

TCP RESET ATTACK ON SSH

Kazi Sajeed Mehrab

Student ID – 1505025

TCP RESET ATTACK

TCP connections can be closed if the RST bit in the header of a TCP packet is set. RST is one of the six code bits in the TCP header. If PC A is connected to PC B, the connection can be closed by any one of the connected nodes sending a reset packet to the other one. This approach is mainly used in emergency situations when some errors are detected.

Since a single packet can close a TCP connection, this can be used as an attack mechanism to close an established connection between two nodes. In order to break up a TCP connection between A and B, the attacker needs to spoof a TCP RST packet from A to B or from B to A.

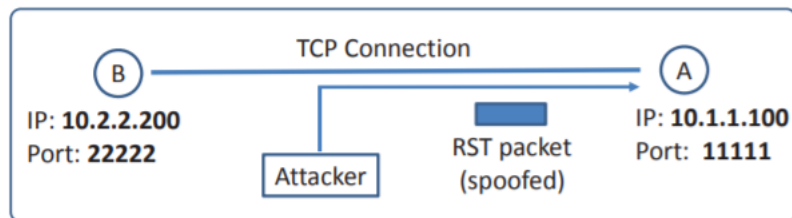


Figure: Attack Diagram of TCP Reset

The diagram above shows the attack diagram for TCP Reset for two PCs connected via a normal TCP connection.

The secure shell (SSH) protocol

The SSH protocol (also referred to as Secure Shell) is a method for secure remote login from one computer to another. It provides several alternative options for strong authentication, and it protects the communications security and integrity with strong encryption. However, the encryption is done in the transport layer, which means only the data in the TCP packets are encrypted. TCP Reset attack only needs to access the header of the packets, which are not encrypted. So the attack can be launched on an SSH connection.

The SSH protocol works in the client-server model. This means that the connection is established by the SSH client connecting to the SSH server. The SSH client drives the connection setup process and uses public key cryptography to verify the identity of the SSH server. After the setup phase the SSH protocol uses strong symmetric encryption and hashing algorithms to ensure the privacy and integrity of the data that is exchanged between the client and server.

The standard TCP port 22 has been assigned for contacting SSH servers.

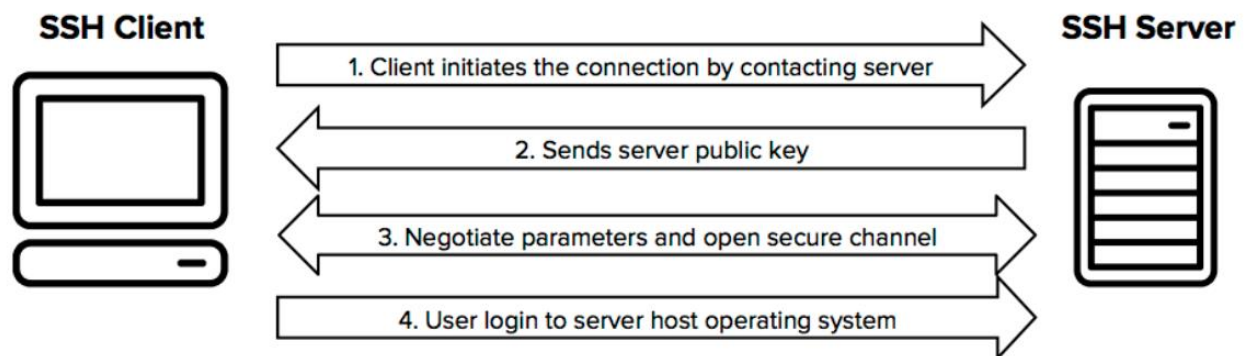


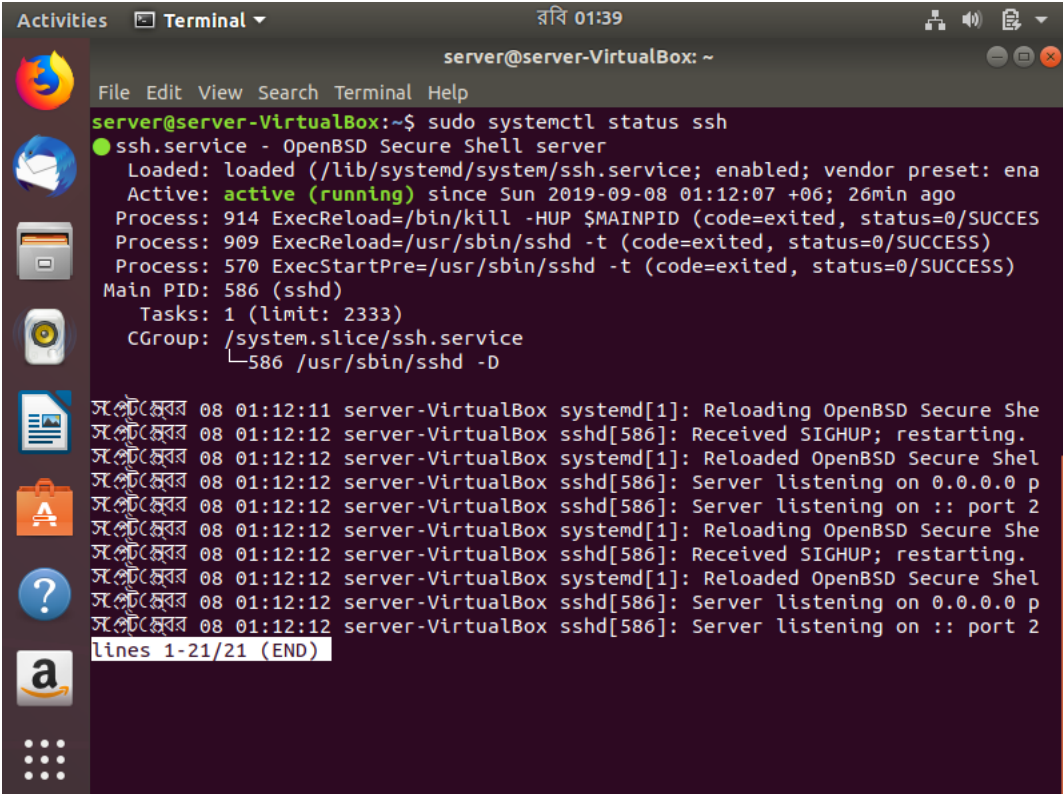
Figure: The SSH client-server model.

The figure above shows the steps by which a SSH client connects to an SSH server.

Steps of the attack

The steps that have been followed to demonstrate the attack are as follows:

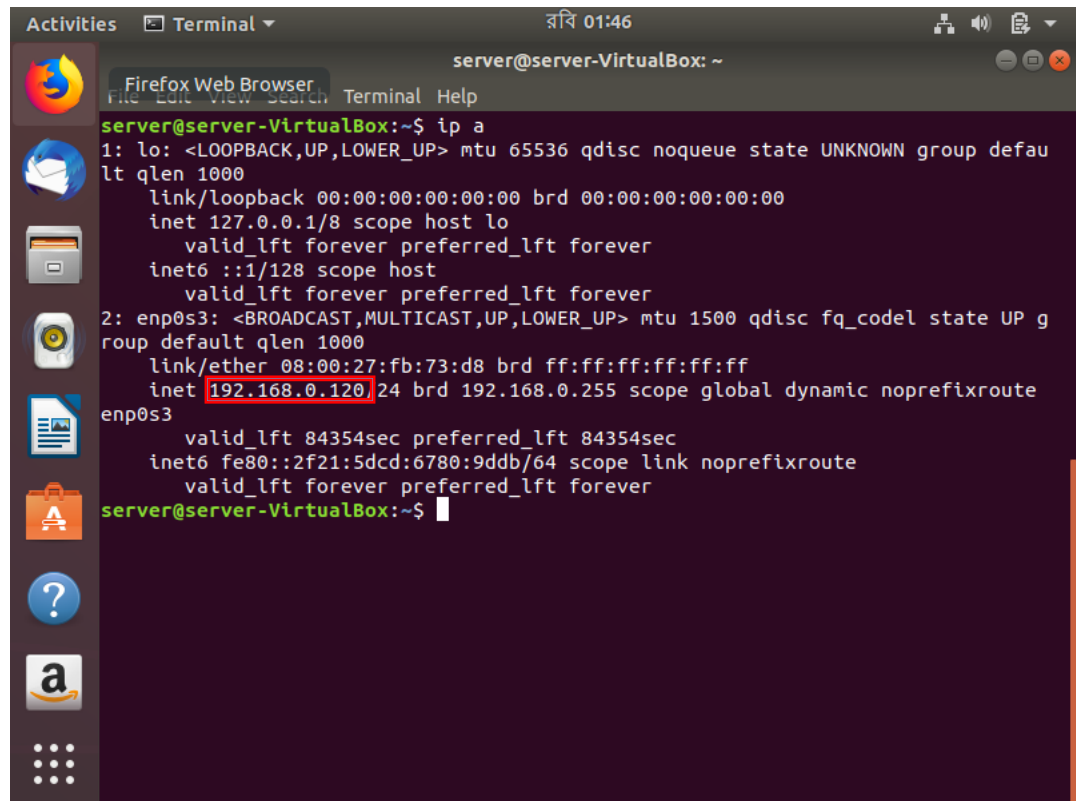
1. Three Ubuntu PCs have been set up using **Oracle VM VirtualBox**. The three PCs were connected to the same LAN via the same router. This was ensured by enabling **promiscuous mode** in all of the three machines. The three PCs have been referred to as **attacker**, **server** and **client** for the rest of this report.
2. An SSH connection have been set up between the **server** and the **client**. To do this from scratch, the following needs to be done:
 - a. From the **server** PC, SSH needs to be installed. After successful installation, the SSH server should be up and running. This can be checked by executing the following command from the **server**:



```
server@server-VirtualBox: ~  
server@server-VirtualBox:~$ sudo systemctl status ssh  
● ssh.service - OpenBSD Secure Shell server  
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: ena  
   Active: active (running) since Sun 2019-09-08 01:12:07 +06; 26min ago  
     Process: 914 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)  
     Process: 909 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)  
     Process: 570 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)  
    Main PID: 586 (sshd)  
      Tasks: 1 (limit: 2333)  
     CGroup: /system.slice/ssh.service  
             └─586 /usr/sbin/sshd -D  
  
08 01:12:11 server-VirtualBox systemd[1]: Reloading OpenBSD Secure She  
08 01:12:12 server-VirtualBox sshd[586]: Received SIGHUP; restarting.  
08 01:12:12 server-VirtualBox systemd[1]: Reloaded OpenBSD Secure Shel  
08 01:12:12 server-VirtualBox sshd[586]: Server listening on 0.0.0.0 p  
08 01:12:12 server-VirtualBox sshd[586]: Server listening on :: port 2  
08 01:12:12 server-VirtualBox systemd[1]: Reloading OpenBSD Secure She  
08 01:12:12 server-VirtualBox sshd[586]: Received SIGHUP; restarting.  
08 01:12:12 server-VirtualBox systemd[1]: Reloaded OpenBSD Secure Shel  
08 01:12:12 server-VirtualBox sshd[586]: Server listening on 0.0.0.0 p  
08 01:12:12 server-VirtualBox sshd[586]: Server listening on :: port 2  
lines 1-21/21 (END)
```

The **active (running)** indicates that the SSH server is running from the **server PC**.

- b. The **IP address of the server** needs to be known, as it will be needed by the **client** to connect to the **server**. This can be done by executing the following command from the **server**:



```
server@server-VirtualBox: ~  
server@server-VirtualBox:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau  
lt qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g  
roup default qlen 1000  
    link/ether 08:00:27:fb:73:d8 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.120/24 brd 192.168.0.255 scope global dynamic noprefixroute  
enp0s3  
        valid_lft 84354sec preferred_lft 84354sec  
    inet6 fe80::2f21:5dcd:6780:9ddb/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
server@server-VirtualBox:~$
```

The **IP address** of the server has been shown in a red frame in the above screenshot. As it can be seen, the **IP address** of the **server** is **192.168.0.120**.

- c. SSH clients can connect to SSH servers by using the following command:

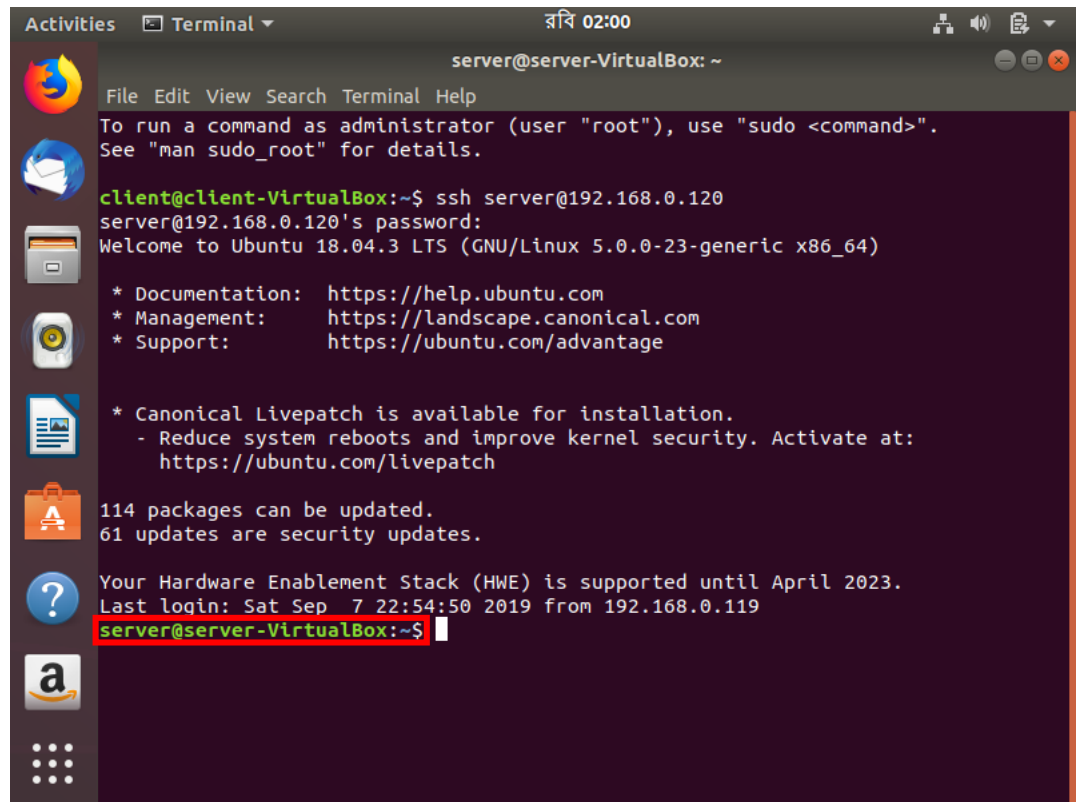
```
ssh server_name@server_ip_address
```

Here, server_name is the name of the server PC. In this case, as seen from the screenshots above, the name of the server PC is “server”. The server_ip_address is the **IP address** of the server PC. In our case, this was found out to be **192.168.0.120** in the previous step. From the **client PC**,

the following command needs to be executed to connect to the server that has been set up:

```
ssh server@192.168.0.120
```

The screenshot below shows the result of running this command from the **client PC**:

A screenshot of a terminal window on a client PC. The window title is "Terminal" and the time is "০২:০০". The terminal shows the command "ssh server@192.168.0.120" being executed. The prompt changes from "client@client-VirtualBox:~\$" to "server@192.168.0.120's password:". After the password is entered, the terminal displays the Ubuntu 18.04.3 LTS login banner, including documentation links, Canonical Livepatch information, and package update notifications. The final prompt is "server@server-VirtualBox:~\$", which is highlighted with a red rectangle to indicate a successful connection to the server's home directory.

```
Activities  Terminal  ০২:০০
server@server-VirtualBox: ~
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
client@client-VirtualBox:~$ ssh server@192.168.0.120
server@192.168.0.120's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

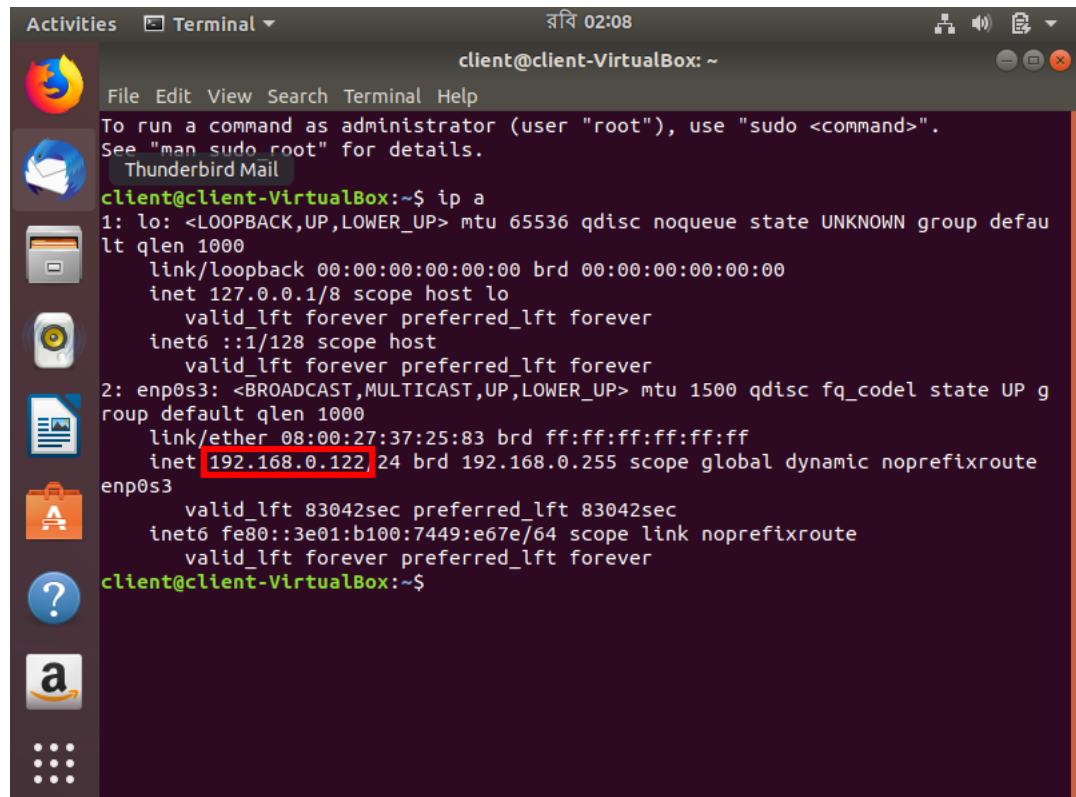
 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

114 packages can be updated.
61 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sat Sep  7 22:54:50 2019 from 192.168.0.119
server@server-VirtualBox:~$
```

The command will prompt the user to enter the **server PC's** password. Upon entering the correct password, a remote connection will be set from the **client PC** to the **server PC**. This can be observed by the fact that the directory in the **client's** terminal actually points to the **server's** home directory, as shown in the red frame. This shows that an **SSH connection has been successfully established**.

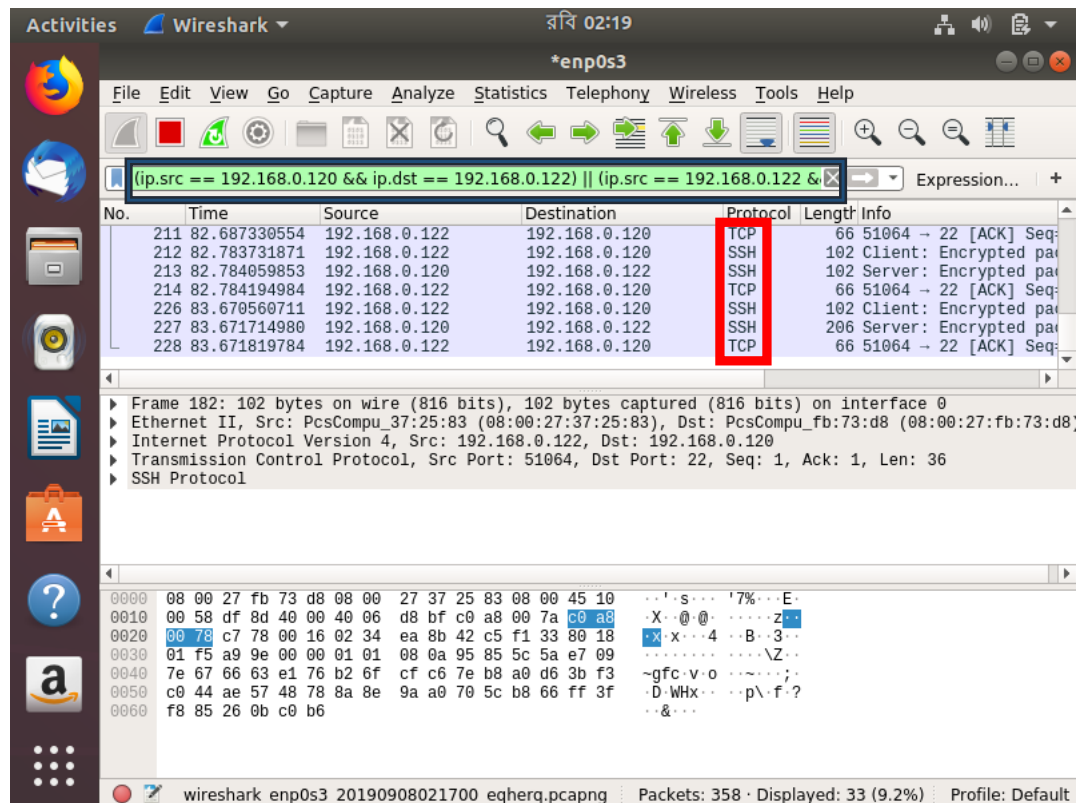
- d. For further demonstration, the **client's IP address** is also noted.



```
client@client-VirtualBox: ~  
File Edit View Search Terminal Help  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo root" for details.  
Thunderbird Mail  
client@client-VirtualBox:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau  
lt qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g  
roup default qlen 1000  
    link/ether 08:00:27:37:25:83 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.122/24 brd 192.168.0.255 scope global dynamic noprefixroute  
    enp0s3  
        valid_lft 83042sec preferred_lft 83042sec  
    inet6 fe80::3e01:b100:7449:e67e/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
client@client-VirtualBox:~$
```

As seen from the screenshot above, the **client's IP address** is found to be **192.168.0.122**.

3. From the **attacker PC**, the following have been done:
- a. **scapy** and **Wireshark** have been installed, as prerequisites of demonstrating the attack.
 - b. To demonstrate that the **attacker** is capable of capturing packets from the **client** to the **server**, and vice versa, Wireshark is opened from the **attacker PC**, and a filter is applied to show all captured traffic between the **server** and the **client**. This is shown in the screenshot below:



The blue frame in the screenshot above shows the filter that has been applied to view packets between the **server** and **client** only. The red frame shows that the **server** and the **client** are communicating via the **TCP** and **SSH** protocols. This proves that a successful **SSH connection** exists between them. This capturing was done from the **attacker PC**, proving that the **attacker** has the ability of capturing, and hence, sniffing packets from between the **server** and the **client**.

4. The code for the attack has been written in python, using scapy. The code sniffs in a packet between the SSH server and the SSH client. This will be possible as long as the **attacker**, the **client** and the **server** are connected to the same network. A **range of sequence numbers** is then set, starting from the **ack** of the sniffed packet to a predefined maximum value. A **forged packet** is then constructed. The **source IP address**, **destination IP address**, **source port number** and **destination port number** of this forged packet are set to be those of the sniffed packet. The **Reset Bit** of the forged packet is set. A loop then sends a series of packets to the victim's machine, with the sequence number of every

packet being set consecutively to a number taken from the range of sequence numbers mentioned earlier.

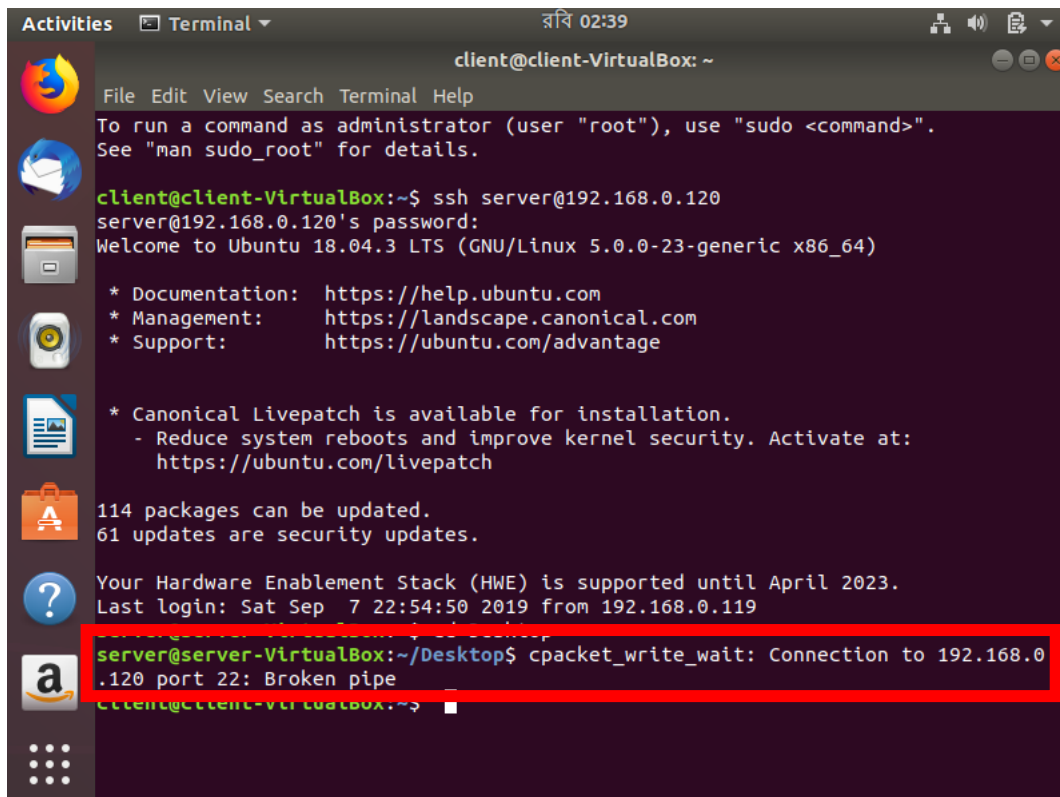
The code will take in the IP address of either the SSH server or the SSH client as its input. These have already been found out in previous steps.

The following screenshot shows the result of running the code from the **attacker PC**, with the **server's** IP address given as input into the code:

[illegible]

The code prints “tcp reset successful” every time it sends a forged packet with a different sequence number to the server.

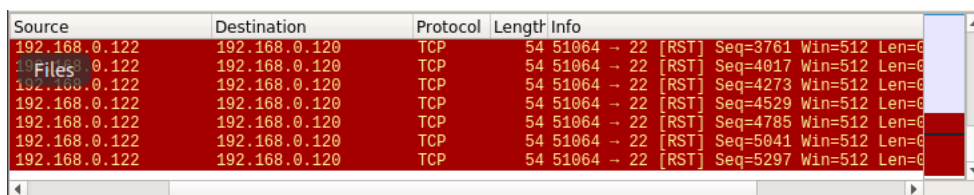
A successful attack will **break the SSH connection** between the **server** and the **client**. This can be observed from the **client** with a notification that reads “Broken pipe”. This is shown in the screenshot below:



```
client@client-VirtualBox: ~  
File Edit View Search Terminal Help  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
client@client-VirtualBox:~$ ssh server@192.168.0.120  
server@192.168.0.120's password:  
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-23-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
* Canonical Livepatch is available for installation.  
- Reduce system reboots and improve kernel security. Activate at:  
  https://ubuntu.com/livepatch  
  
114 packages can be updated.  
61 updates are security updates.  
  
Your Hardware Enablement Stack (HWE) is supported until April 2023.  
Last login: Sat Sep  7 22:54:50 2019 from 192.168.0.119  
  
server@server-VirtualBox:~/Desktop$ cpacket_write_wait: Connection to 192.168.0.120 port 22: Broken pipe  
client@client-VirtualBox:~$
```

As shown in the red frame of the screenshot above, the connection to the **server(192.168.0.120, port 22)** has been broken. **Port 22** is the port over which **TCP connection** for **SSH** is normally established.

The success of the attack can be further assessed by observing the traffic between the **server** and the **client**, from the **attacker**, by using **Wireshark**.



Source	Destination	Protocol	Length	Info
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=3761 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=4017 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=4273 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=4529 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=4785 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=5041 Win=512 Len=0
192.168.0.122	192.168.0.120	TCP	54	51064 → 22 [RST] Seq=5297 Win=512 Len=0

The screenshot above has been taken from Wireshark running in the **attacker PC**. It clearly shows that **TCP RST packets** have been received from the client to the server. This can be seen by the **[RST]** bits under the “info” section of the screenshot.

Therefore, the **TCP RST attack** has been successful.

Possible Countermeasures against TCP RESET attacks

- Choosing sequence numbers (seq's) and acknowledgement numbers (ack's) "randomly" will help to prevent TCP Reset attacks. The attack depends largely on the attacker's ability to "guess" the ack of recent packets between the connections they are trying to break. Assignment of seq's and ack's in a way that is only known to the source and the receiver, but appears random to the attacker, will help prevent TCP Reset attack.
- TCP pacing is a technique where the TCP packet sources can space the data packets to be sent away from each other, so that it is possible to mitigate the burst packet transmissions. TCP Reset attack will lead to denial of service attacks, leading to congestions and eventual packet losses. By preventing burst transmissions, the durations of congestions and dropping of packets by the receiver will be reduced.