**操作系统实验二实验报告**

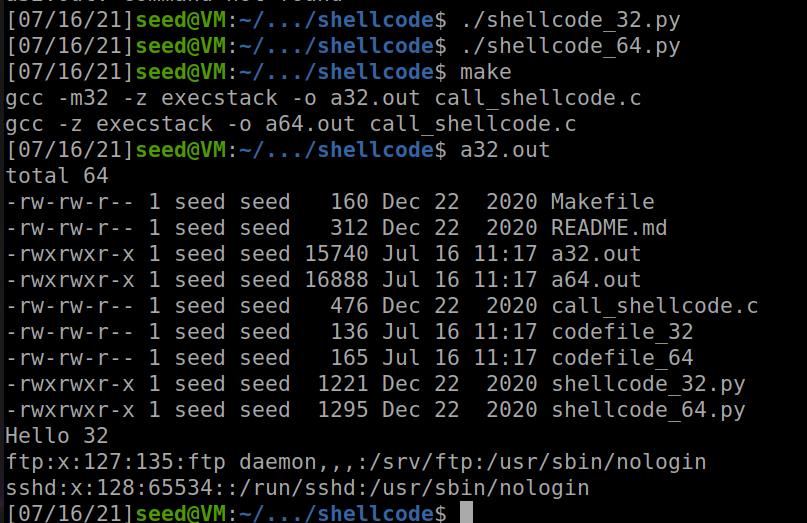
**基本信息：**

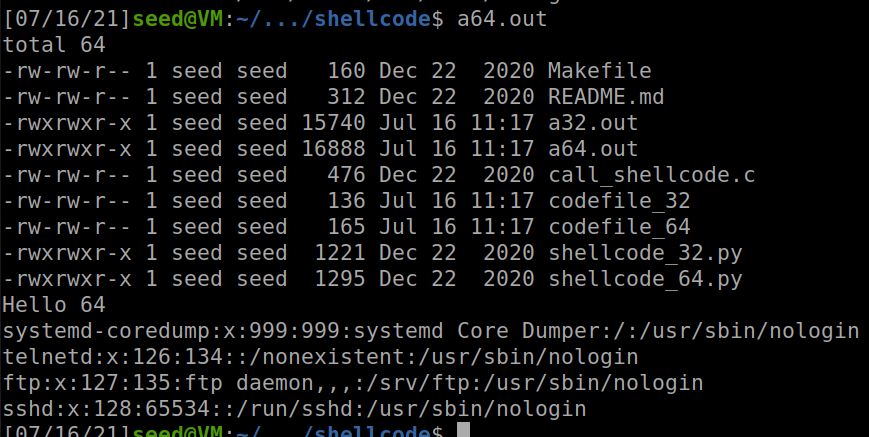
姓名：马运聪 学号：57119115 完成日期：2021.7.17

**实验内容：**

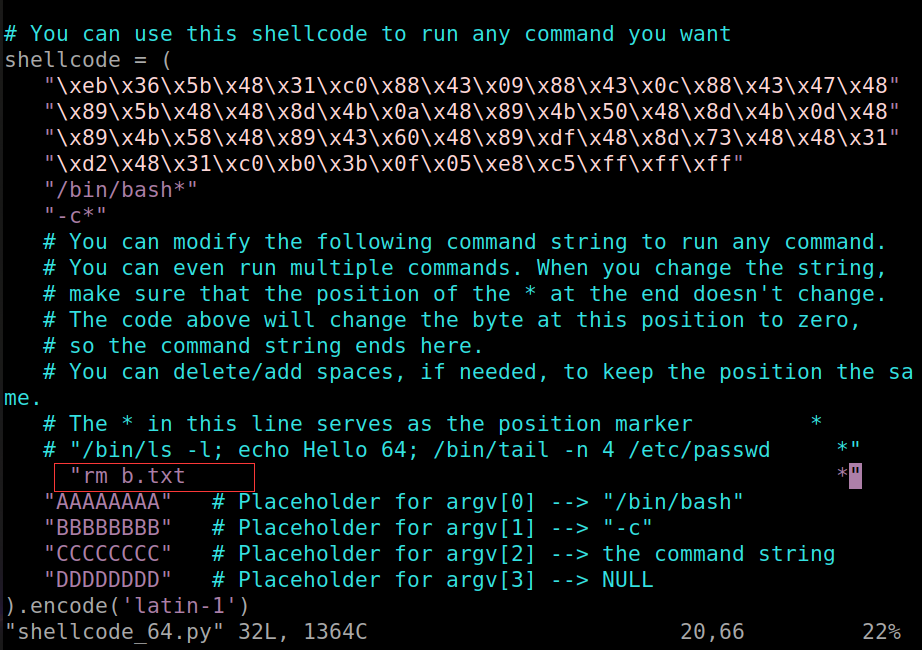
**Task 1: Get Familiar with the Shellcode**

编译文件进行测试

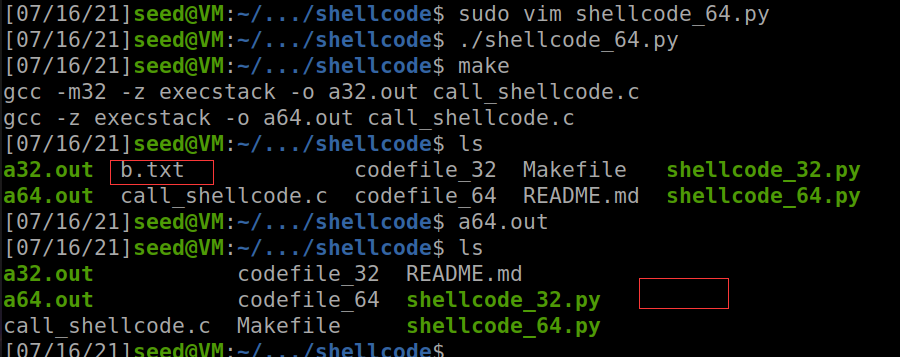




修改shellcode—32.py文件使功能改为删除b.txt文件



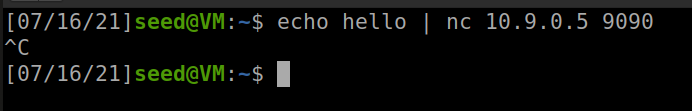
在编译执行后，b.txt文件已被删除

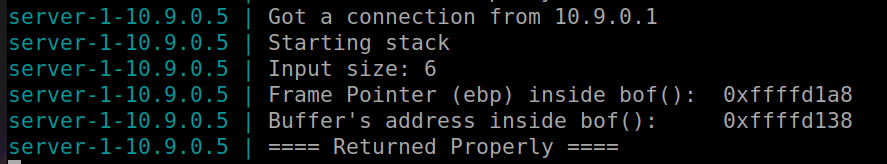


攻击成功

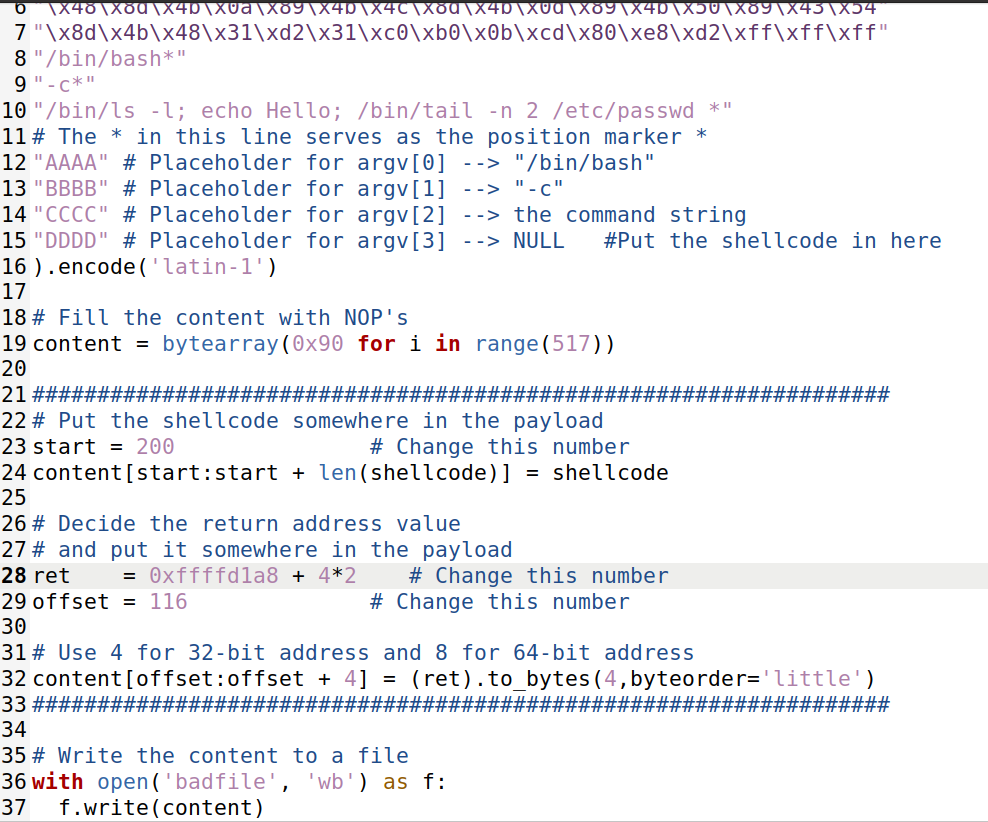
**Task 2: Level-1 Attack**

测试服务器，获得地址：

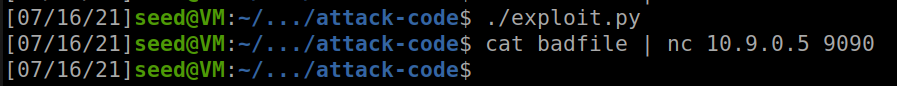




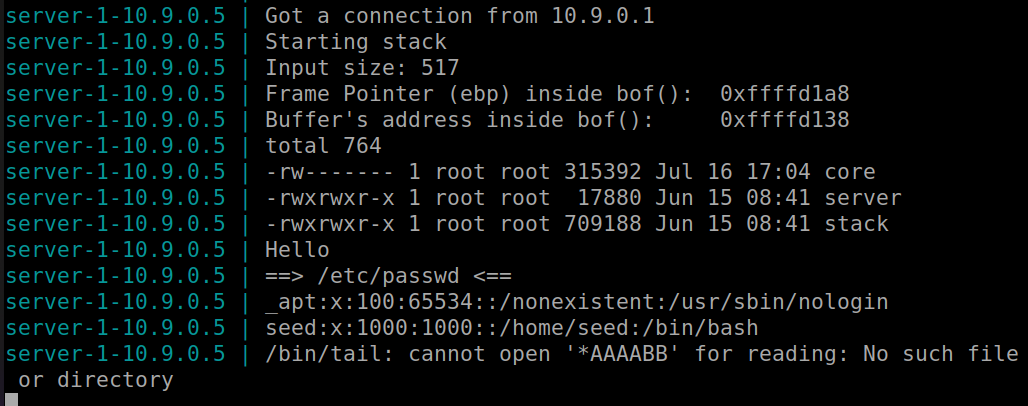
修改exploit.py文件



编译并攻击

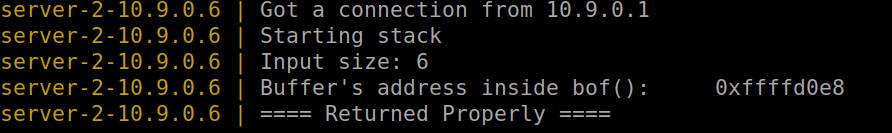


从服务器上可看到执行的攻击——攻击成功

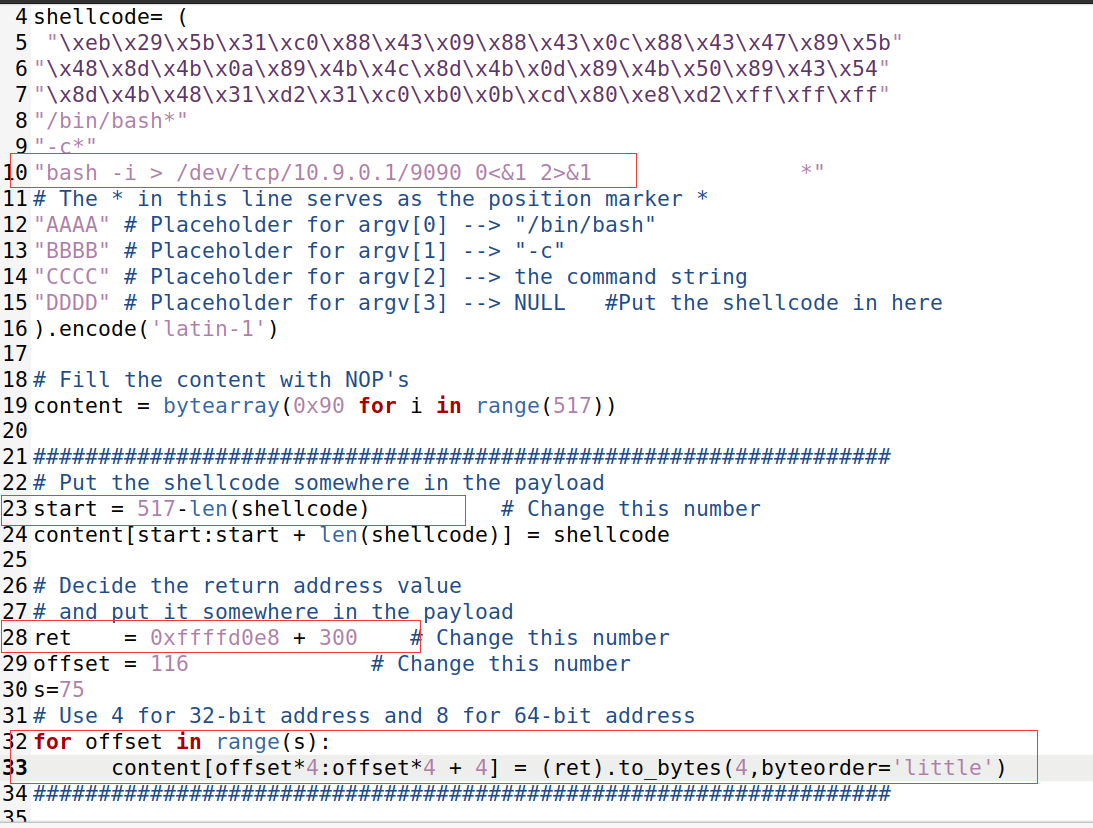


**Task 3: Level-2 Attack**

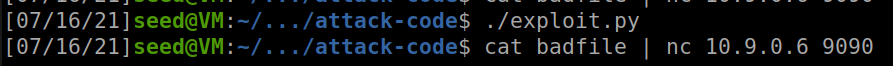
向服务器发送信息查看地址

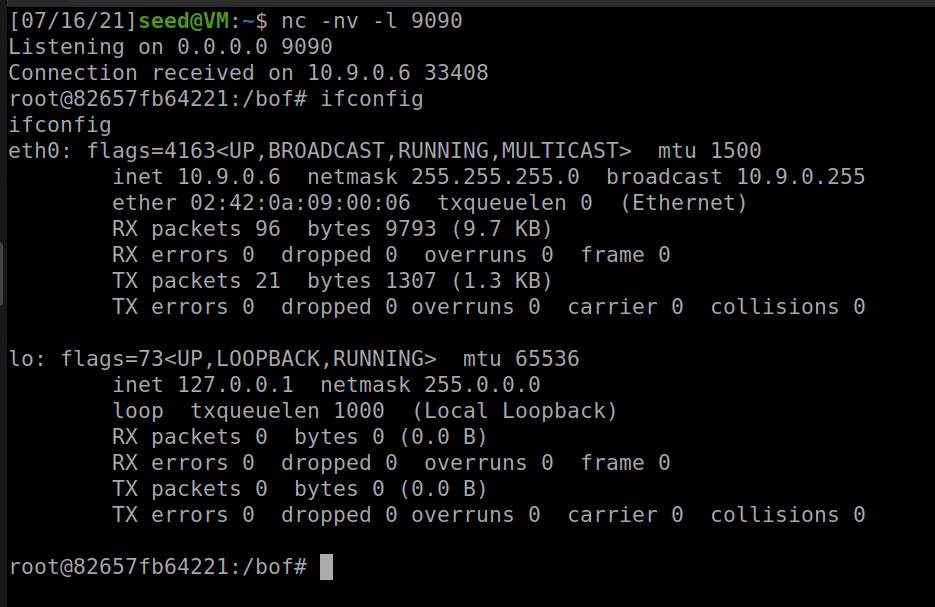


改写攻击程序：该换shellcode代码，然后利用循环在shellcode 前的每个位置都输入这个地址，使总有一个覆盖到返回地址。



编译文件，执行攻击，同时在本地客户端上监听

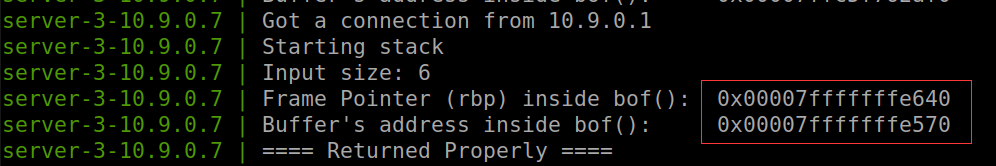




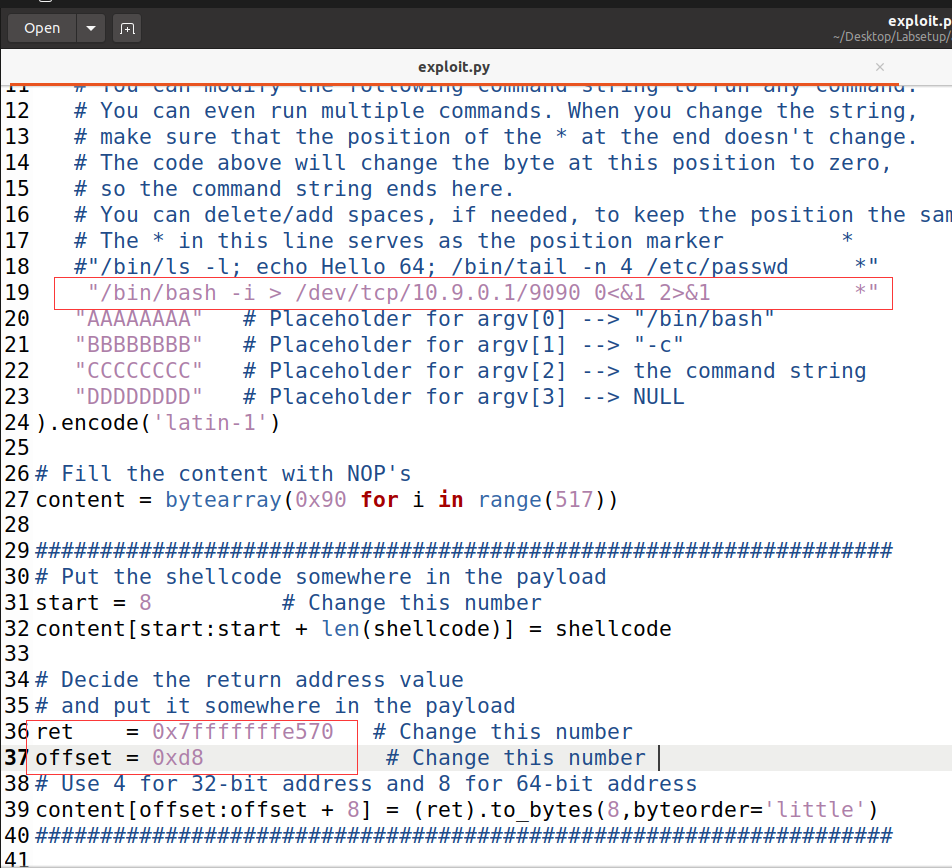
从监听器上可见攻击成功

**Task 4: Level-3 Attack**

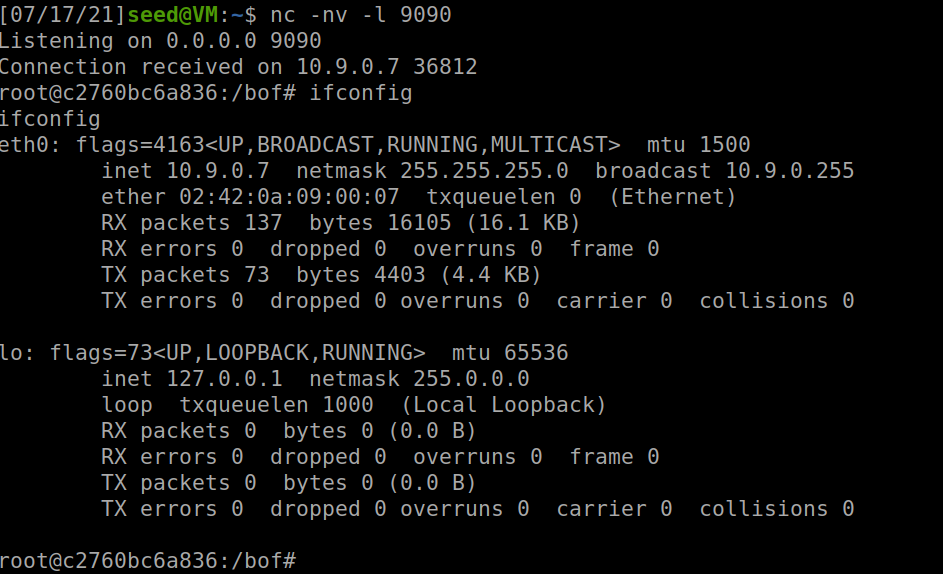
链接服务器获取得到地址信息：



修改exploit.py文件，使用task1的shellcode。由于 64 位地址 0 的存在，因此当攻击程序在刚刚写到返回地址时便会停止写入（srtcpy（）函数遇到0停止复制）。把攻击程序写到 buffer 内，同时修改返回地址为 buffer 的入口地址，这样经过指令跳转可以执行到攻击程序。



运行exploit.py文件和执行攻击，同时在监听端观察：



由图可知入侵成功！

**实验体会：**

在本次实验中，我学习了如何利用缓冲区溢出的漏洞进行攻击的方法，主要思想为：

当程序执行strcpy时，程序将char[]中的内容拷入buffer中，当参数容量大于buffer的长度时，buffer后面的字节内容被覆盖掉了，其中包括了ebp, ret地址。因此当我们把char[]内容改造，即更改为攻击程序badfile的函数地址时，在运行程序后，就会跳转到程序内按规定不能到的地方，则可以使它变成我们想要跳转到的bug函数地址。

而且，在进行攻击之前，我们必须更改栈地址随机化机制，因为这是预防缓冲区溢出的防御之一。