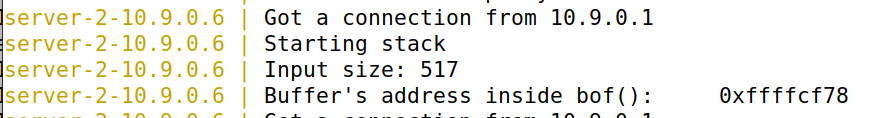
在本地客户端上监听：

可见，已经可以在攻击端上执行服务器的 shell 了。

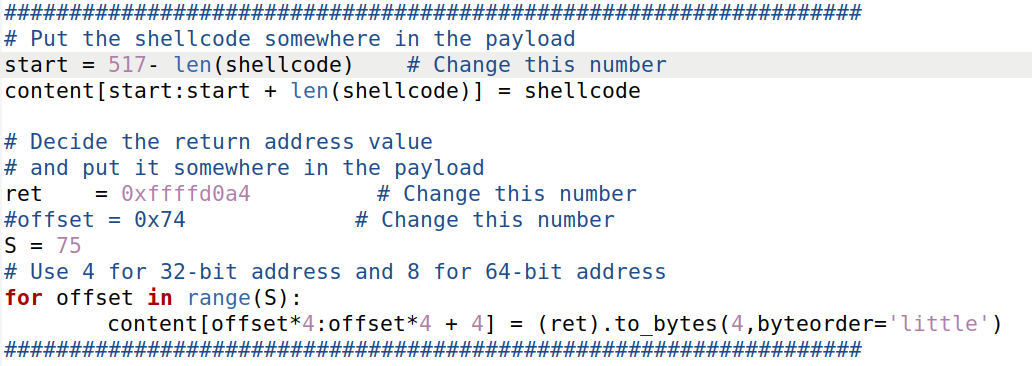
攻击成功！

**Task 3: Level-2 Attack**

正常链接服务器获取其 Buffer 地址：



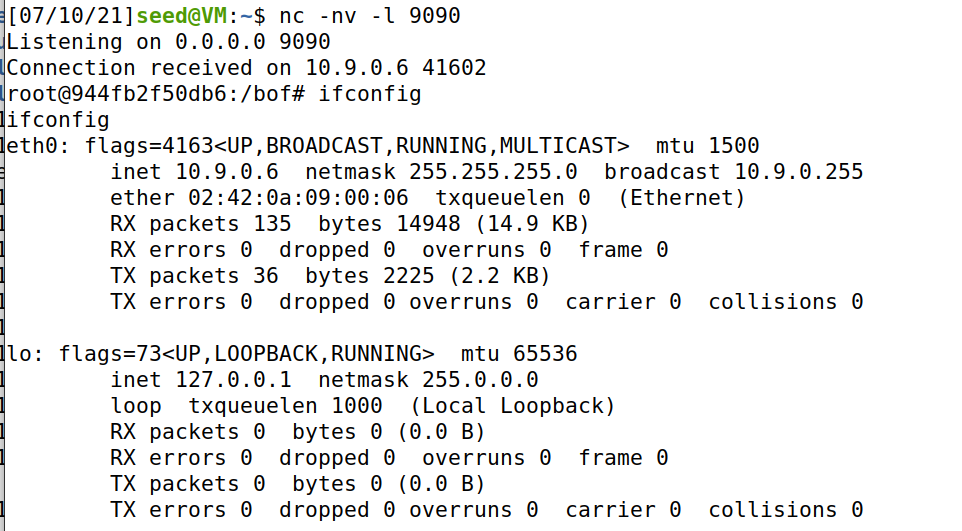
改写攻击程序： 利用循环在我们插入的 shellcode 前的每一个位置都输入其地址，总会有一个覆盖到返回地址。



编译攻击：

IMG_256

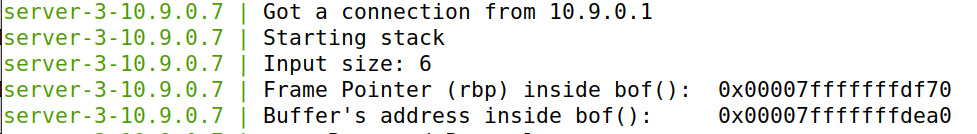
监听端：



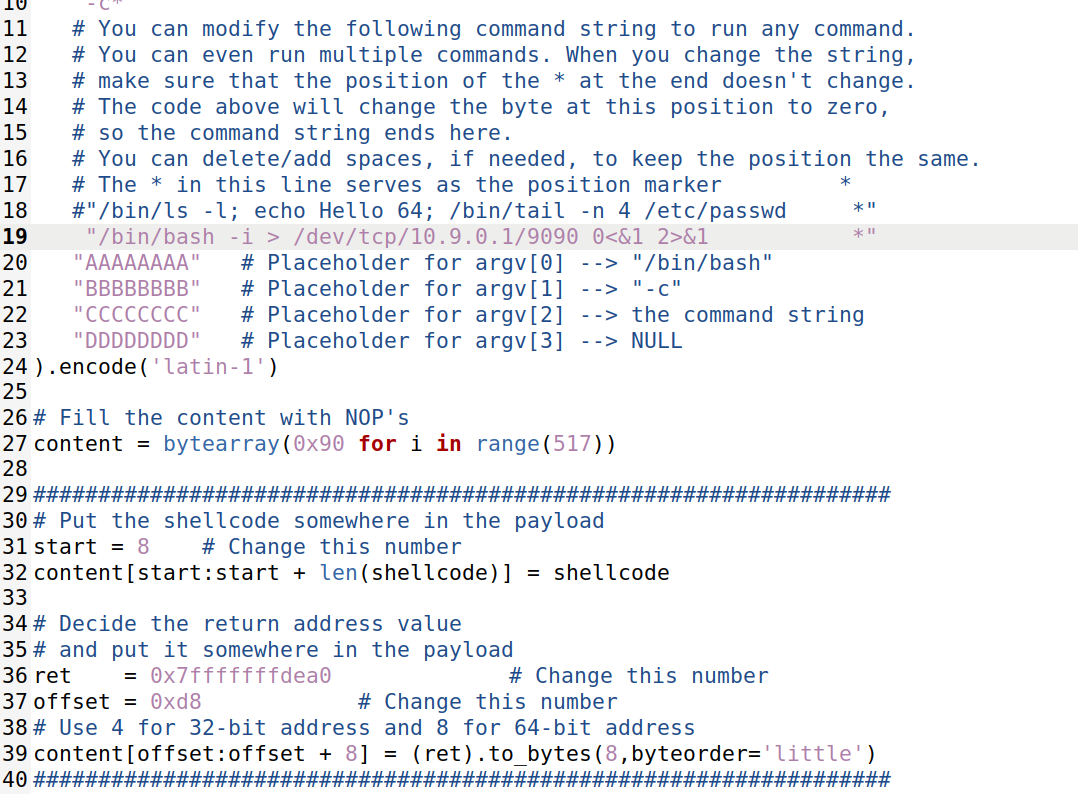
Ip 地址为服务器的地址，说明入侵成功。

**Task 4: Level-3 Attack**

正常链接服务器获取相关信息：



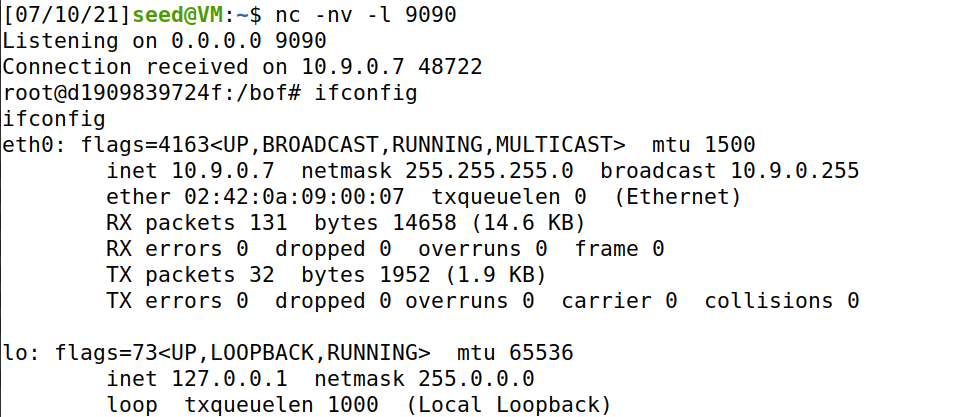
修改攻击代码，由于 64 位地址 0 的存在，我们的攻击程序在刚刚写到返回地址时便会停止，不再进行后面的写入。所以我们需要把我们的攻击 程序写到 buffer 内（返回地址下方），这样才能在写入停止前把所有攻击程序录入完毕，同时注意修改返回地址为 buffer 的入口地址，这样经过指令跳转可以执行到我们的攻击程序。



攻击端：

IMG_256

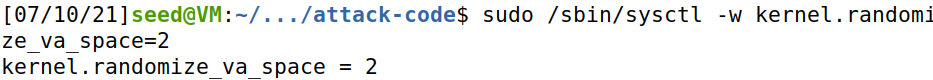
监听端：



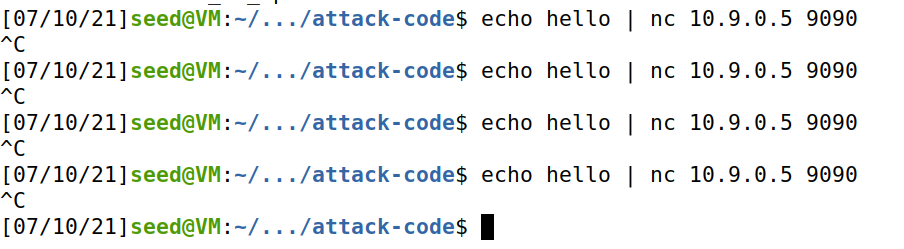
Ip 地址为服务器的地址，说明入侵成功！

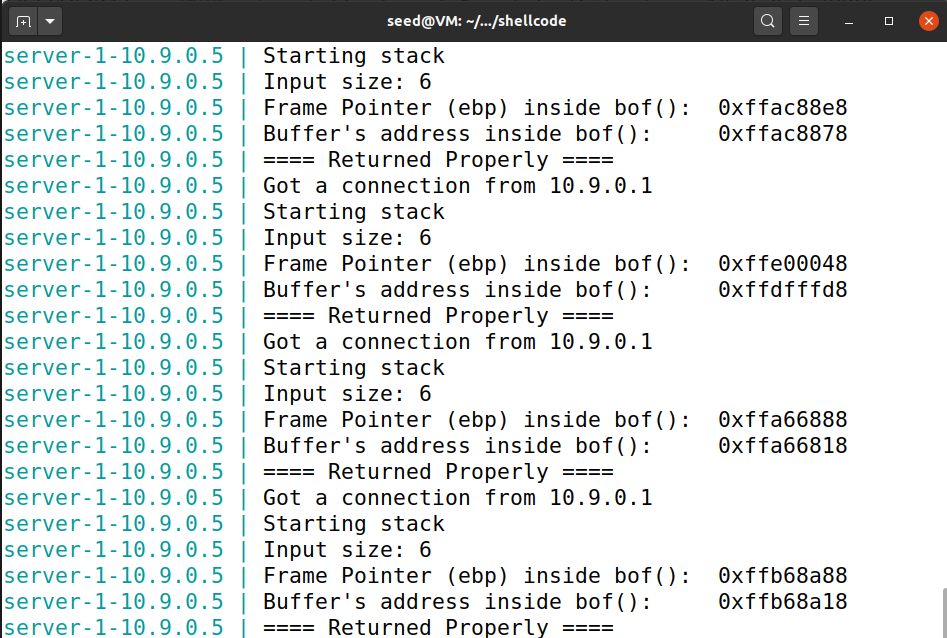
**Task 6: Experimenting with the Address Randomization**

打开ASLR:

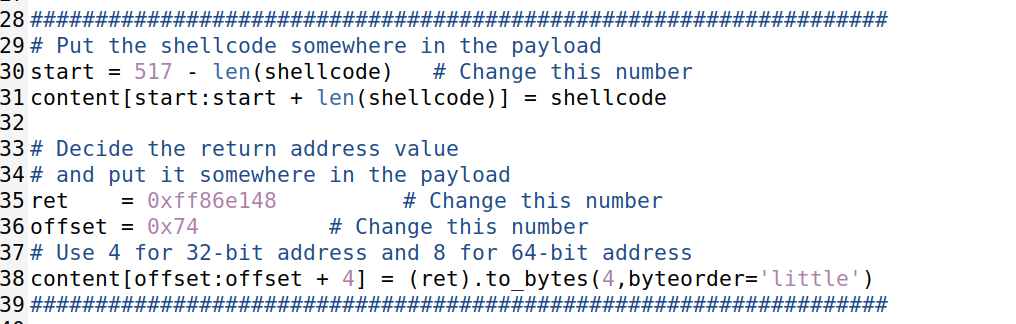


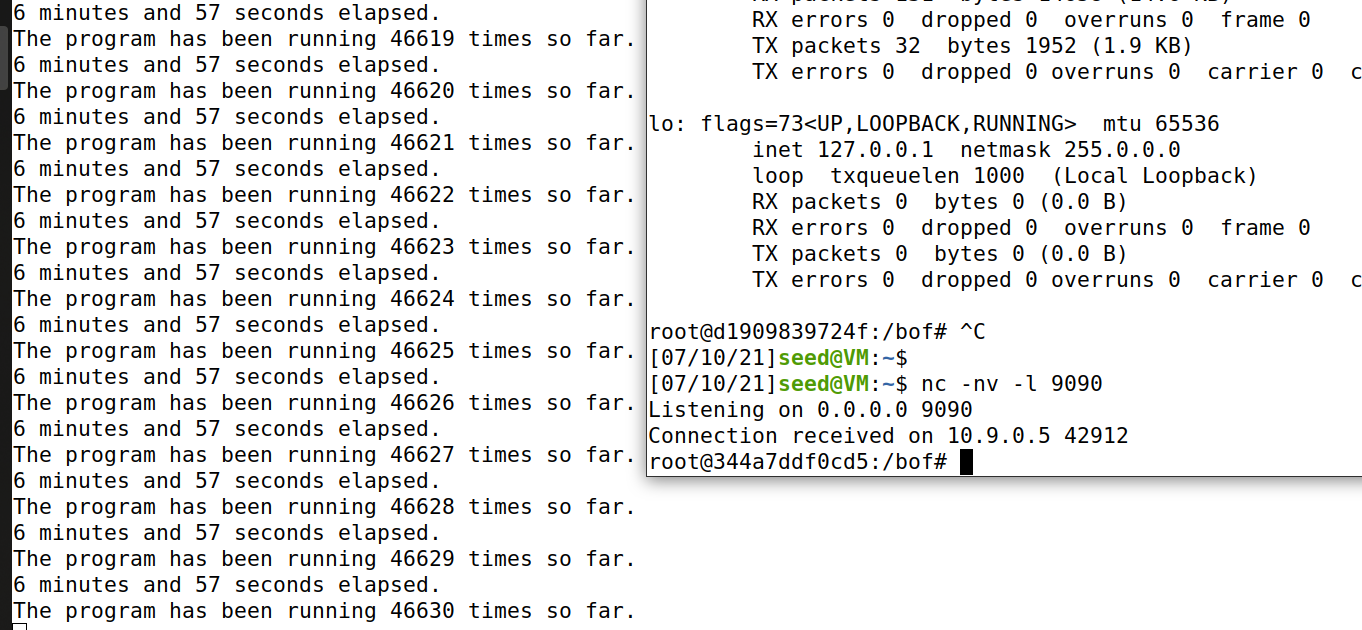
观察栈地址变化。





大致调整代码后，使用 shell 脚本暴力攻击。





耗时6分57秒暴力攻击成功。

**实验体会：**

本次实验是我们的第二次实验，经过本次实验，我总结了如下的知识点:

①通过 strcpy()等不检查越界的函数的复制溢出，我们可以把我们的代码复制到栈内， 并且通过获得栈指针和变量指针并通过计算来修改其函数的返回位置为我们插入的代码地址，以此使得被攻击者在返回函数时返回到了我们插入的代码位置，并以其高权限来执行我们的代码，从而达到攻击目的。

②为了使得我们的攻击不那么具有随机、低精确性，我们在我们要插入的代码中填满nop指令，其作用是跳到下一个指令，这样我们在指定我们插入的代码地址时，也可以指定到其前面的某一个nop指令，最终其也会一步一步跳到我们插入的代码段并执行。

③对抗堆栈溢出攻击的方式

1. 栈地址随机化，每次在其前加入一段随机的地址。
2. 检测栈是否越界