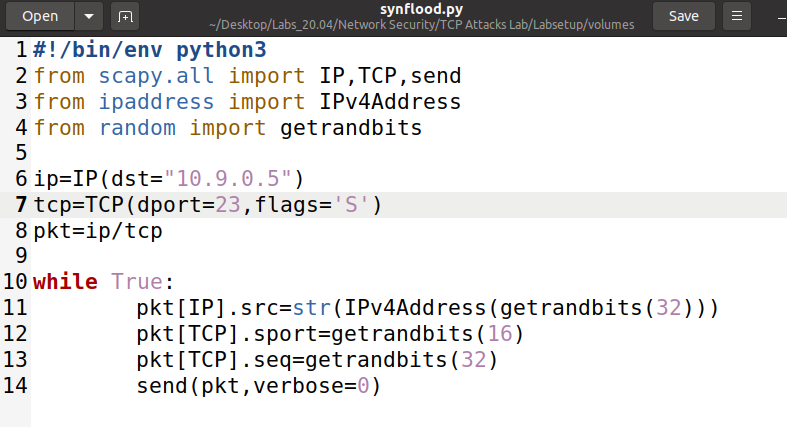
**TCP/IP Attack Lab**

**57118113 蔡义涵**

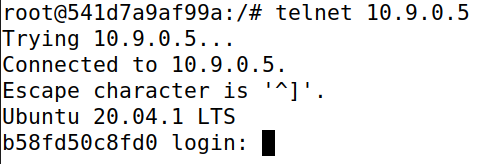
**Task 1: SYN Flooding Attack**

**Task 1.1: Launching the Attack Using Python**

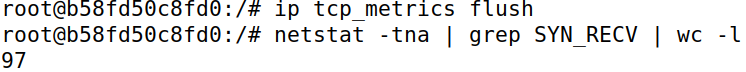
首先，完成python程序synflood.py，设置victim为10.9.0.5，端口为23。



使用seed-attacker10.9.0.1执行上述程序，一分钟后在user1-10.9.0.6上对10.9.0.5发起telnet请求。

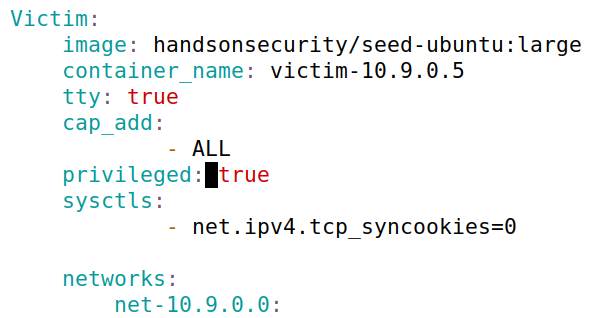


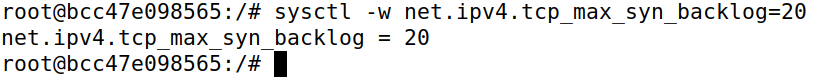
发现telnet请求响应时间较长，但仍然成功。采用ip tcp\_metrics flush命令清除tcp连接缓存，同时使用netstat -tna | grep SYN\_RECV | wc -l命令查看半连接队列长度为97，使用sysctl net.ipv4.tcp\_synack\_retries命令查看SYN+ACK报文重传次数为5。

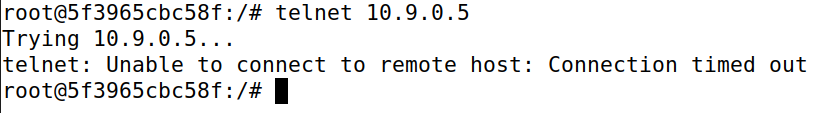




关闭docker，设置docker-compose.yml文件，为victim添加privileged:true项。设置半连接队列长度为20，同时运行两个synflood.py，再次执行telnet，发现连接失败。







**Task 1.2: Launch the Attack Using C**

首先查看SYN队列最大值为128

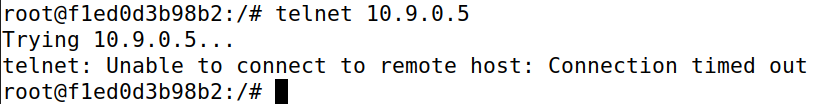


首先使用seed-attacker10.9.0.1上执行synflood.c

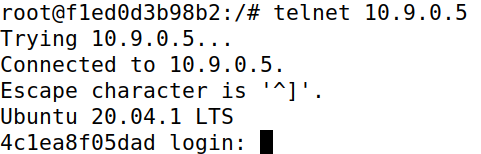




一分钟后在user1-10.9.0.6上执行对victim-10.9.0.5的telnet请求：



发现telnet连接失败，停止synflood程序后，再次执行telnet请求，发现成功



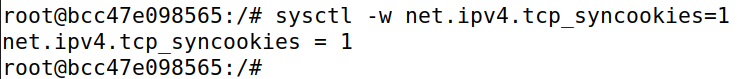
**总结**：synflood.c的攻击效果比synflood.py要好很多，即使是半开放序列长度为初始值，只在一个终端内运行synflood.c，也可以达到攻击效果。而synflood.py则需要自行降低半开放序列长度，并在多个终端运行synflood.py。

C语言程序比python语言程序执行效果好，推测有下面几个原因：

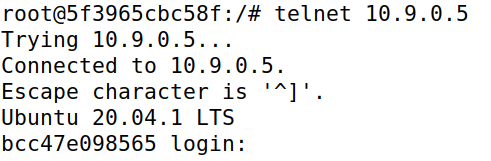
1. C语言代码执行效率本身比较高，这是C语言的固有优势，但写起来比python复杂得多；
2. C语言代码有1500字节的负载，可以加重目标机器的资源消耗。

**Task 1.3: Enable the SYN Cookie Countermeasure**

启动SYN Cookie机制，重复上述攻击，发现攻击失效。

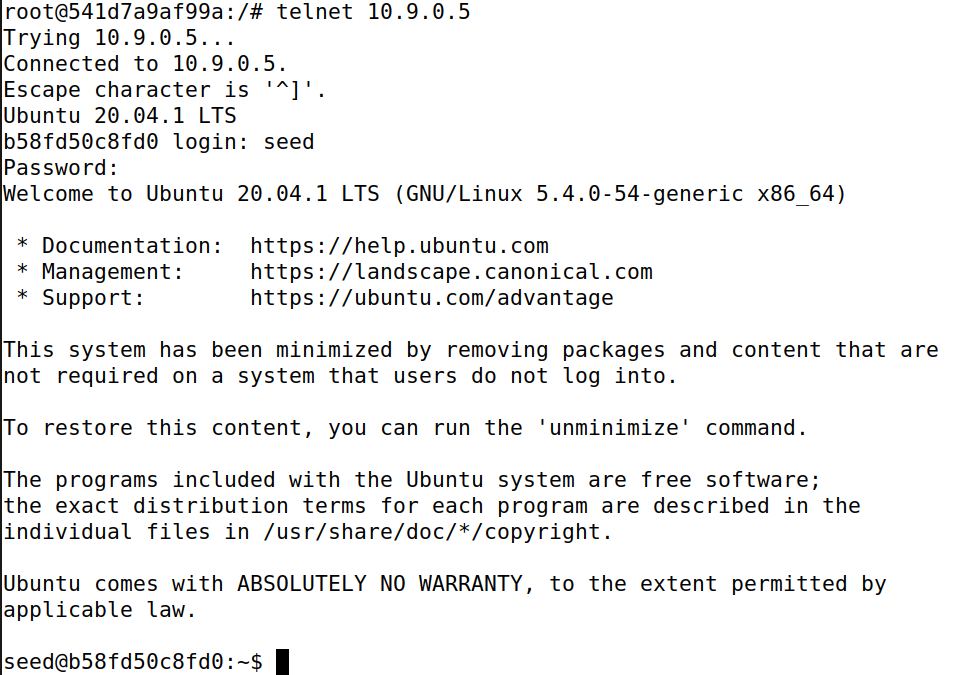




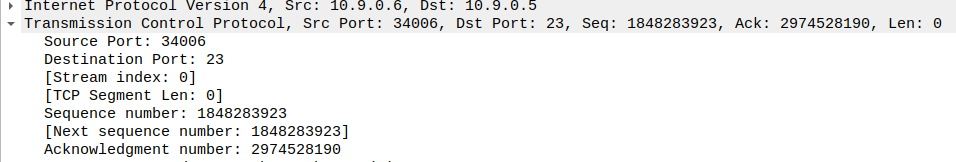


**Task 2: TCP RST Attacks on telnet Connections**

首先打开wireshark抓取br-b7f0fad2feff网卡的tcp流量信息，随后由user1-10.9.0.6对victim10.9.0.5发起telnet请求，并登录。



同时查看wireshark抓取到的最新的tcp连接报文。



利用以上信息填写tcp-rst.py程序，并在10.9.0.1上执行。



得到结果如下：

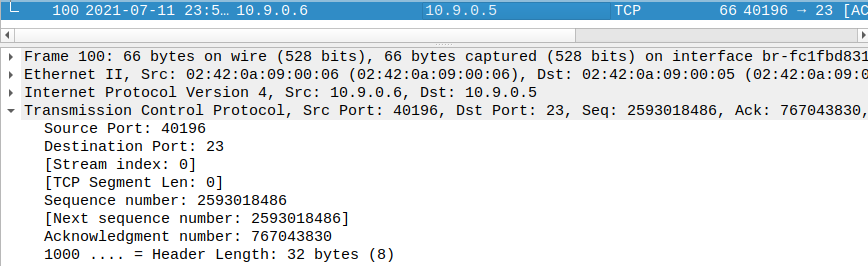


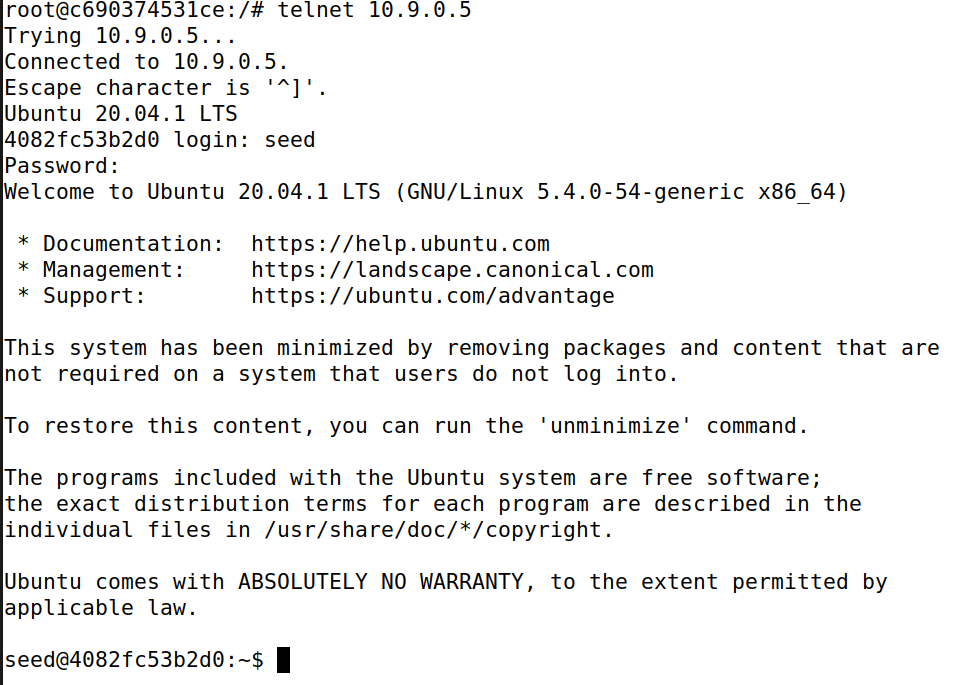
可以看到telnet连接被终止，wireshark新抓到以下包：



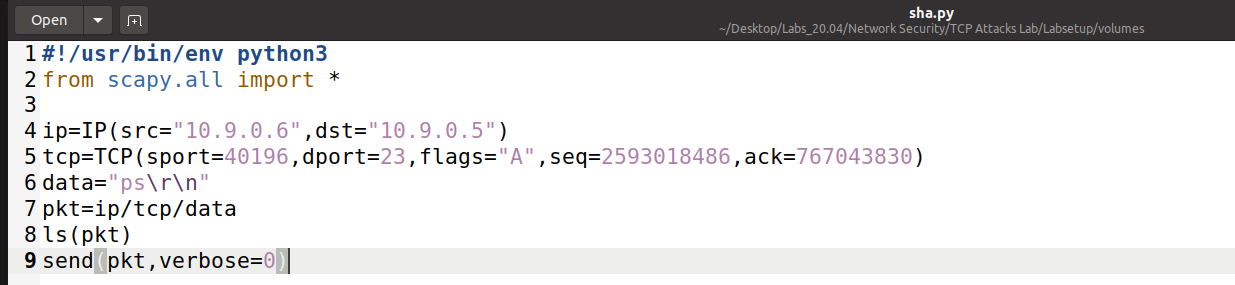
**Task 3: TCP Session Hijacking**

首先打开wireshark抓取br-fc1fbd83180d网卡上的tcp报文信息。同时使用user1-10.9.0.6执行对victim-10.9.0.5的telnet。

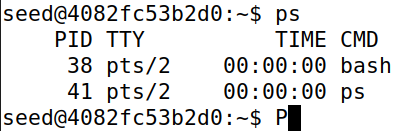




根据抓取到的报文完成sha.py程序，在attacker-10.9.0.1中执行sha.py。

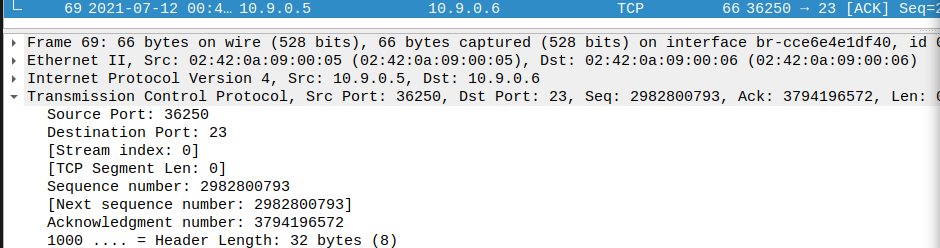


由于telnet会回传发送的命令进行确认，使得A上出现了ps命令的结果。

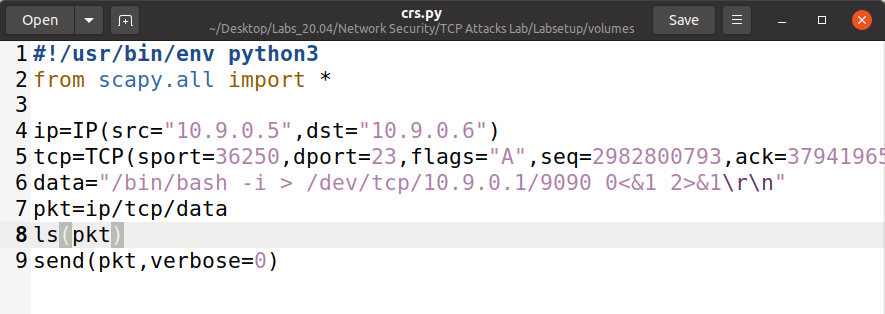


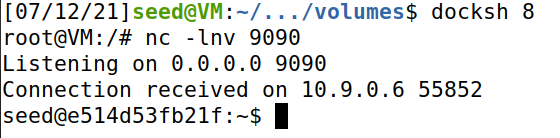
**Task 4: Creating Reverse Shell using TCP Session Hijacking**

我们使用victim-10.9.0.5对user1-10.9.0.6执行远程telnet，同时用wireshark进行抓包，观察最近一条tcp报文的内容。



根据最新的报文完成crs.py，在10.9.0.1上执行nc -lnv 9090，并在另一个终端上执行crs.py。





可以看到，反向shell成功。