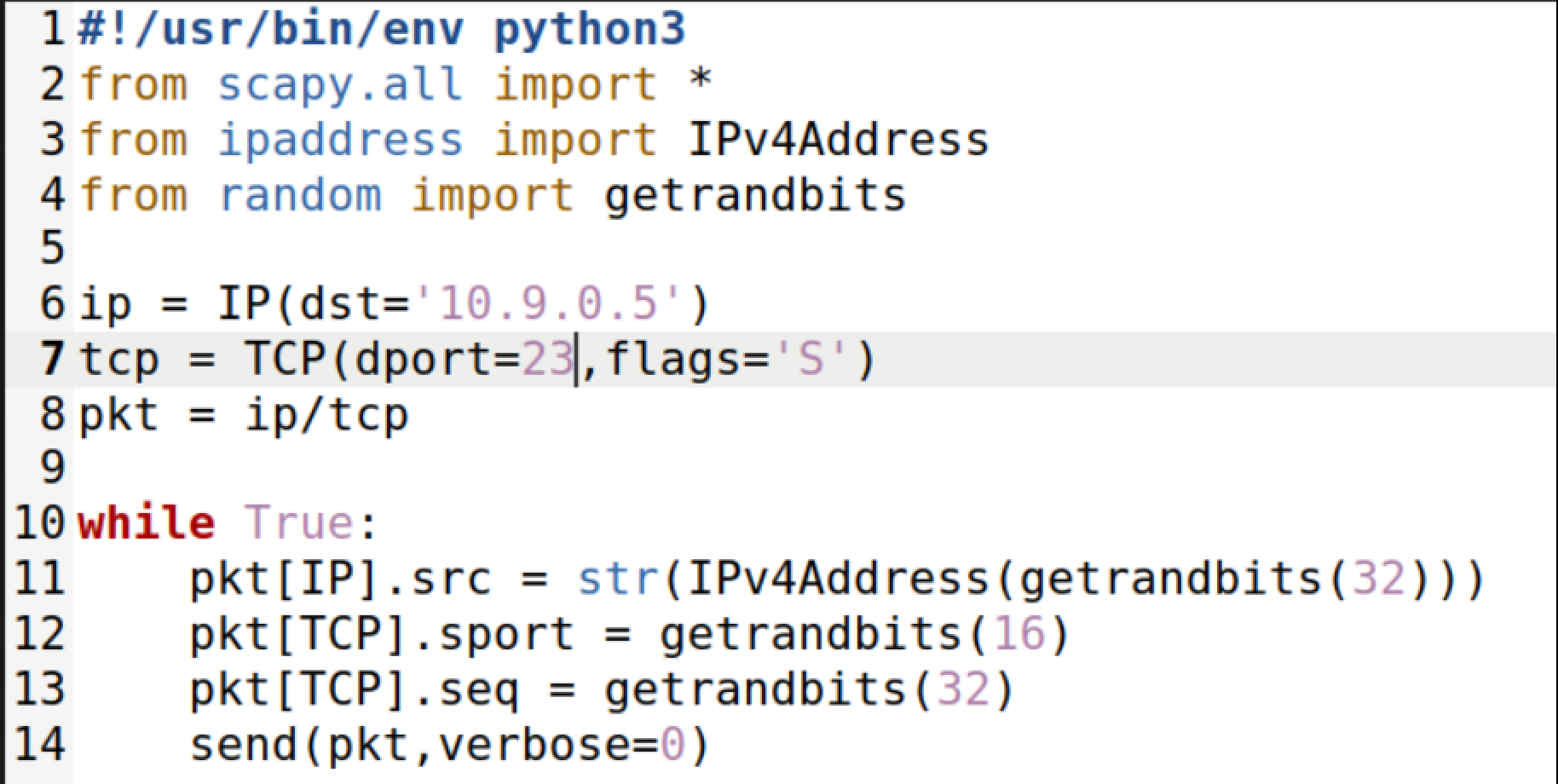
lab[2]-report

——57118207魏宇萌

**Task 1.1**

（1）实验内容：用python实现syn flood攻击。

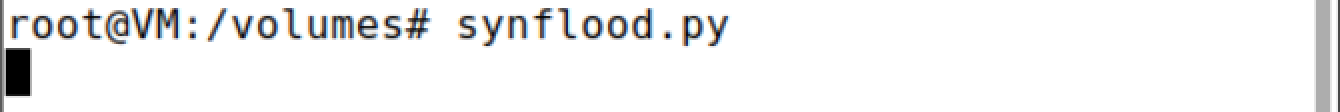
（2）代码：



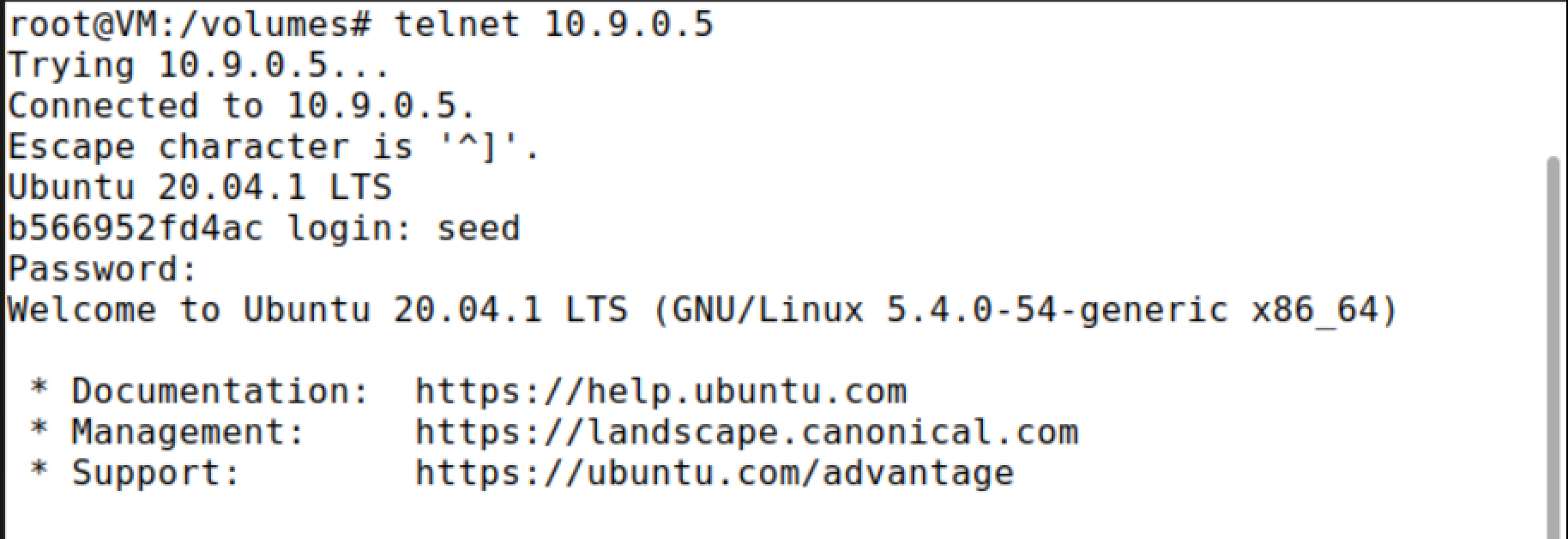
synflood.py

（3）程序执行情况：

（3.1）首先执行代码，再telnet victim主机10.9.0.5，发现可以telnet victim主机，syn flood攻击失败。



首次执行synflood.py

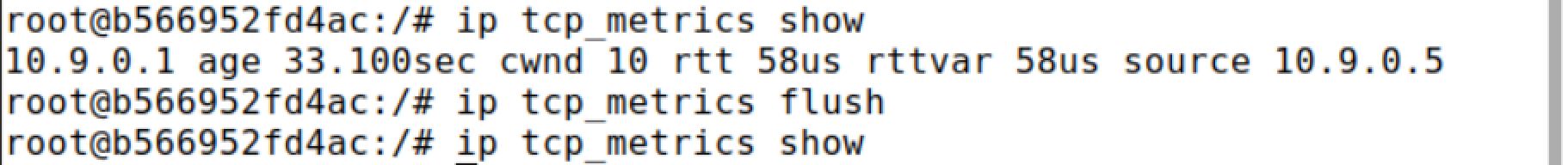


首次执行synflood.py后syn flood攻击失败

（3.2）根据可能使syn flood攻击失败的原因，重新展开攻击。首先检查syn cookie防御措施被关闭；其次由于TCP请求队列固定有四分之一的空间留给曾经建立过的连接，所以删除曾经建立过的连接记录；然后由于TCP请求队列对半连接的请求有重传确认机制，所以加大syn flood攻击的密度，同时执行三个synflood.py实例。

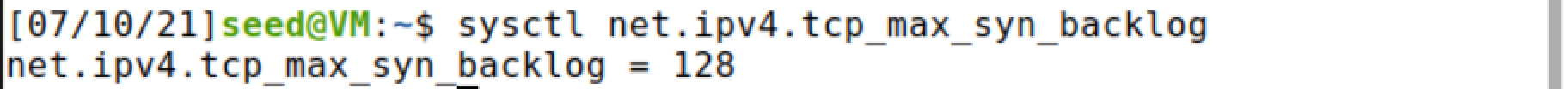


syn cookie防御措施被关闭

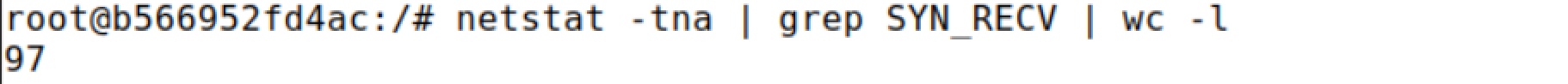


删除victim主机队列中曾经建立过的连接记录

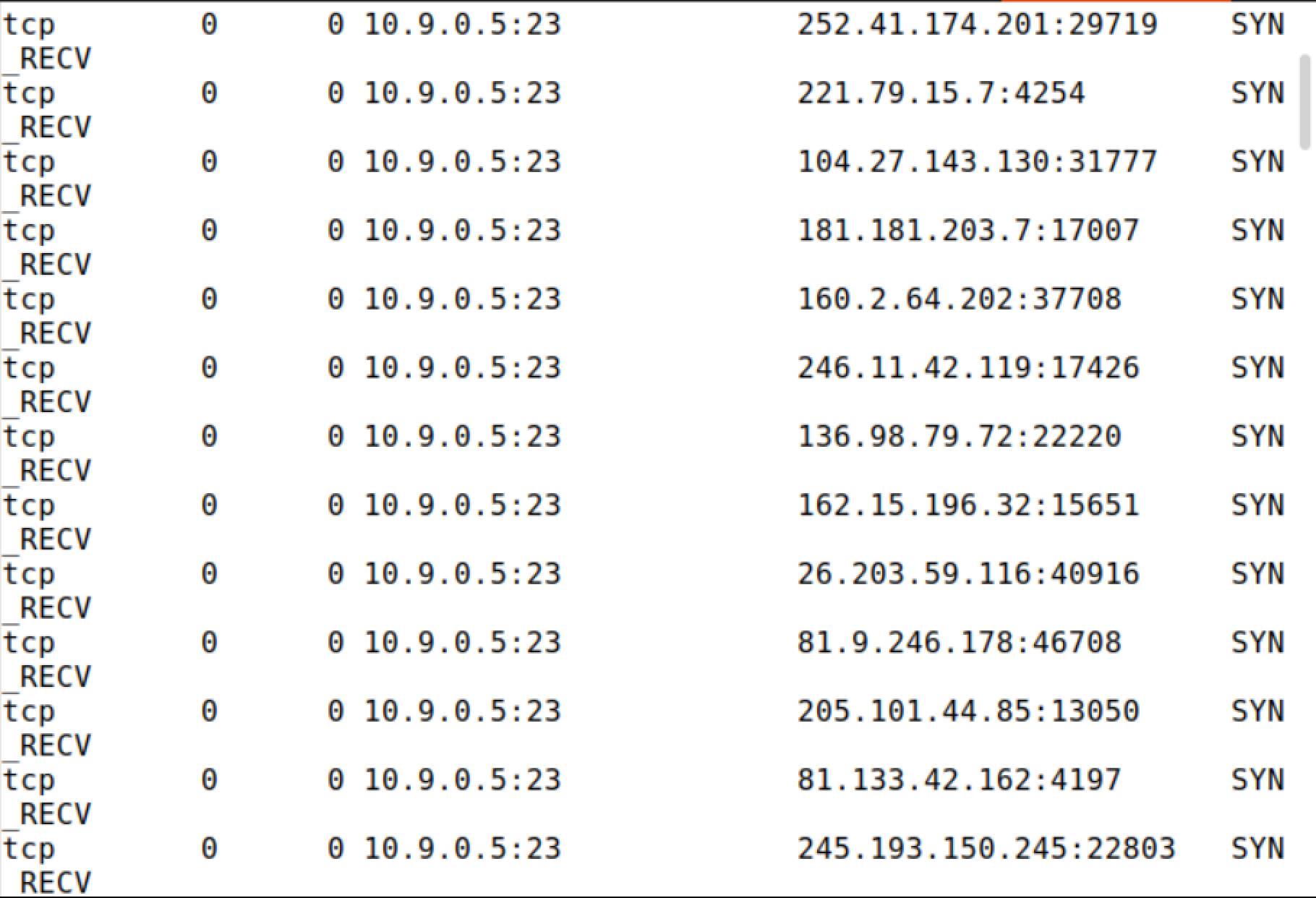
三个程序实例执行后，检查victim主机队列情况，发现当前处于半连接的请求为97个，根据victim主机队列大小128可知，其四分之三的大小用于新连接，即 96，所以当前队列中用于新连接建立的队列已满，此时telnet victim主机10.9.0.5发现则不能顺利建立连接，证明syn flood攻击成功。



TCP请求队列大小



victim主机队列中半连接的个数



victim主机TCP队列情况



syn flood攻击成功

**Task 1.2**

（1）实验内容：用C实现syn flood攻击。

（2）代码：由实验提供的synflood.c

（3）程序执行情况：将synflood.c编译后执行，再telnet victim主机10.0.9.5，发现无法telenet到victim主机，证明syn flood攻击成攻。这说明一个synflood.c实例的攻击密度高于一个synflood.py实例的攻击密度，这是因为C语言是编译语言，直接执行机器码，python语言是解释语言，只有执行到一句话时才将该句话转化为机器码执行，所以C程序的执行速度上更快从而实现的攻击密度更高。



syn flood攻击成功

**Task 1.3**

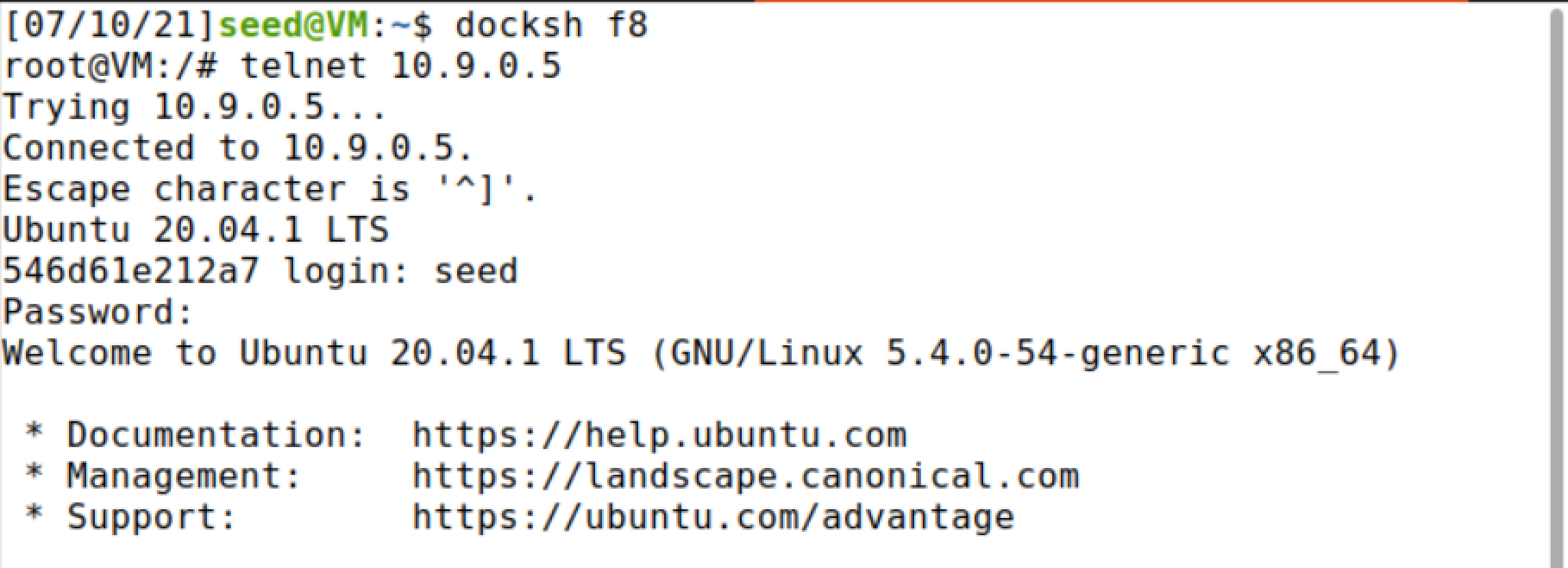
（1）实验内容：打开syn cookies防御措施，再次尝试syn flood攻击。

（2）代码：由于synflood.c的攻击效果好于synflood.py的攻击效果，所以使用由实验义工的synflood.c

（3）程序执行情况：打开syn cookies防御措施，重新dcup实验环境，执行程序，发现可以telnet victim主机，证明syn flood攻击失败，防御措施对syn flood确实有效。

****

syn cookies防御措施被打开

****

syn flood攻击失败

**Task 2**

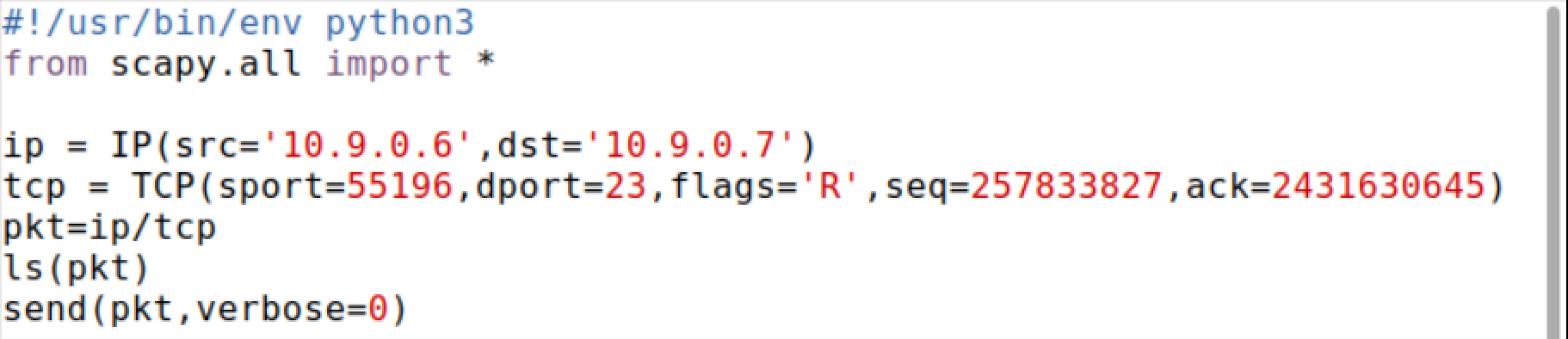
（1）实验内容：实现TCP RST攻击。

（2）代码与程序执行情况：以telnet为例，telnet是使用TCP协议的应用，先在两个victim主机10.9.0.6和10.9.0.7之间建立telnet连接，通过Wireshark观察最后的报文。

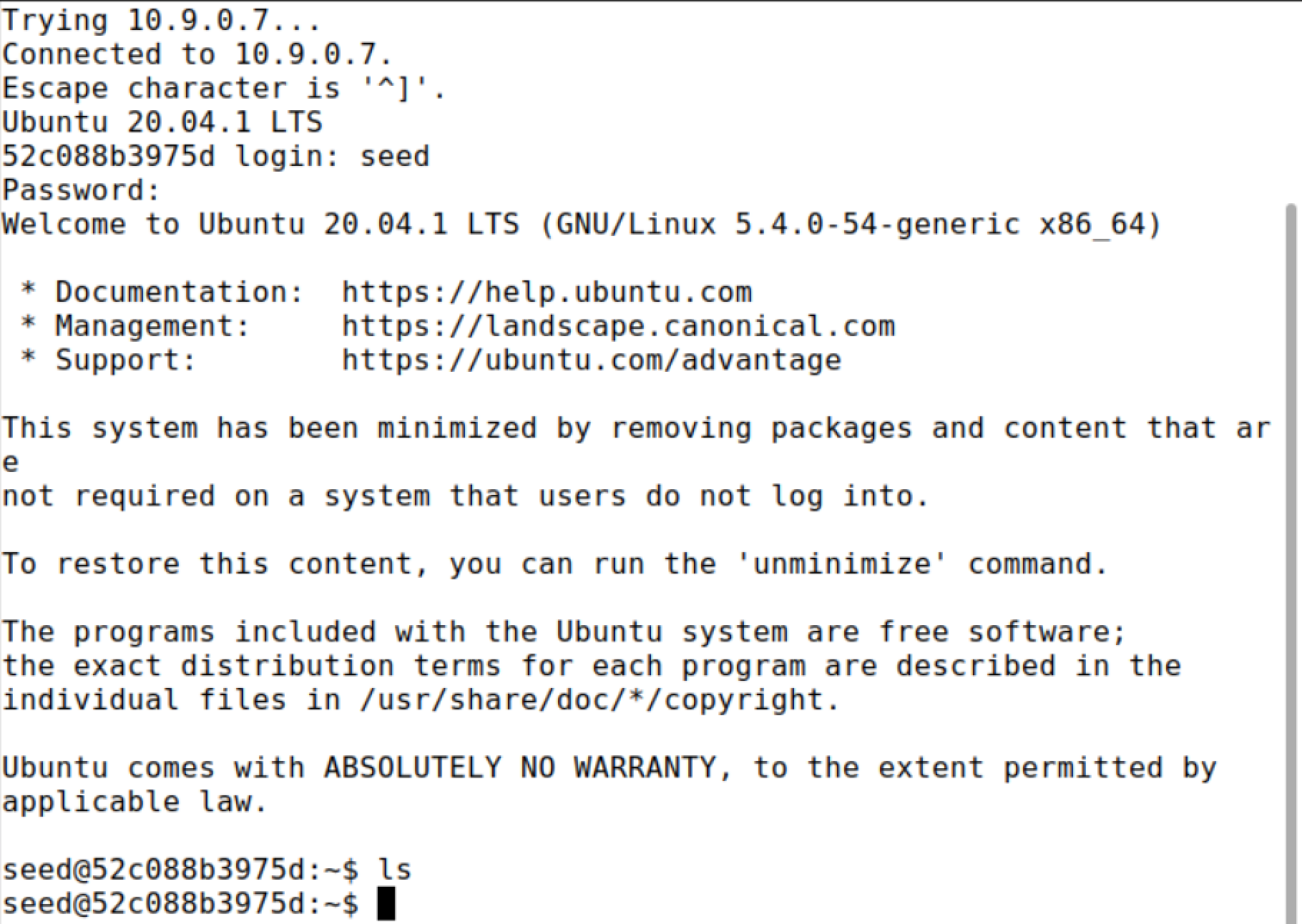
（2.1）伪造telnet客户10.9.0.6发出的RST报文，观察到源端口号为55196，下一个报文的序列号应为257833827，ACK应为2431630645。执行程序，发现10.9.0.6客户并不知道连接已断开，但是当10.9.0.6客户再想给10.9.0.7服务器发送消息时，10.9.0.7服务器会向10.9.0.6客户发送RST报文通知连接已断开，此时10.9.0.6客户的连接显示中断。



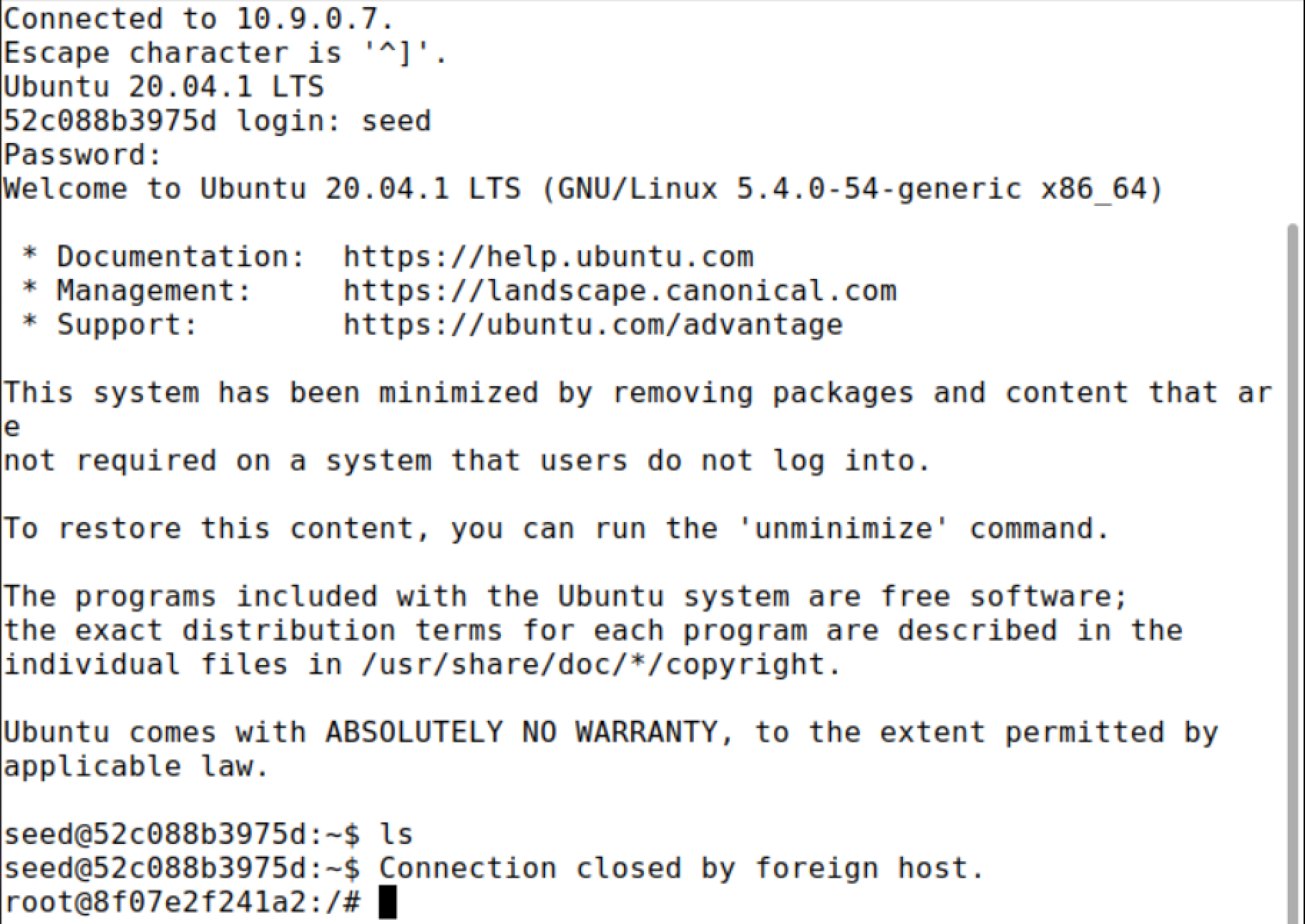
Wireshark嗅探到的最后的报文



伪造telnet客户发送RST报文

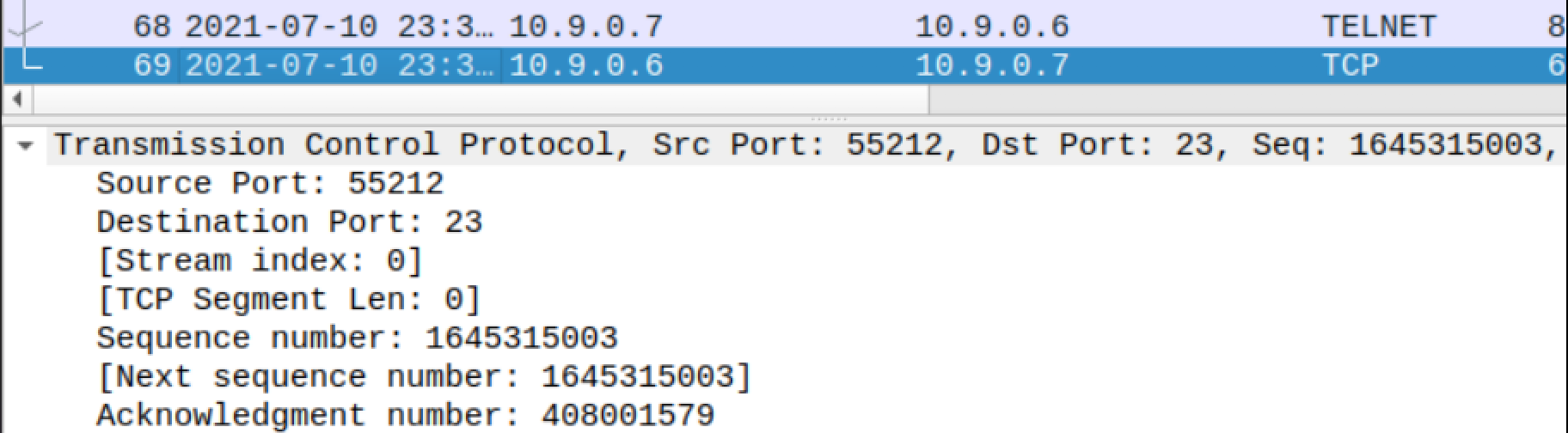


程序执行前，telnet连接保持中

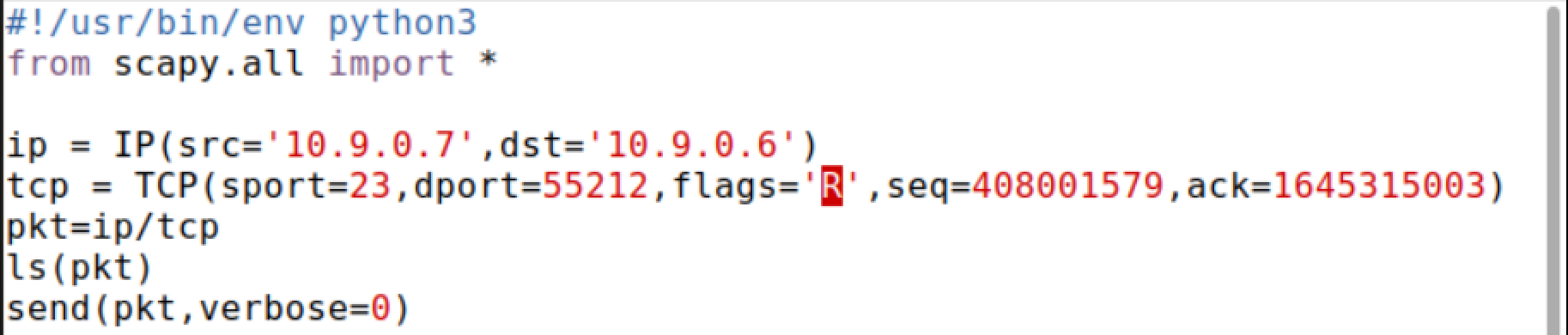


TCP RST攻击成攻，telnet连接中断

（2.2）伪造telnet服务器10.9.0.7发出的RST报文，观察到目的端口号应为55212，下一个报文的序列号应为408001579，ACK应为1645315003。发现10.9.0.6客户立刻断开了连接，TCP RST攻击成功。



Wireshark嗅探到的最后的报文



伪造telnet服务器发送RST报文

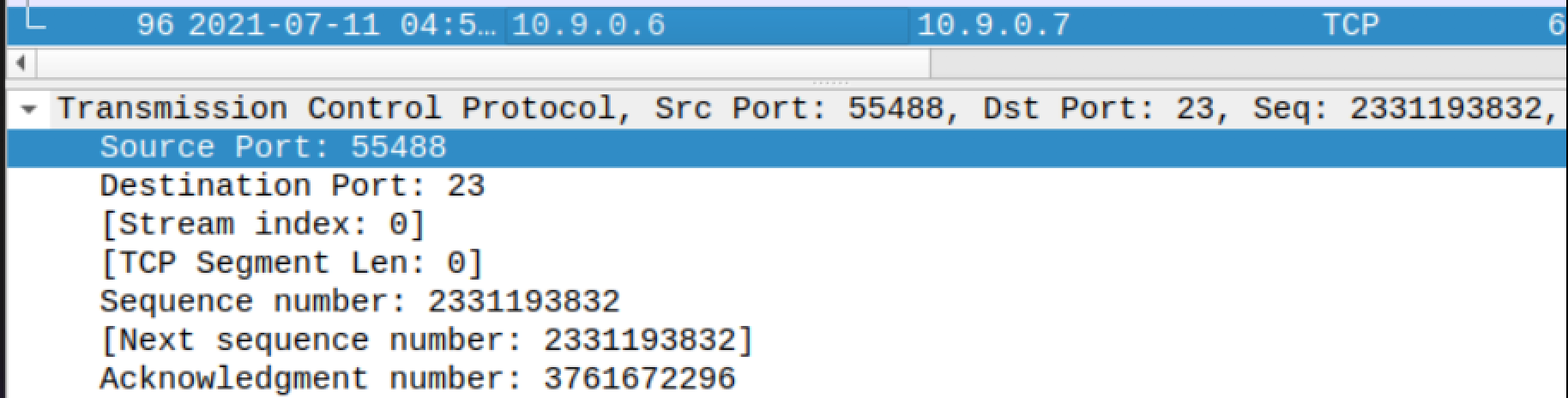


TCP RST攻击成攻，telnet连接中断

**Task 3**

（1）实验内容：实现TCP劫持攻击

（2）代码：在客户10.9.0.6与服务器10.9.0.7建立telnet连接，通过Wireshark观察最后一个报文，则劫持报文的源端口号应为55488，序列号应为2331193832，ACK应为3761672296。TCP劫持攻击需有恶意的数据负载，本实验中执行一个恶意指令，输出秘密文件内容。

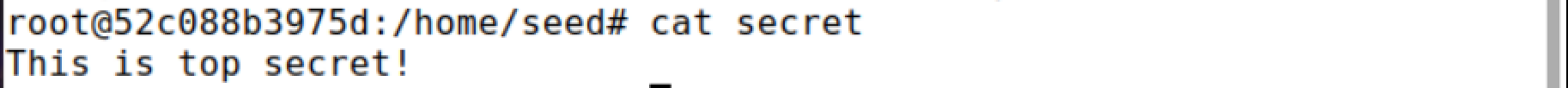


Wireshark嗅探到的最后的报文

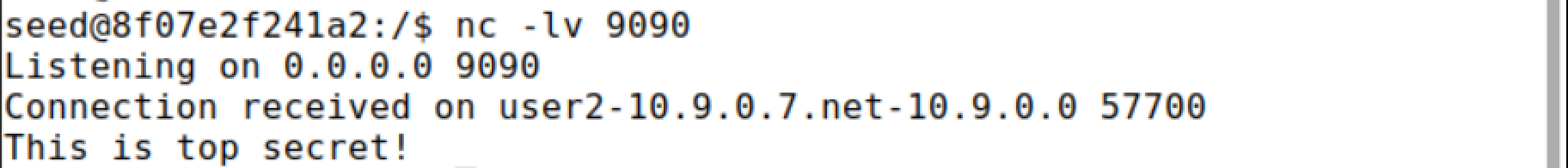


hijack.py

（3）程序执行情况：先在telnet服务器创建一个秘密文件secret用于劫持攻击的目标；再在attacker中seed用户下监听9090端口；最后执行hijack.py。发现监听的9090端口输出秘密文件中的内容，TCP劫持攻击成功。



秘密文件secret内容

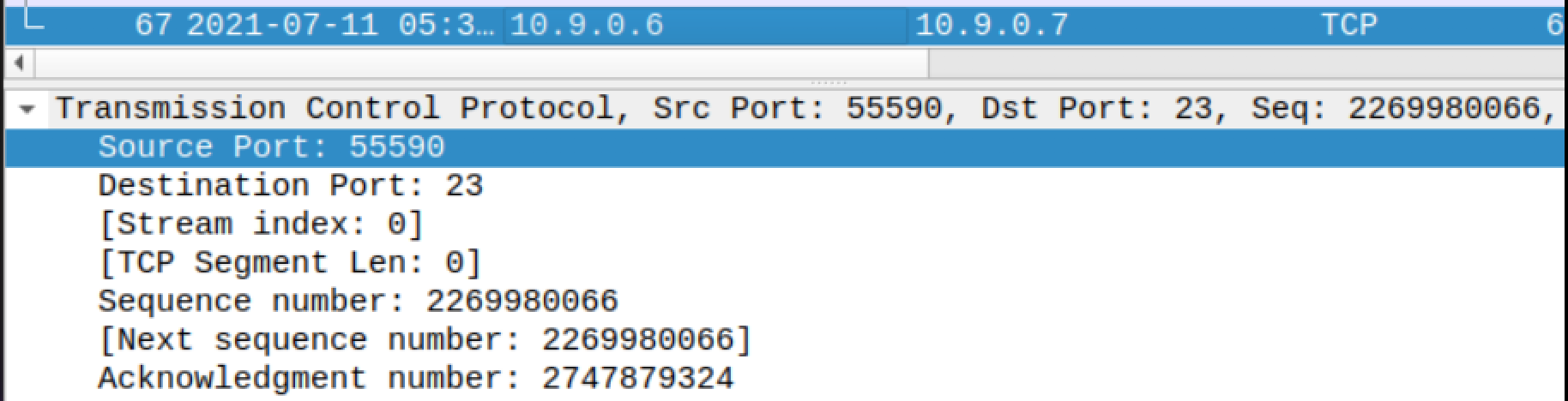


TCP劫持攻击成功

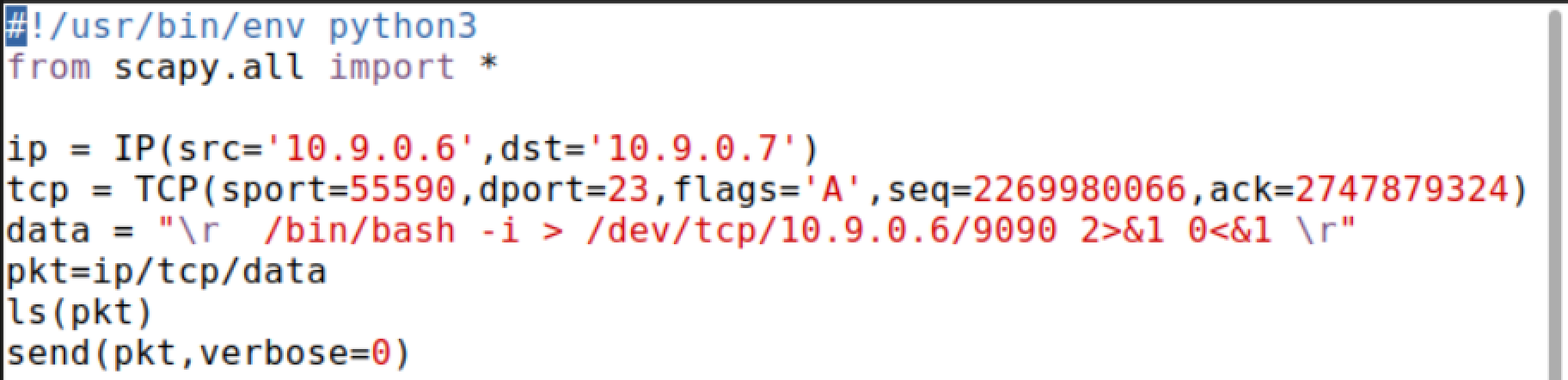
**Task 4**

（1）实验内容：通过TCP劫持攻击获得反向shell。

（2）代码：在客户10.9.0.6与服务器10.9.0.7建立telnet连接，通过Wireshark观察最后一个报文，则劫持报文的源端口号应为55590，序列号应为2269980066，ACK应为2747879324。

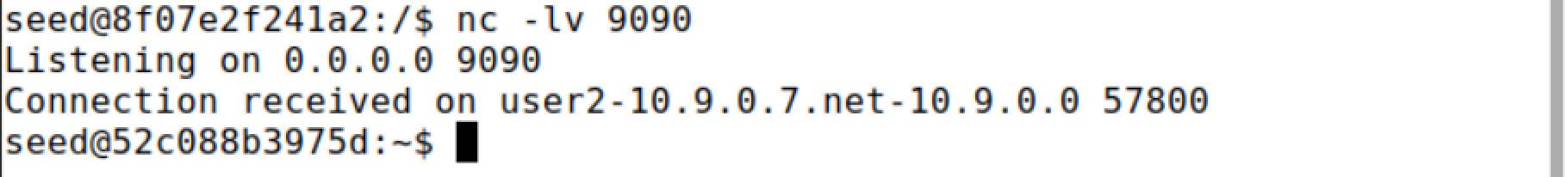


Wireshark嗅探到的最后的报文



hijack1.py

（3）程序执行情况：在attacker监听9090端口，再执行程序，发现attacker成功获得telnet服务器的shell，TCP劫持攻击获得反向shell成功。



TCP劫持攻击成功，获得反向shell