**Buffer Overflflow Attack Lab (Server Version)**

作者：黄禧敏

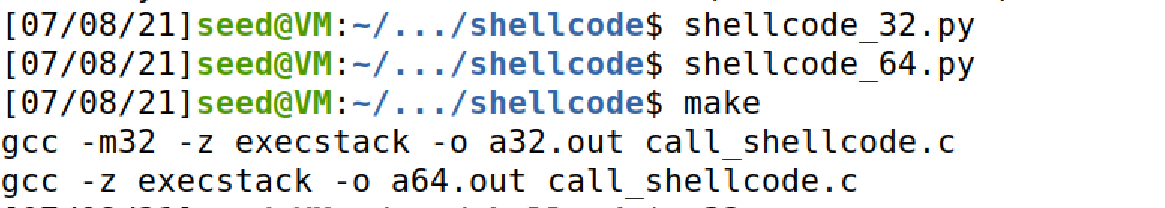
学号：57119109

日期：2021/7/13

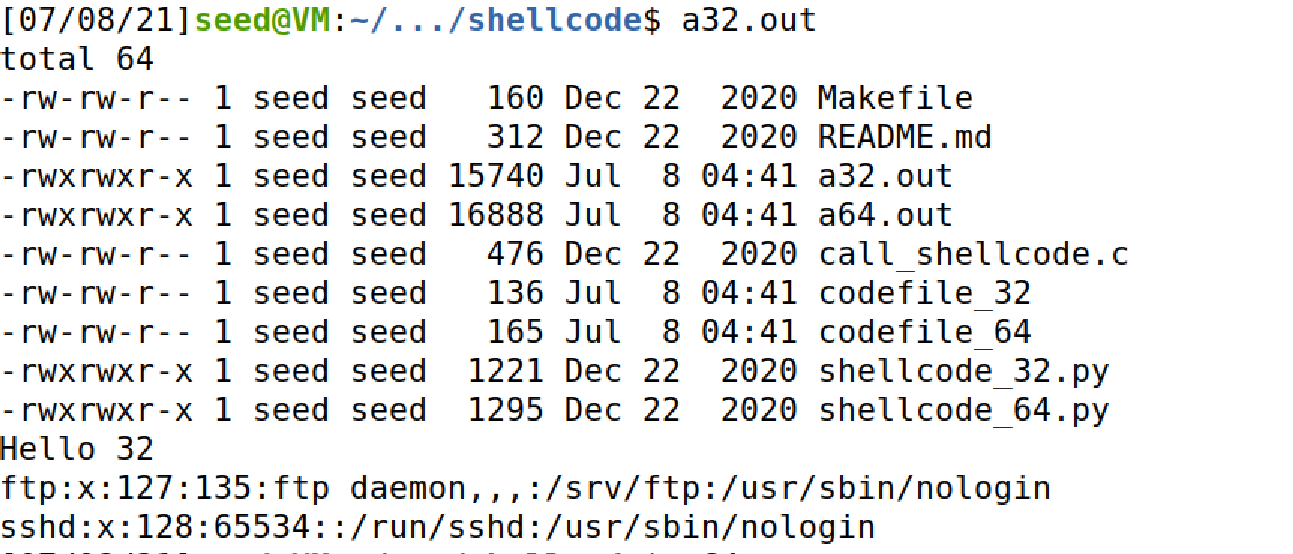
**Task1:**

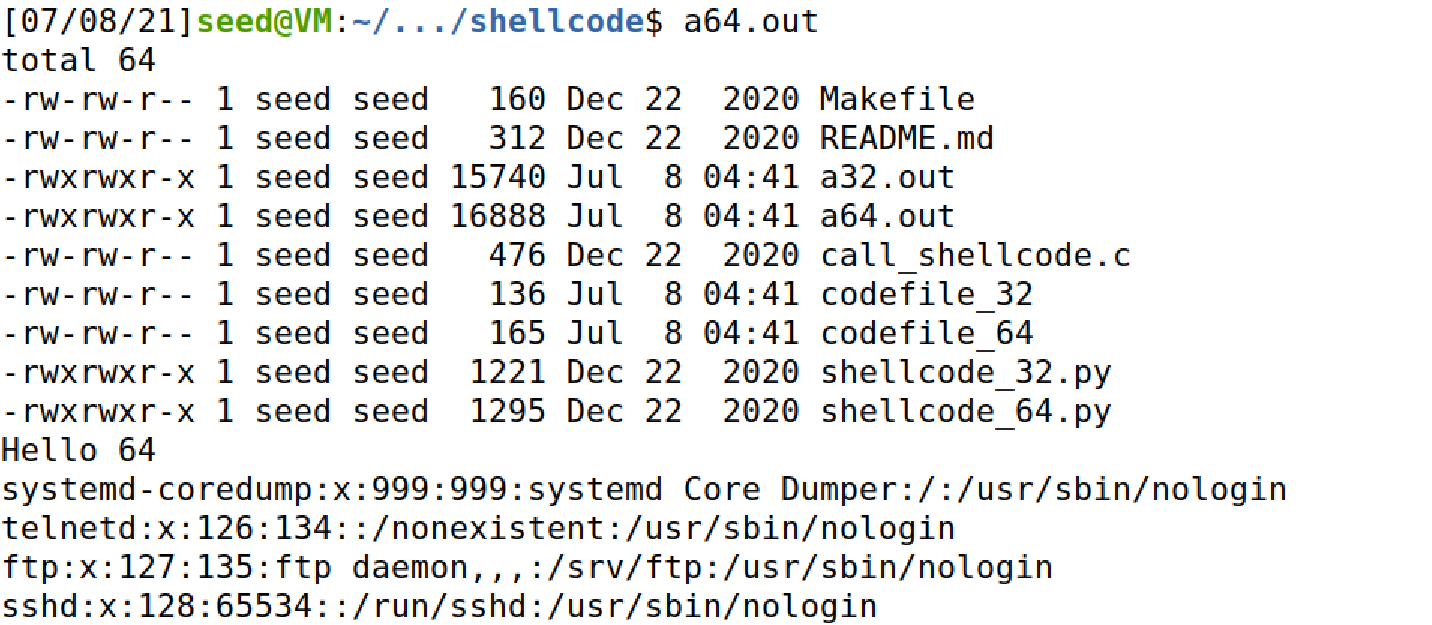
Get Familiar with the Shellcode

运行shellcode\_32.py shellcode\_64.py生成badfile;编译运行a32.out a64.out



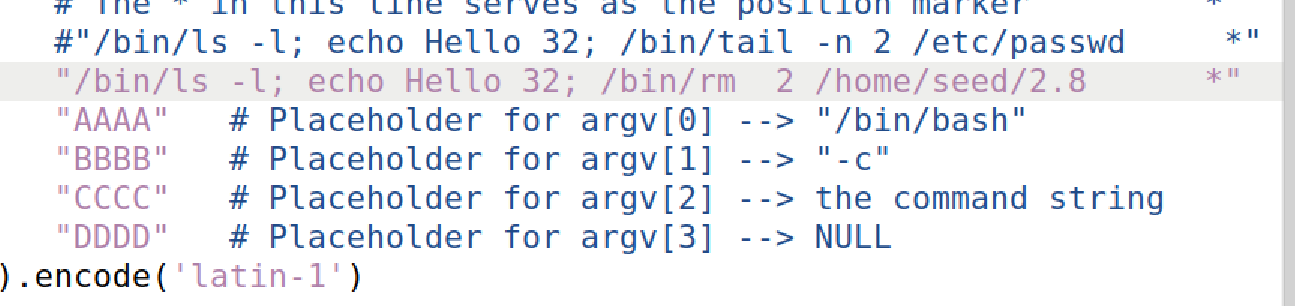
实现攻击代码，打印passwd；



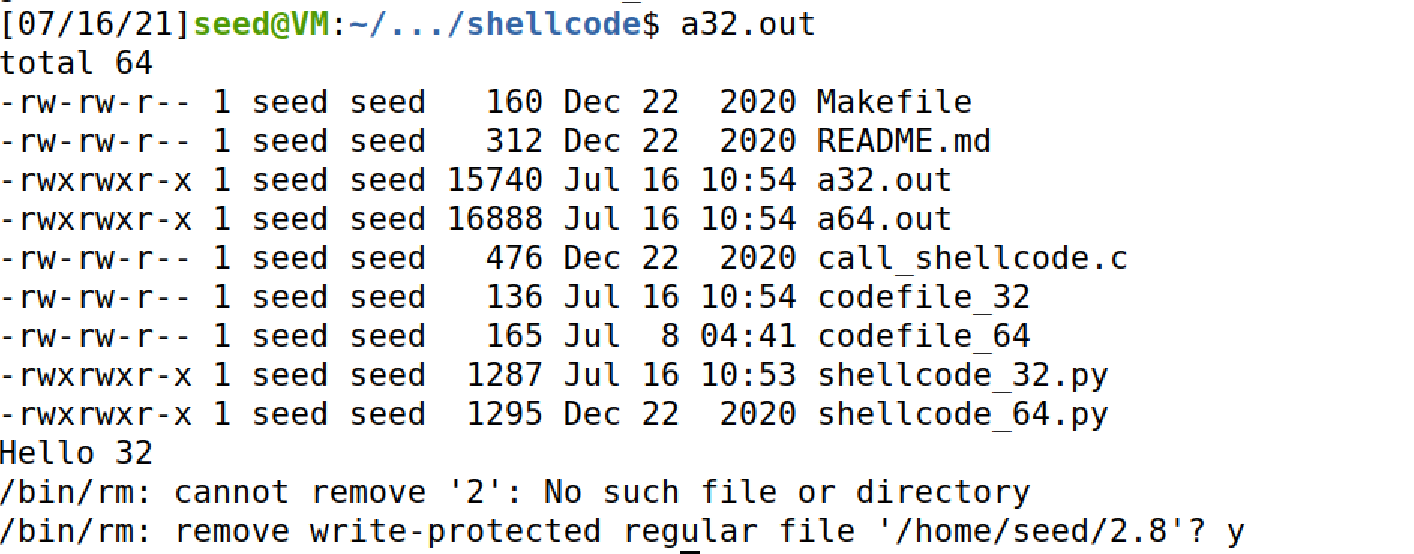


Task: Please modify the shellcode, so you can use it to delete a fifile. Please include your modifified the shellcode in the lab report, as well as your screenshots.

讲shellcode中的目标代码改为删除/home/seed中的一个文件2.8

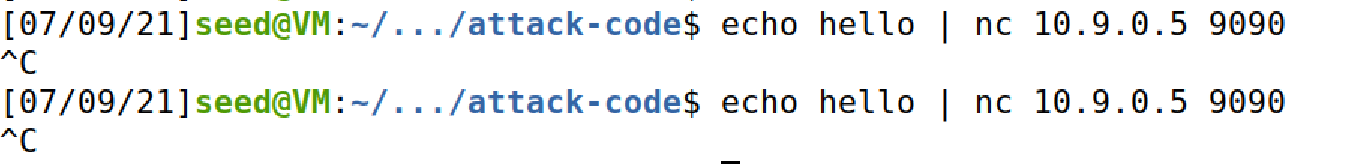


运行后成功删除；

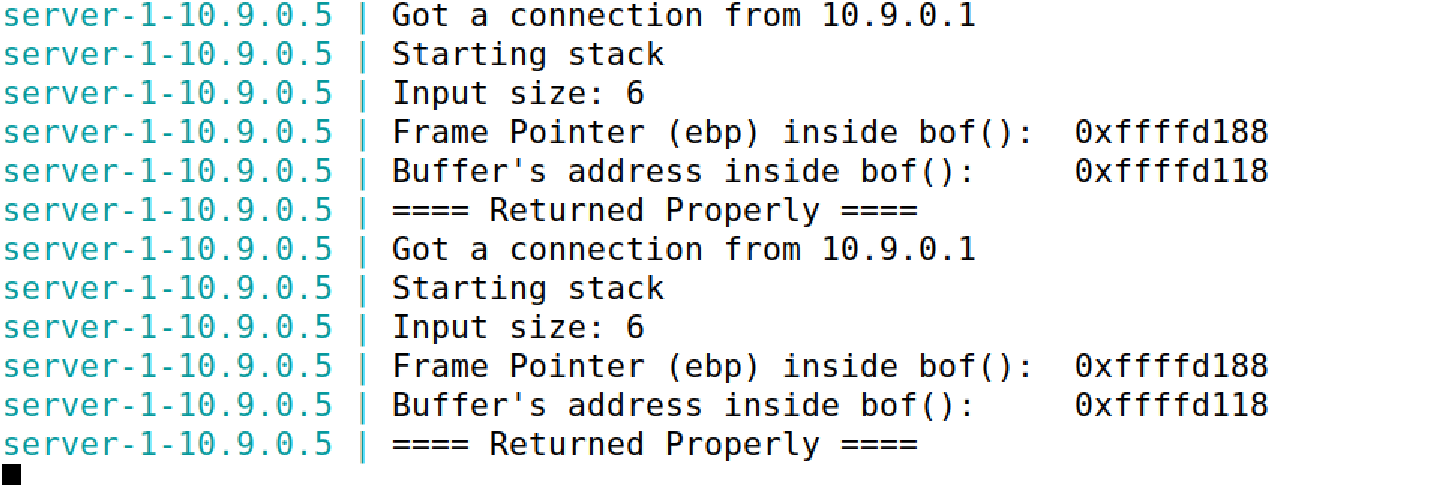


**Task 2:**

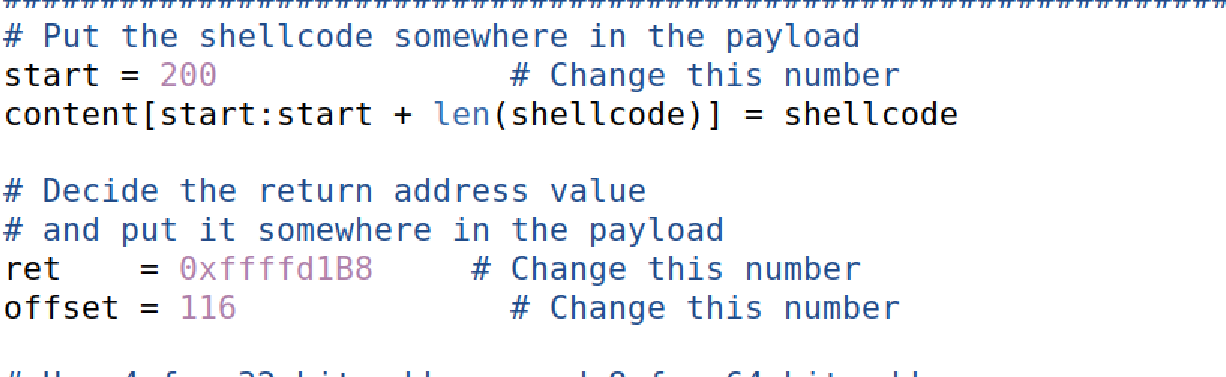
利用缓冲区溢出对return address重定向，执行自定义攻击代码；



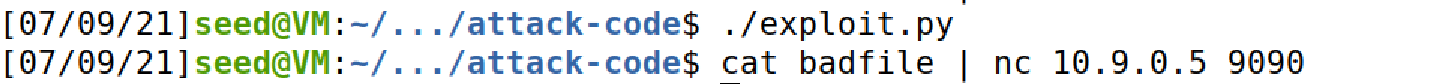
连续两次得到同样ebp和buffer地址



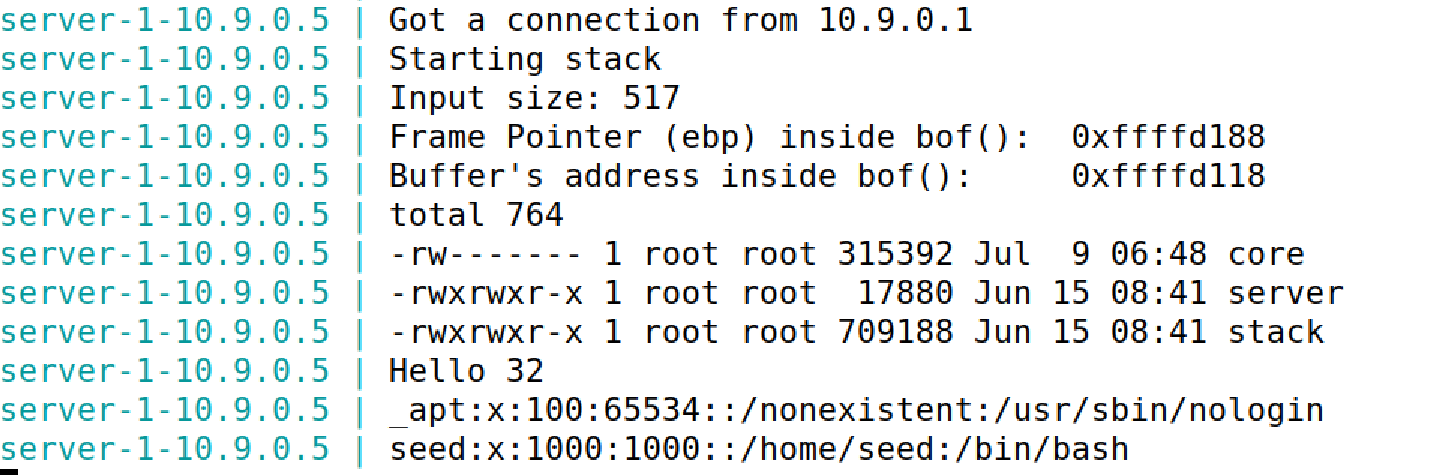
修改exploit.py



上一步已知ebp地址为0xffffd188，令新的返回地址稍大于ebp，为0xffffd1B8，将ret放至原return address的位置, 即相对buffer首地址为ebp-buffer+4=116，将shellcode放置ret之后。



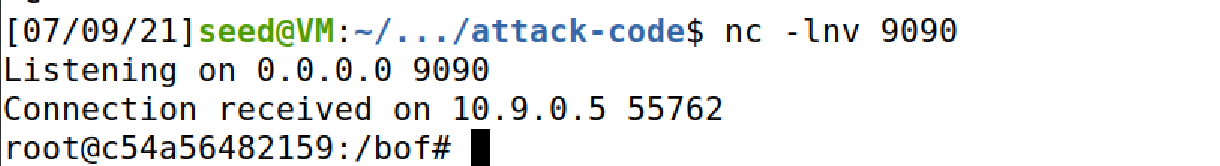
在attack-code中运行exploit.py，生成badfile，将badfile传入10.9.0.5，server端运行stack程序，读入badfile，函数返回地址越界，运行恶意代码，输出passwd;



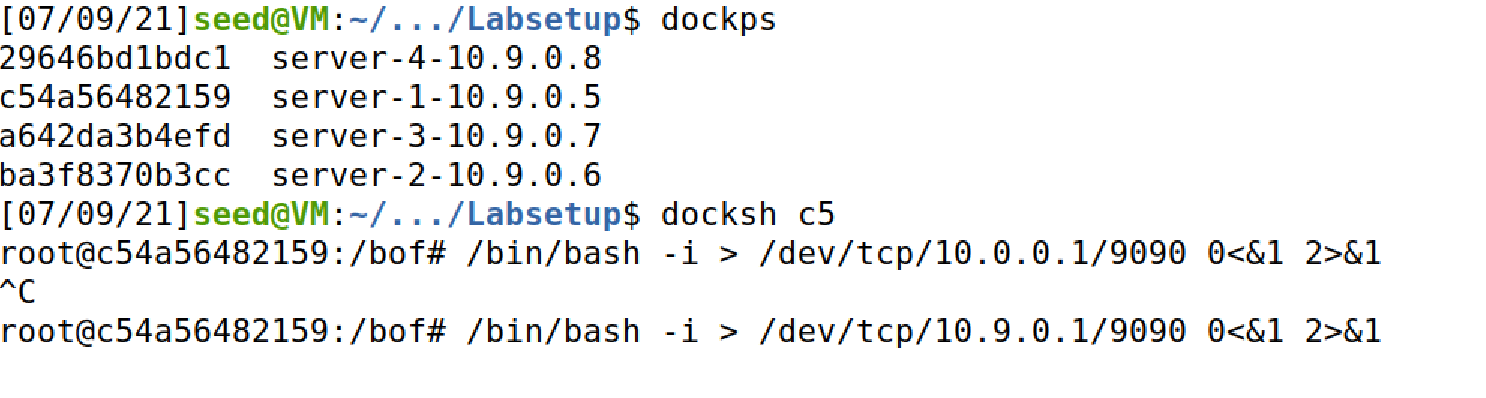
**Reverse shell**

拿到server的root shell并交互；

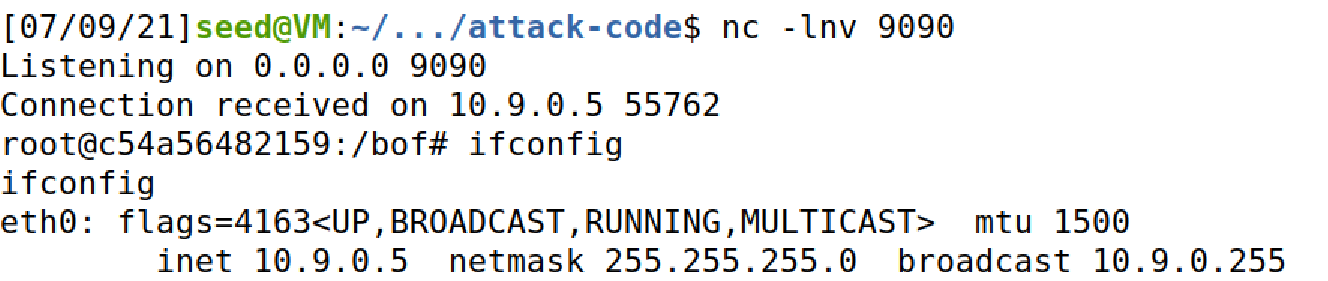
客户端打开9090端口，作为server接受tcp链接，监听9090端口；



服务器端向客户端10.9.0.1的9090端口发起tcp链接，输出重定向；

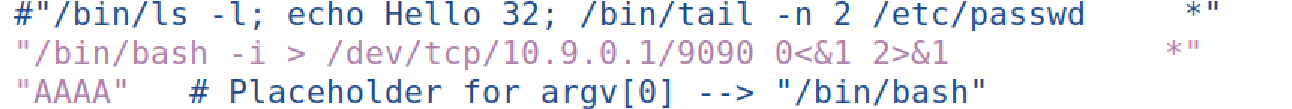
可以看到提示符已经为服务器端，输入ifconfig，可以看到输入被重定向，ip地址确实为服务器ip地址。

2>&1服务器端输出重定向到客户端；0<&1本地输入重定向到服务器端；

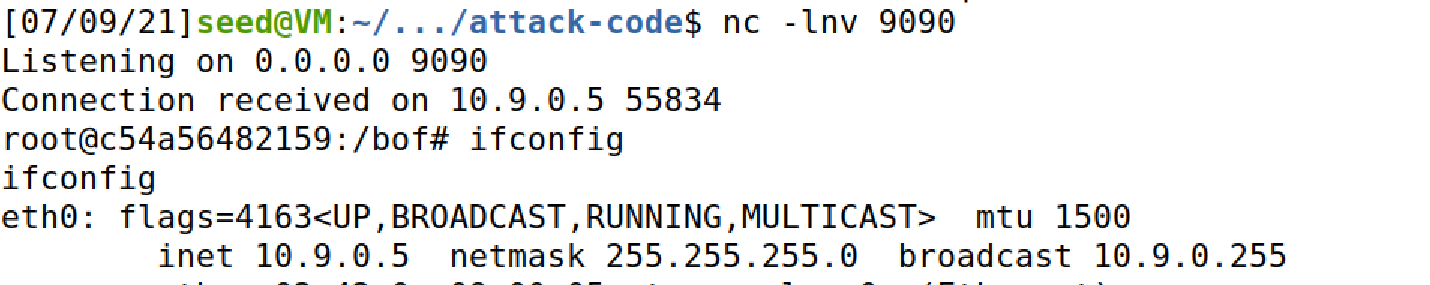


使用attack-code实现：

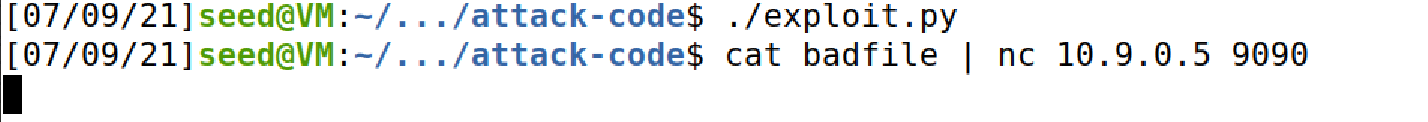
修改shell-code:



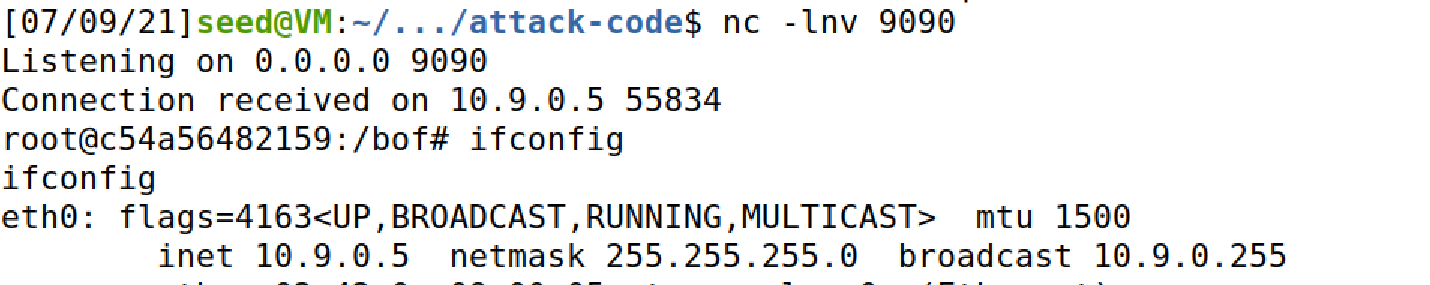
客户端Terminal 1打开监听端口9090



客户端Terminal 2 运行攻击程序

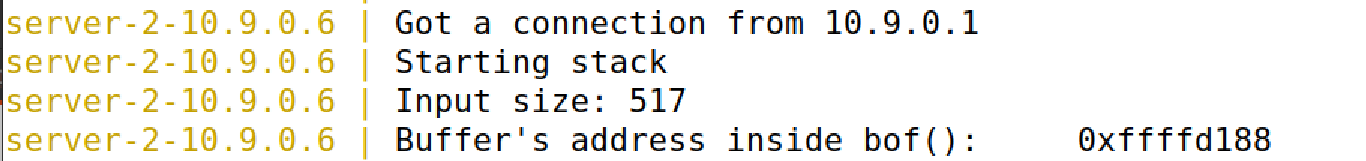


Terminal 1回显服务器shell。

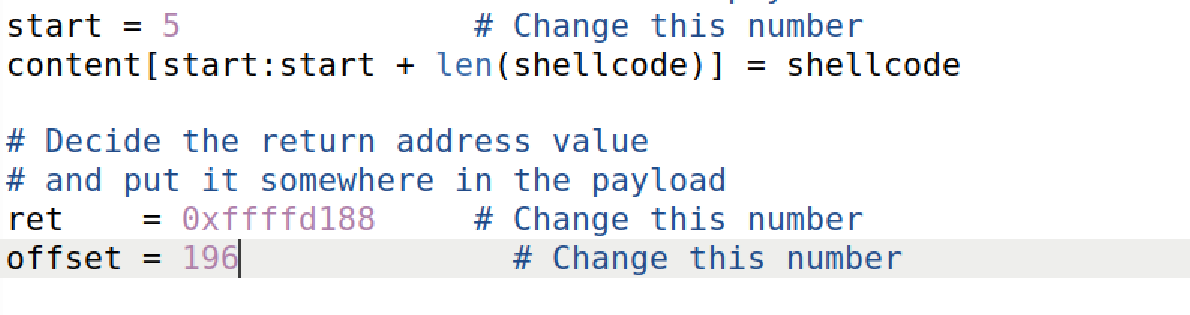


**Task 3：**

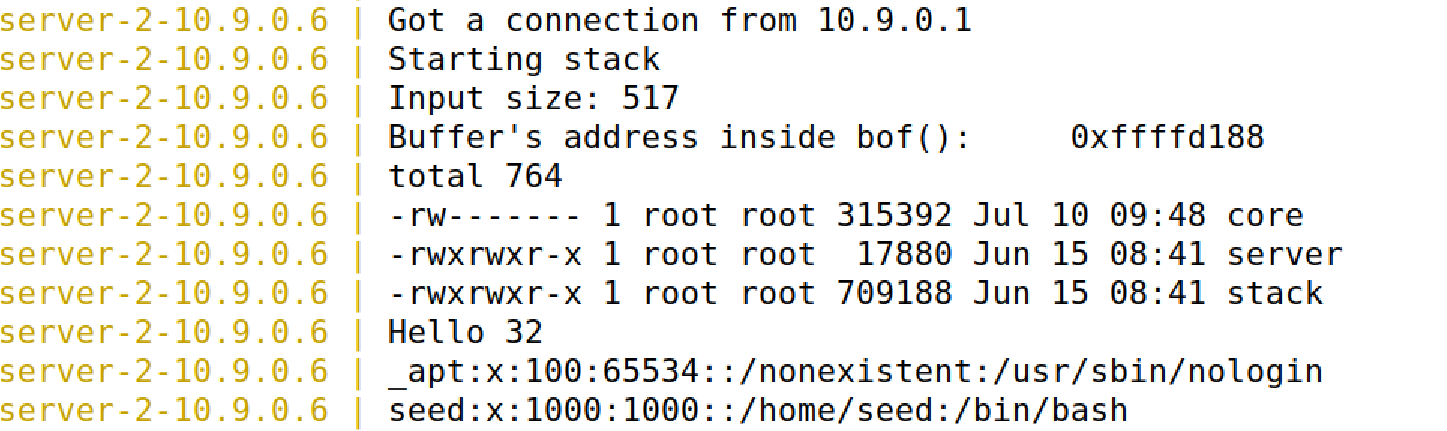
首先获得buffer基地址



将shellcode插入buffer前端，ret设为buffer首地址，这样只需每次修改offset即可试探原先return address；



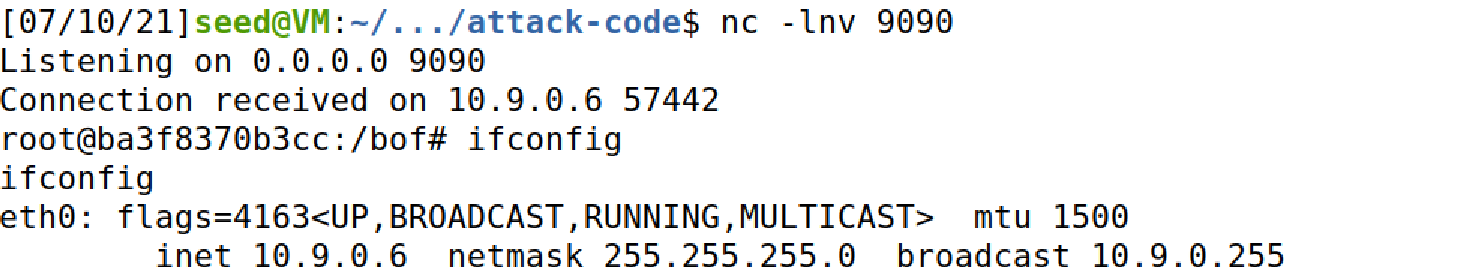
从160开始试探，每次加4，到196时成功；



**Reverse shell:**

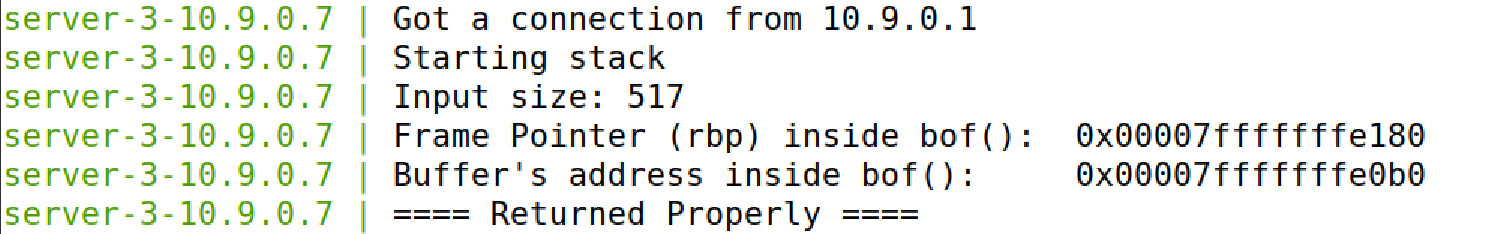
与attack 1步骤完全一致；

Terminal 1获得server2 root shell，通过ifconfig确认；

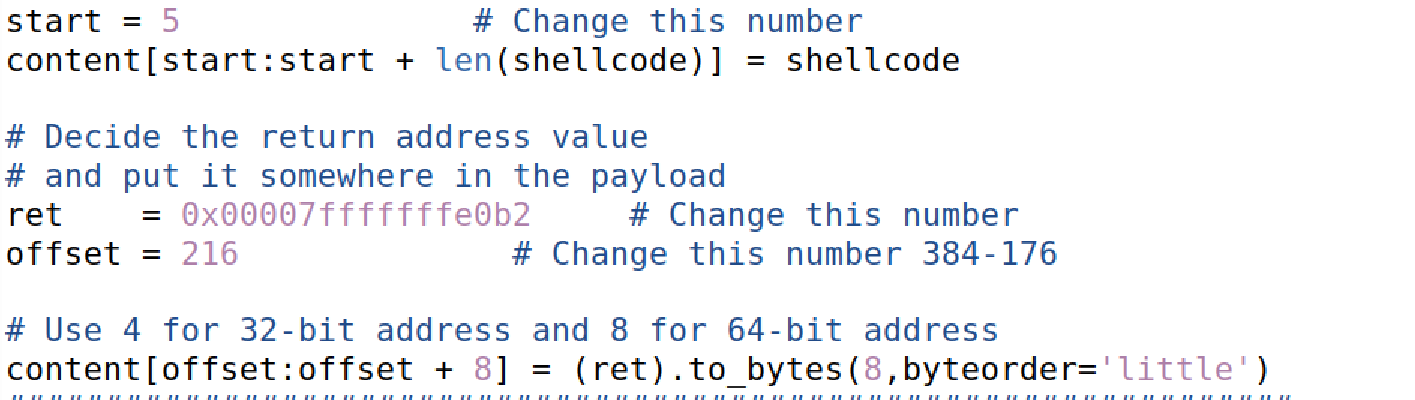


**Task 4 :**

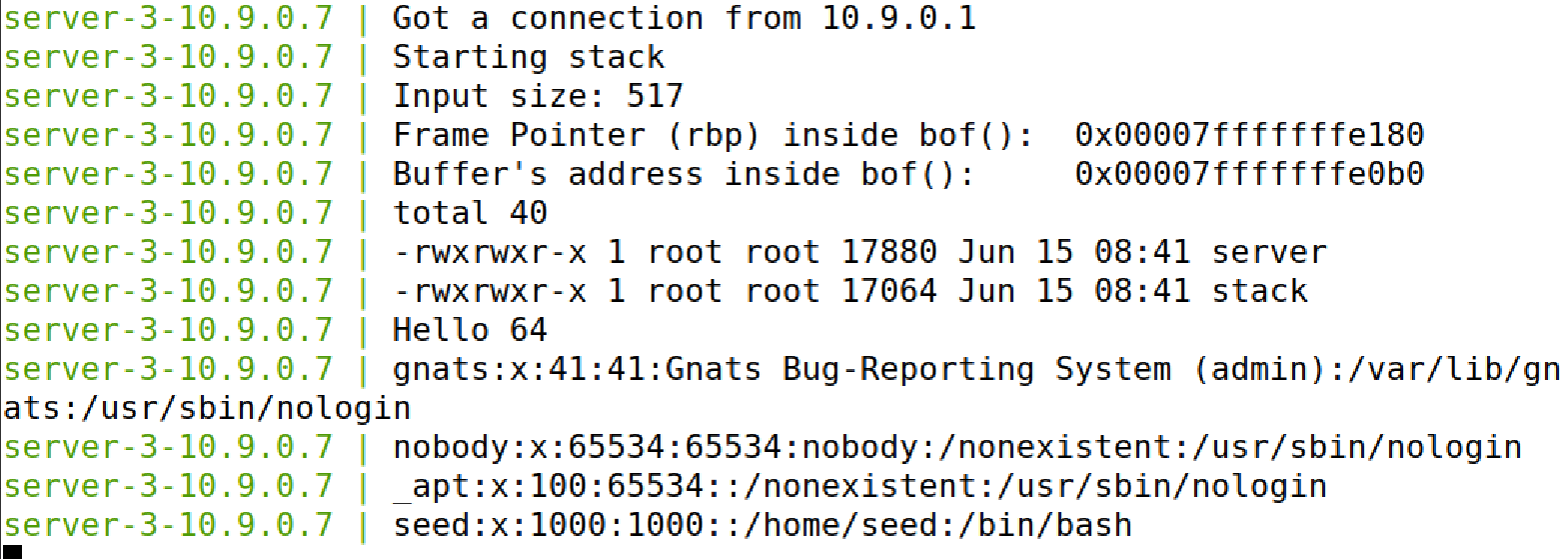
得到buffer和rbp地址:



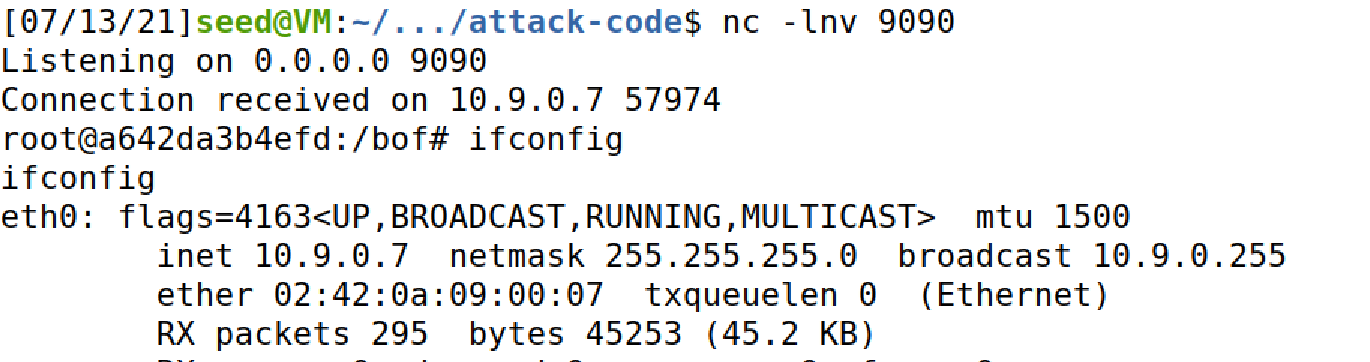
将攻击代码插入buffer前端，由于地址为小端存储，因此有效位放在低地址，字符串复制遇0x00终止不影响最终地址；将return地址指向buffer首地址，即可运行攻击指令；



运行结果：



**Revershell：**



**实验总结：**

本次实验主要利用栈溢出，用攻击代码重写return address，使返回地址重定向到写入的buffer或溢出栈的shellcode之前，需要打开execstack，使之可以执行栈中攻击代码。