Name: Bharath Karumudi
Lab: TCP Attacks

Task 1: SYN Flooding Attack

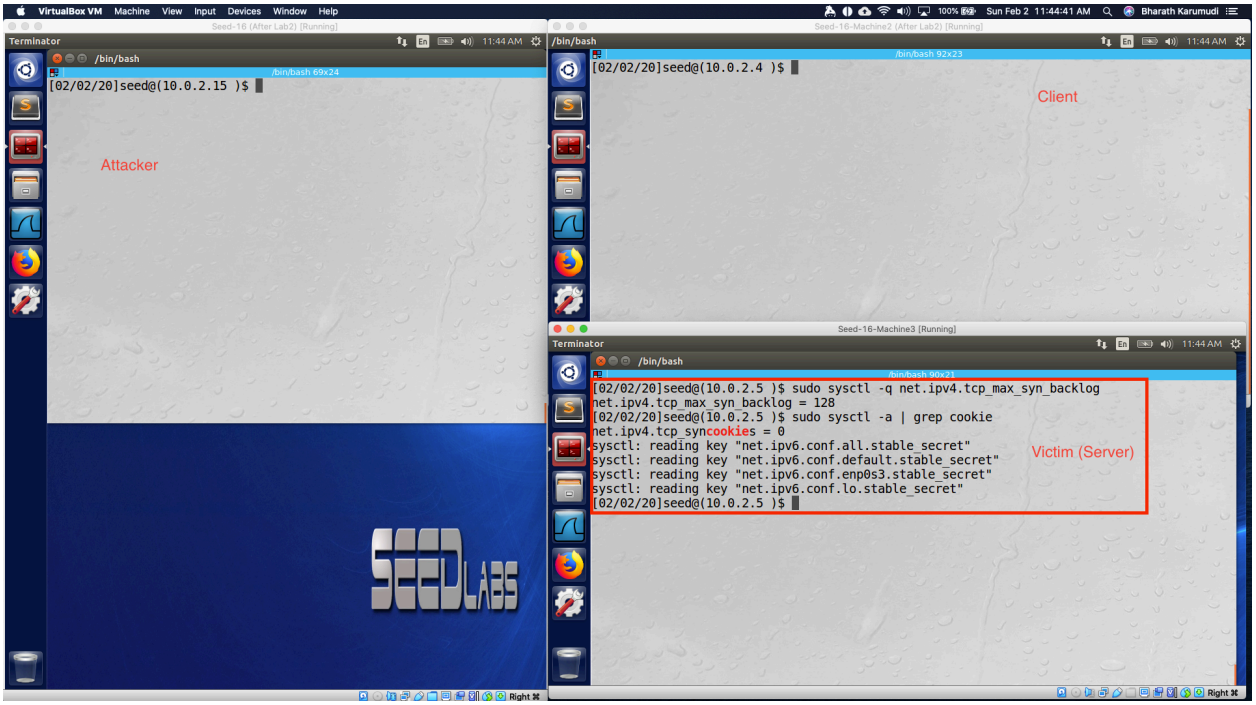

Fig1: Showing the TCP Queue size and SYN Cookie settings

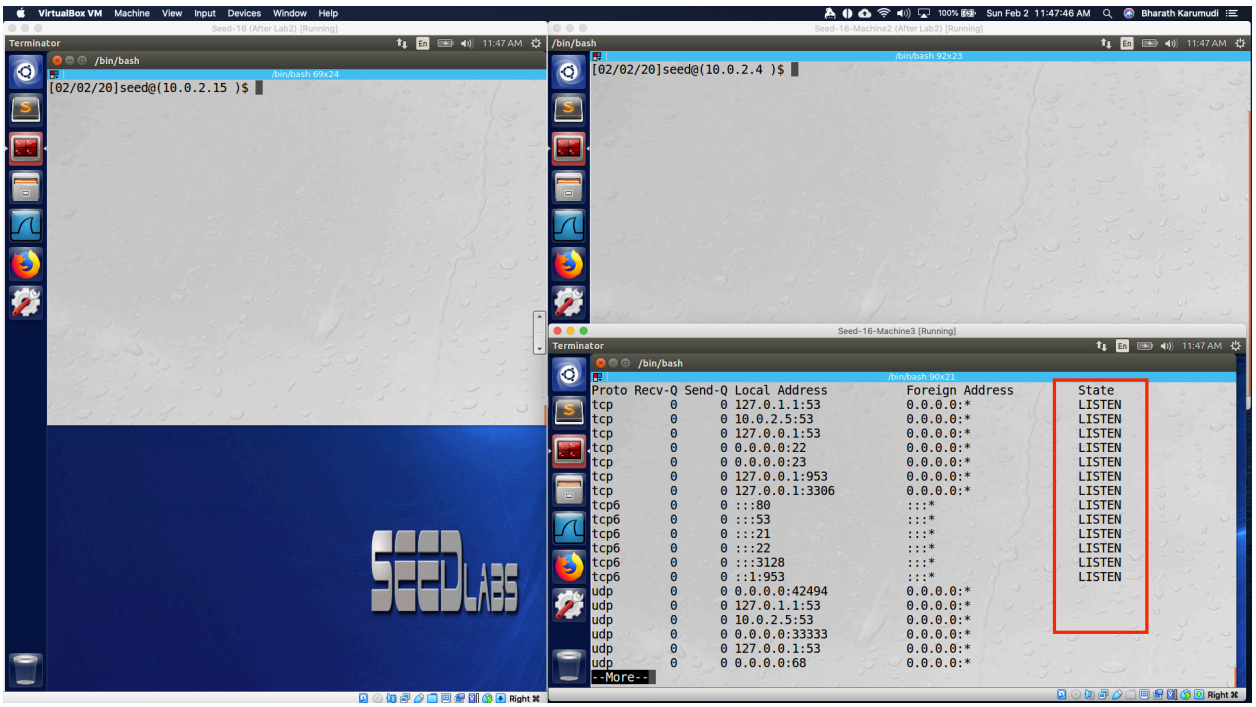

Fig2: Connection Status before SYN Flood attack

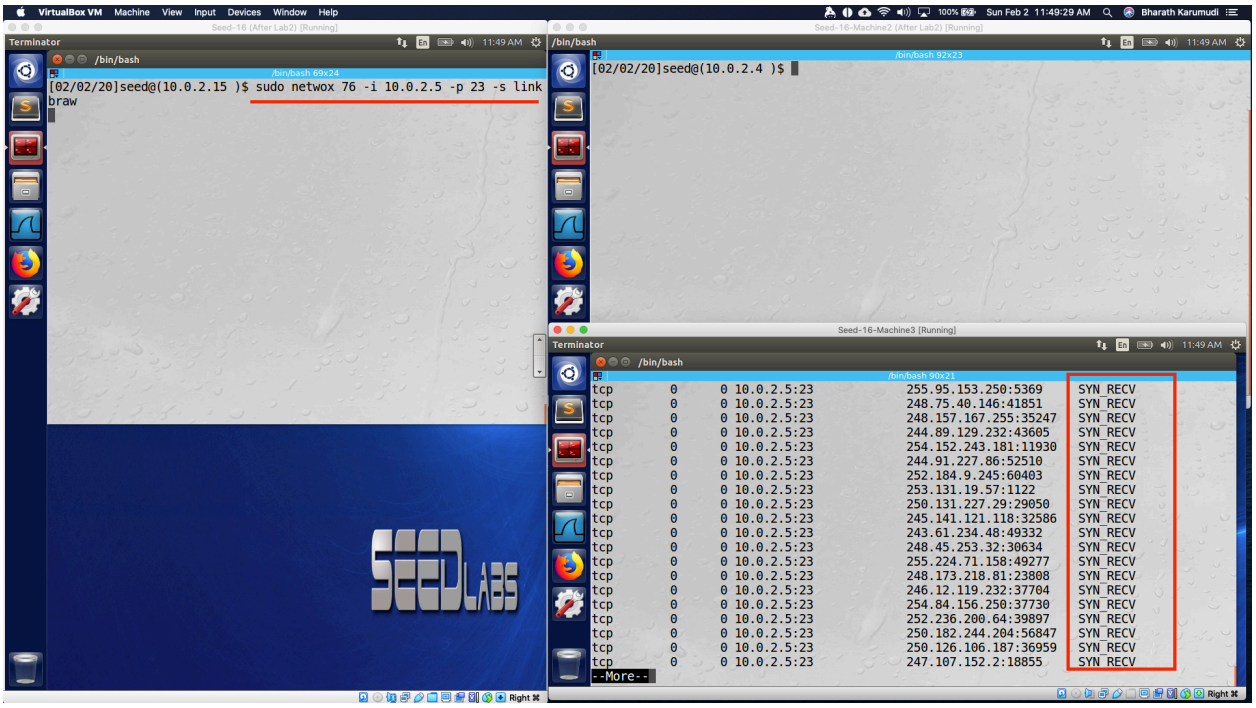

Fig3: Connection status during the SYN Flood attack

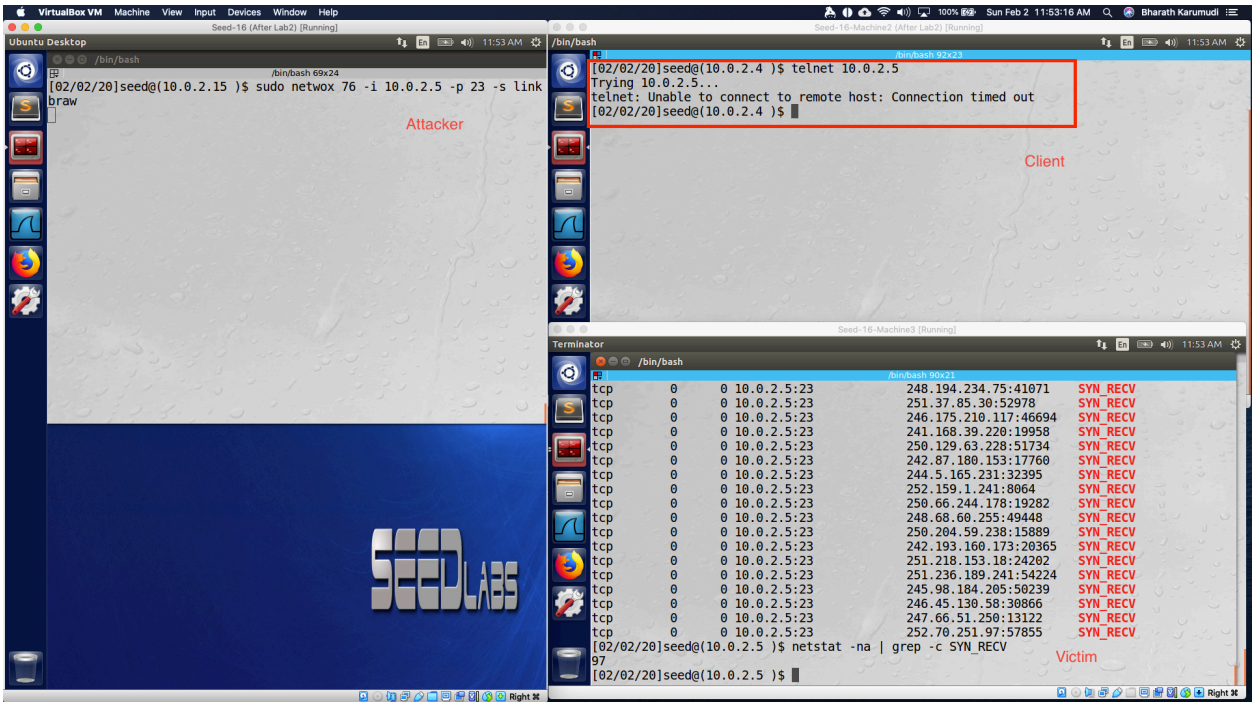

Fig4: Client session timeout during SYN Flood attack

Observation: Disabled the SYN Cookie settings on the Victim server (10.0.2.5) and ran the SYN flood using the netwox utility on victim port 23 and during this time when client is trying to connect to victim server, the server is unable to establish the tcp connection.

Explanation:

1. Using the command “sudo sysctl -w net.ipv4.tcp_syncookies=0” disabled the SYN cookie on the server.
2. Ran the attack from attack from attacker machine using netwox 76 utility: “sudo netwox 76 -i 10.0.2.5 -p 23 -s linkbraw”
   After running the SYN flood, the server tcp queue is having half-open connections represented with “SYN_RECV”. As the SYN cookies are disabled, the system will keep the half open connections in the queue and once it is full, it will be unable to process any new connections. To demonstrate, when client is trying to connect to the server on telnet, the client was unable to connect and got session timeout message.

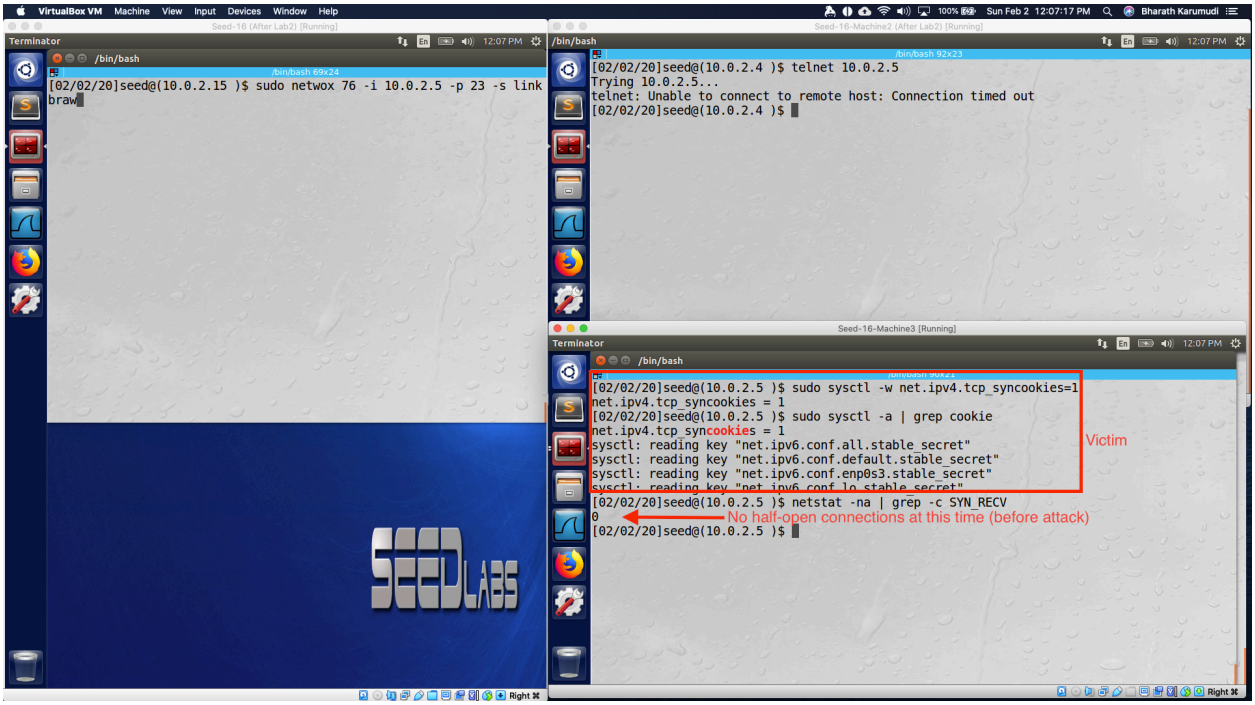

Fig5: Enabled the SYN Cookie on the victim server

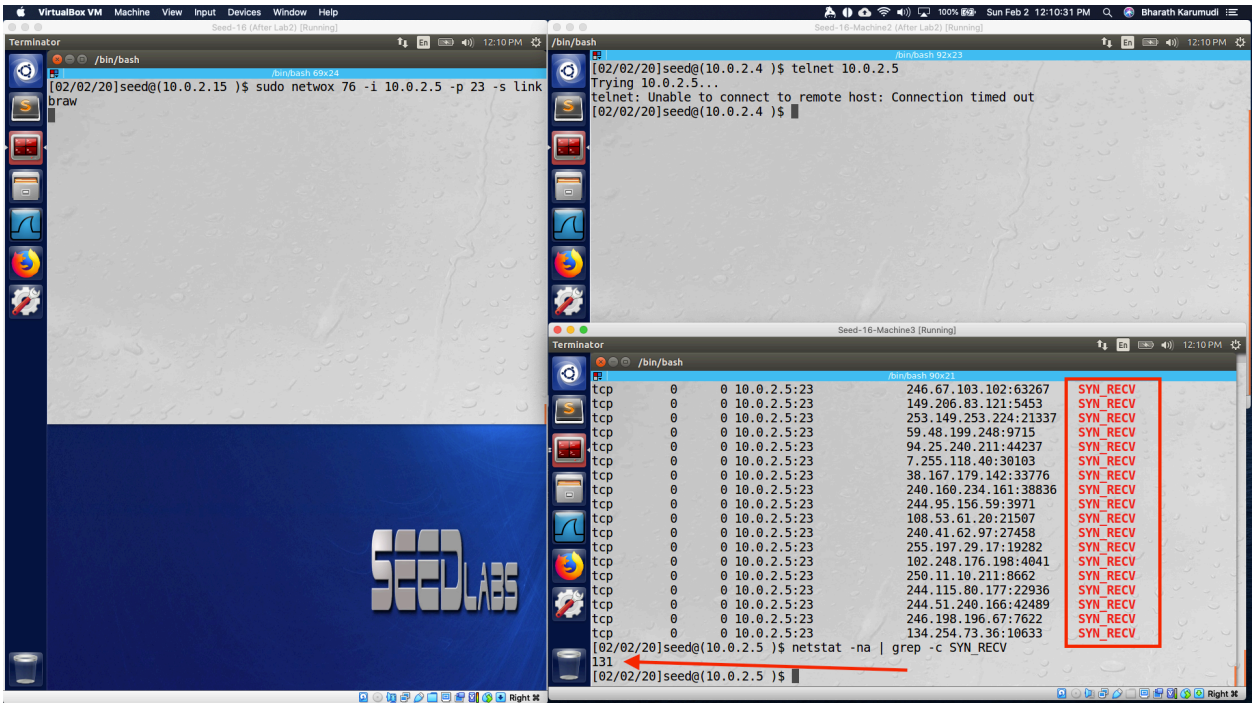

Fig6: Connection status during SYN flood attack

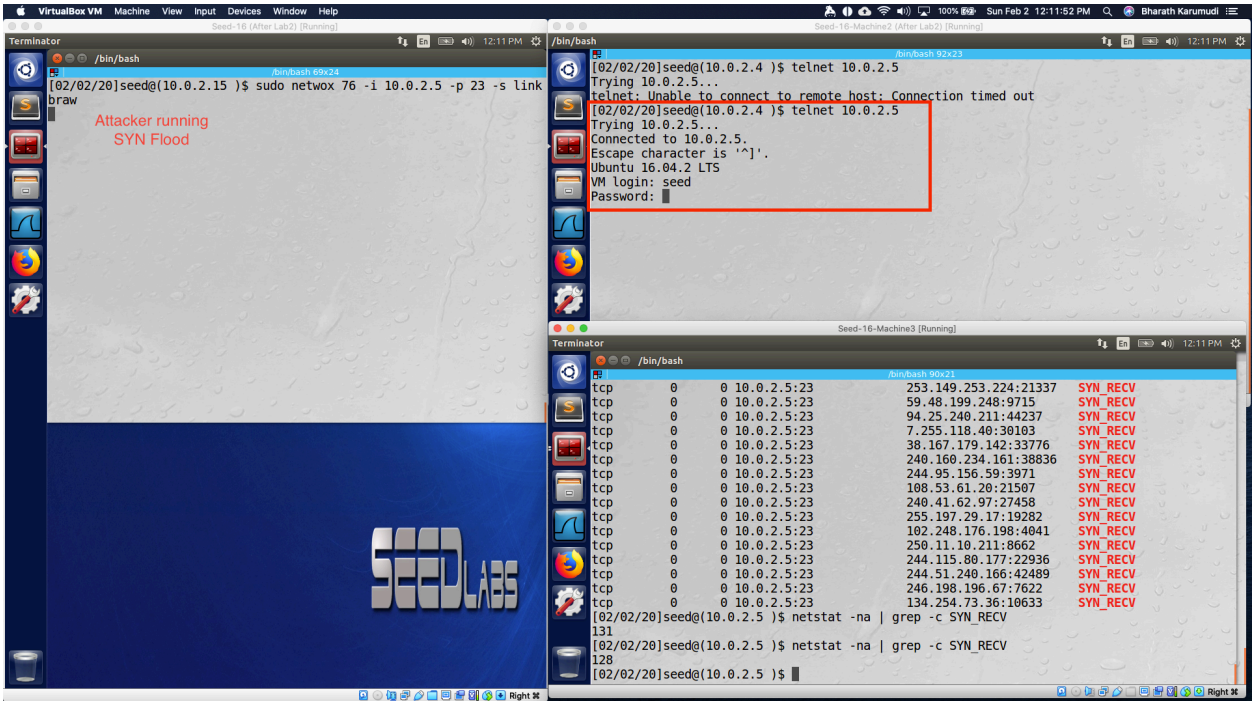

Fig7: Client able to connect during SYN Flood attack

Observation: Enabled the SYN Cookie settings on the Victim server (10.0.2.5) and ran the SYN flood using the netwox utility on victim port 23 and during this time when client is trying to connect to victim server, the server is able to establish the tcp connection.

Explanation:

1. Using the command “sudo sysctl -w net.ipv4.tcp_syncookies=1” enabled the SYN cookie on the server.
2. Ran the attack from attack from attacker machine using netwox 76 utility: “sudo netwox 76 -i 10.0.2.5 -p 23 -s linkbraw”
   After running the SYN flood, the server tcp queue is having half-open connections represented with “SYN_RECV”. As the SYN cookies are enabled, the system will allocate only for the replies received for the ACK+SYN. To demonstrate, when client is trying to connect to the server on telnet, the client was able to connect.

   The SYN Cookie is a defense mechanism and triggers when the system seems there is a SYN Flood attack and to avoid the tcp queue full, it will allocate to the connections who responded to the SYN+ACK.

Task 2: TCP RST Attacks on telnet and ssh Connections

A. Attack on telnet

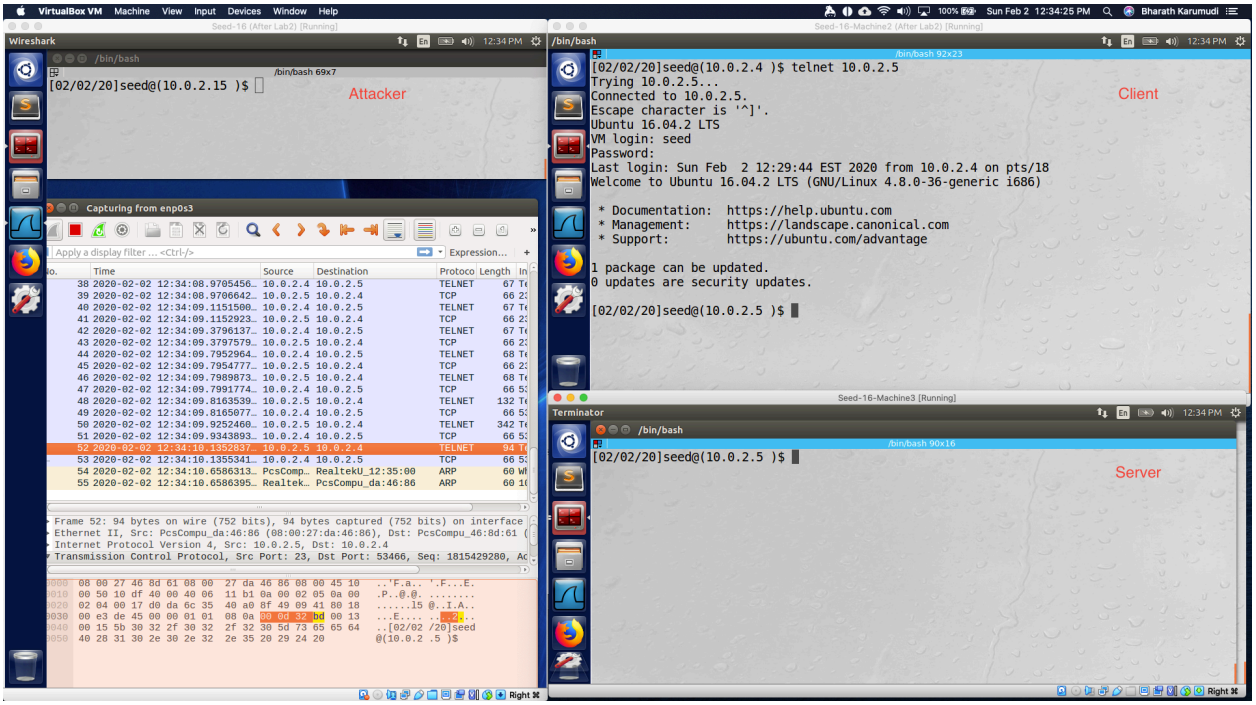

Fig1: A telnet connection established between Client and Server over TCP

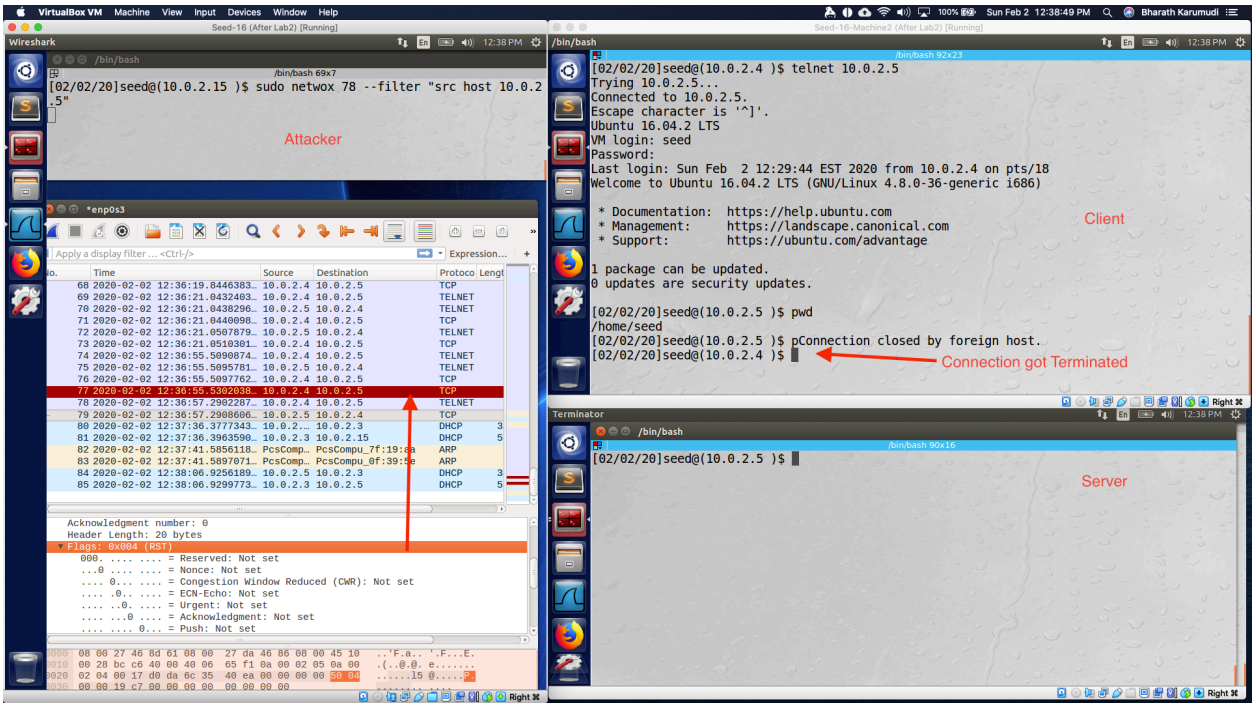

Fig2: TCP connection for telnet between Client and Server got terminated with RST packet

Observation: Established a telnet connection from Client to Server and the attacker sniffed the traffic and created an RST packet using netwox 78 and filter. When the session transmission was happening, the RST packet was sent between client and server and the session got terminated.

Explanation: To demonstrate the attack, established a telnet connection from Client to Server as shown in Fig1. From the attacker machine, sniffed the traffic and gained the information needed like source and destination IP and ports. Based on that created a pcap filter in network 78 utility and ran “sudo netwox 78 --filter "src host 10.0.2.5"”. When the client is typing and immediately the attacker initiated the spoofed RST packet from server to client to terminate the session. With that, the established telnet connection was terminated.

B. Attack on SSH

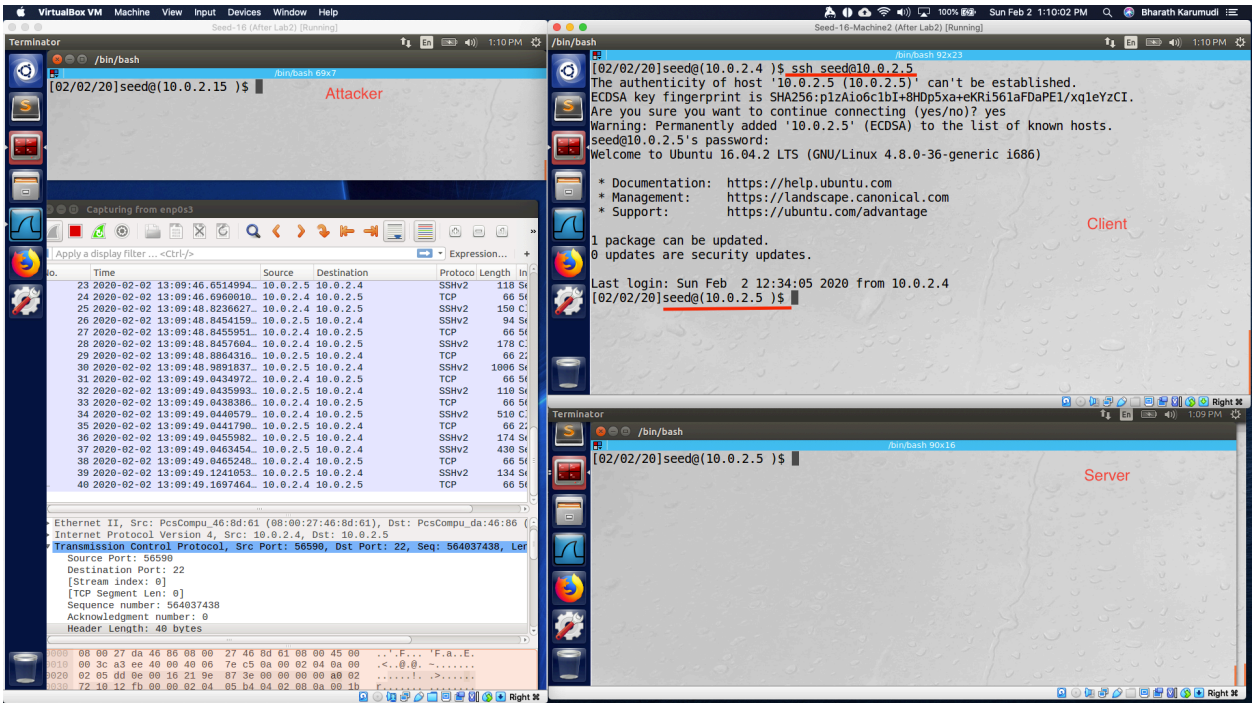

Fig3: Established SSH connection between Client and Server

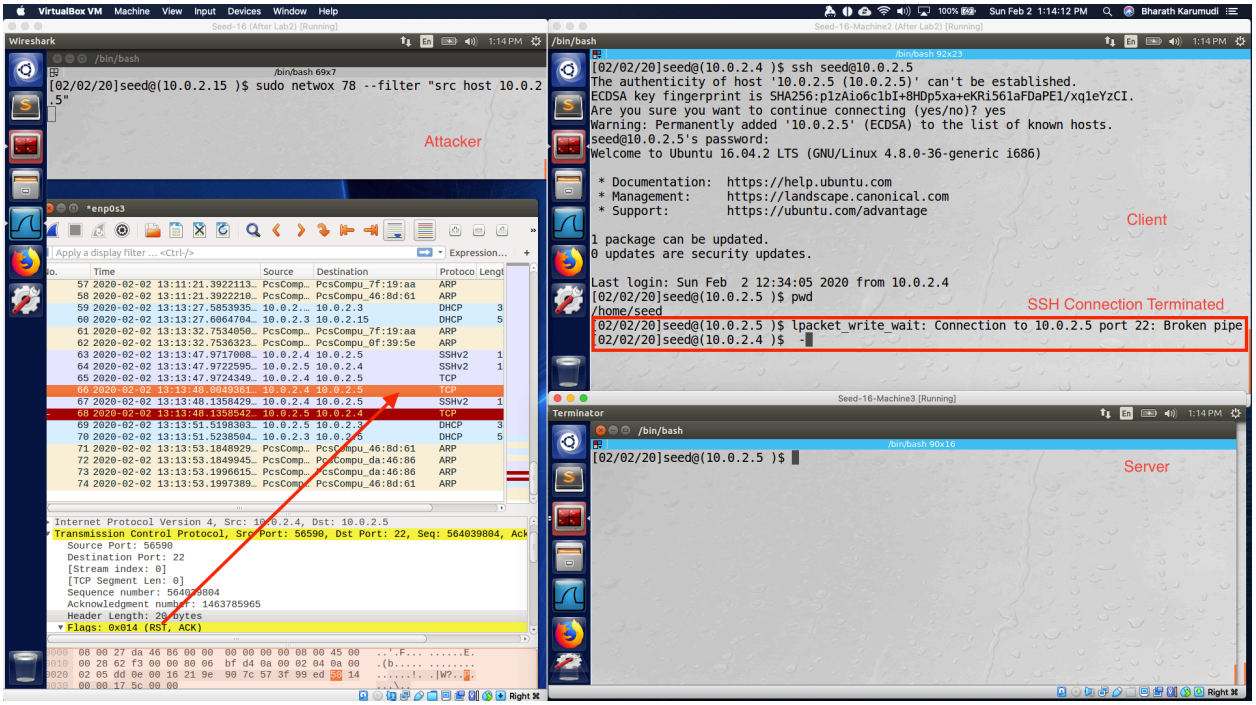

Fig4: SSH Connection got terminated by RST spoofed packet imitated by Attacker.

Observation: Established a SSH connection from Client to Server and the attacker sniffed the traffic and created an RST packet using netwox 78 and filter. When the session transmission

was happening, the RST packet was sent between client and server and the session got terminated.

Explanation: To demonstrate the attack, established a SSH connection from Client to Server as shown in Fig3. From the attacker machine, sniffed the traffic and gained the information needed like source and destination IP and ports. Based on that created a pcap filter in network 78 utility and ran “sudo netwox 78 --filter "src host 10.0.2.5"”. When the client is typing and immediately the attacker initiated the spoofed RST packet from server to client to terminate the session. With that, the established SSH connection was terminated.

Task 3: TCP RST Attacks on Video Streaming Applications

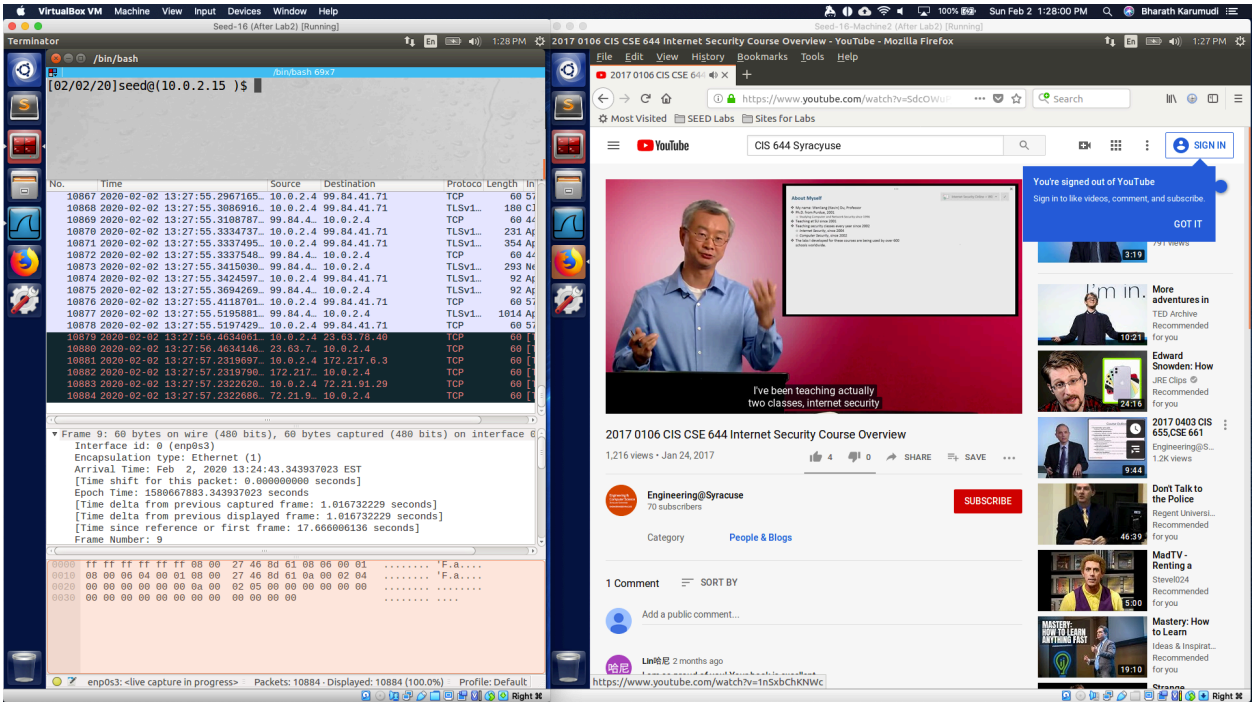

Fig1: Playing the YouTube video normally before the attack

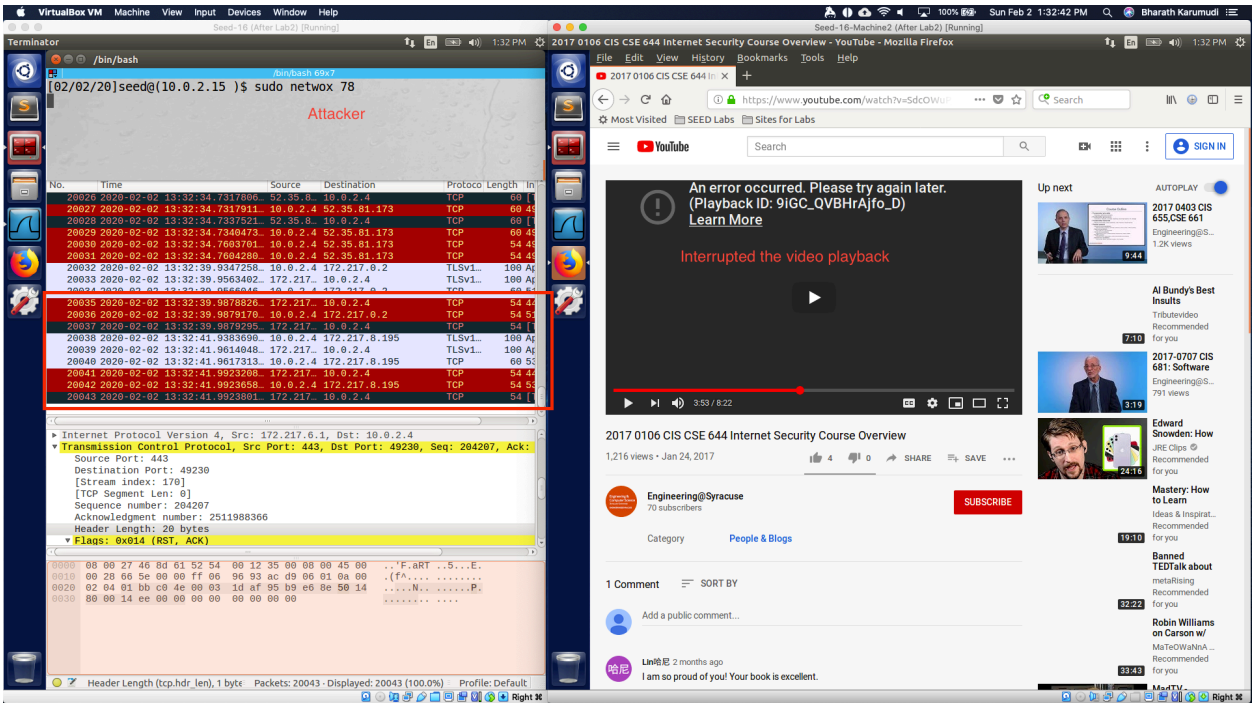

Fig2: Video playback was interrupted on the victim machine (Right) by the attacker

Observation: Initially the video playback was started normally and based on the traffic, the attacker on the same network initiated the RST packet to terminate the TCP connection between the Victim and YouTube and the connection was terminated successfully and the playback was interrupted.

Explanation: The RST packet will tell the receiver that the sender is terminating the session. In the case, first the client started a TCP connection with YouTube to play the video and then the attacker sent the RST packet and terminated the connection. With the termination, the video stopped playing. We can also see the RST packets in the Wireshark.

Task 4: TCP Session Hijacking

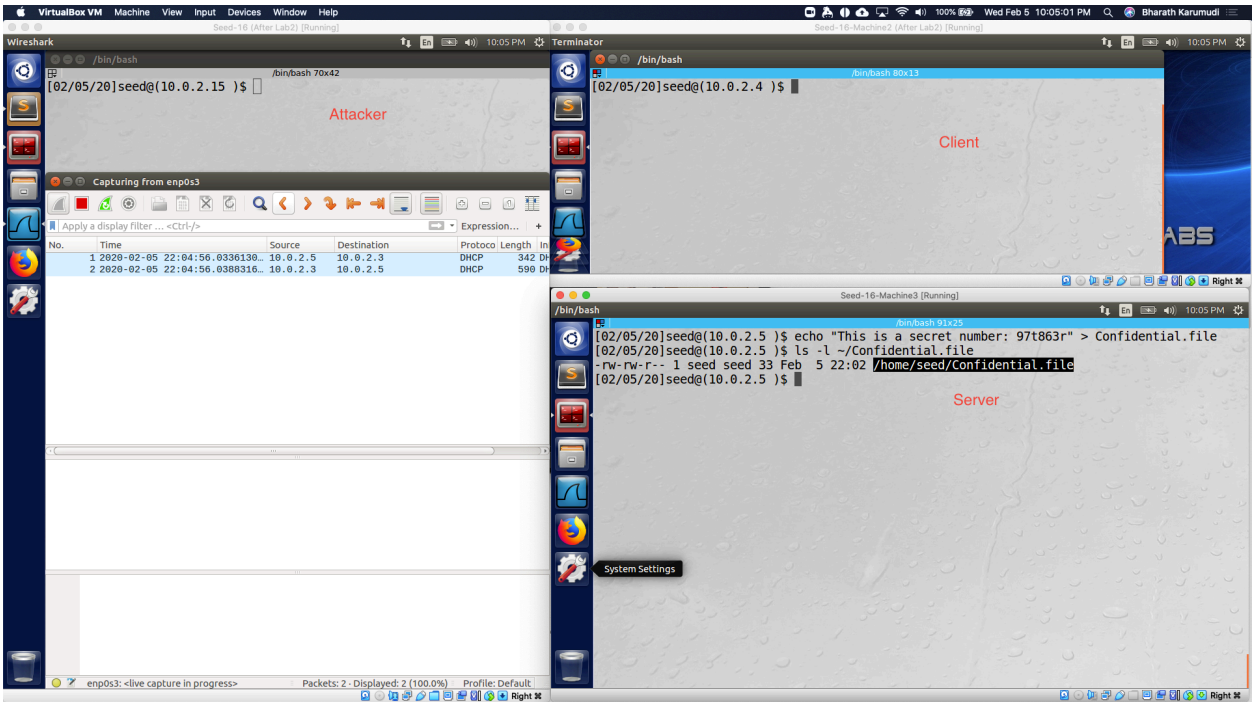

Fig1: Created a *confidential.file* on the server

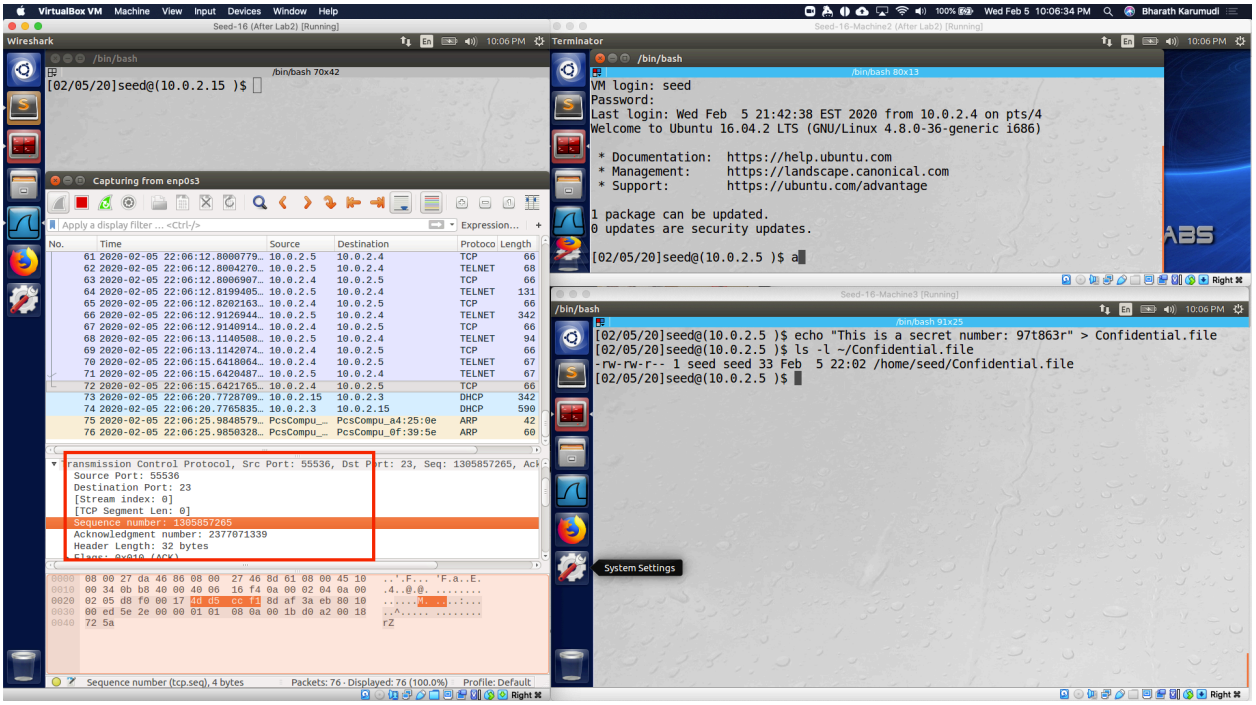

Fig2: Gathering required information from Wireshark on the attacker machine

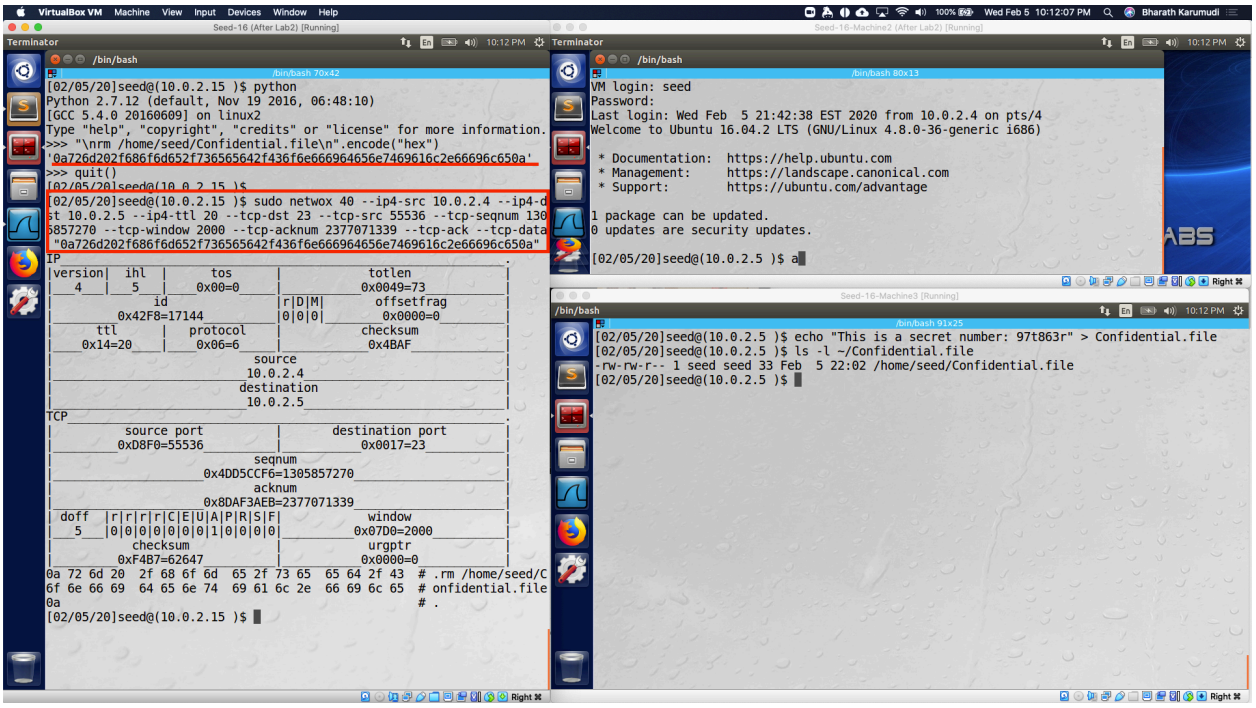

Fig3: Preparing the attack

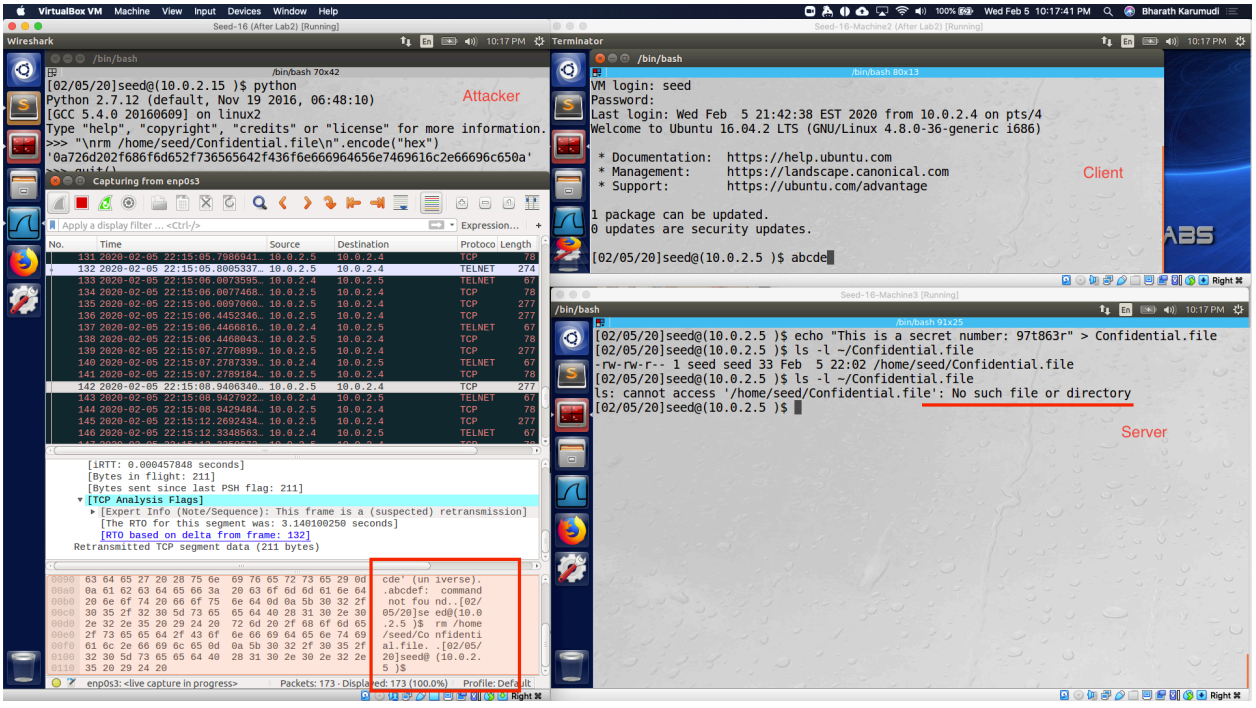

Fig4: Hijacked the session and removed the file

Observation: Created a confidential.file on the server, and sniffed the TCP traffic between client and server. Using the sniffed details, hijacked the established TCP session and injected the TCP packet with a buffered Seq number that removes the confidential file. When the user reached the injected sequence number the file got removed.

Explanation:

1. Created a confidential.file on server which has confidential information under /home/seed.
2. Client Established a telnet connection from client machine to server.
3. As an attacker, using the Wireshark sniffed the traffic between Client and Server to gather the Source IP, Destination IP, Source Port, Destination Port and more importantly Sequence Number which is 1305857265 (Fig 2).
4. Once this information is gathered, created a payload by converting the command to hex and using netwox with a sequence number (+5) 1305857270 and the spoofed packet was sent out (Fig 3).
5. When the user is typing on his terminal and the TCP sequence is matched with the attacker packet, the spoofed TCP packet was taken, and the instruction was executed to remove the file and the file was removed (Fig 4).

Task 5: Creating Reverse Shell using TCP Session Hijacking

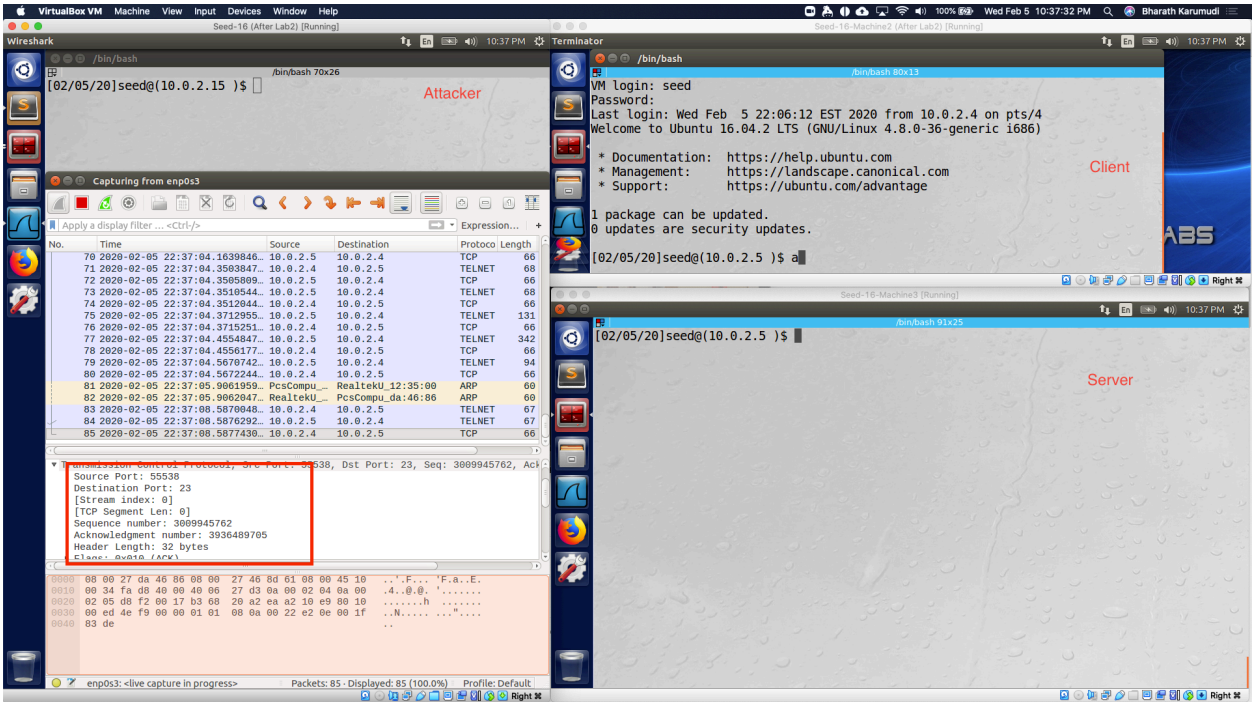

Fig1: Gathering the required details for attack

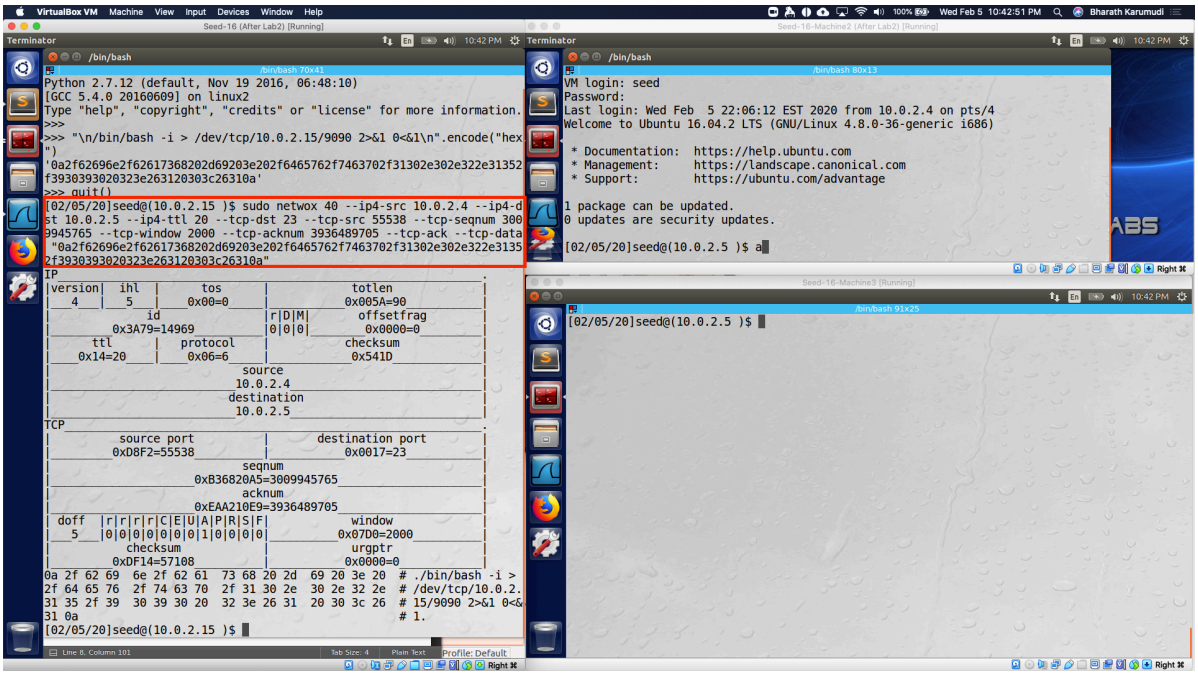

Fig2: Sending the attack

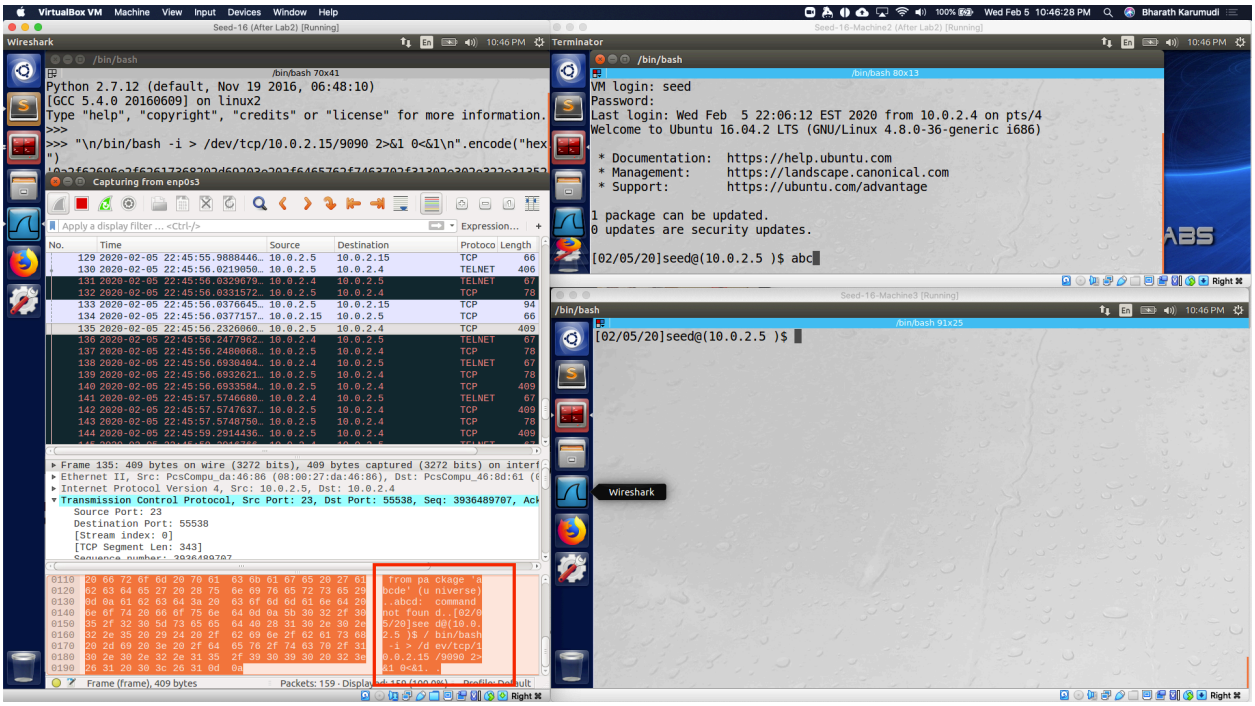

Fig3: Spoofed packet executed

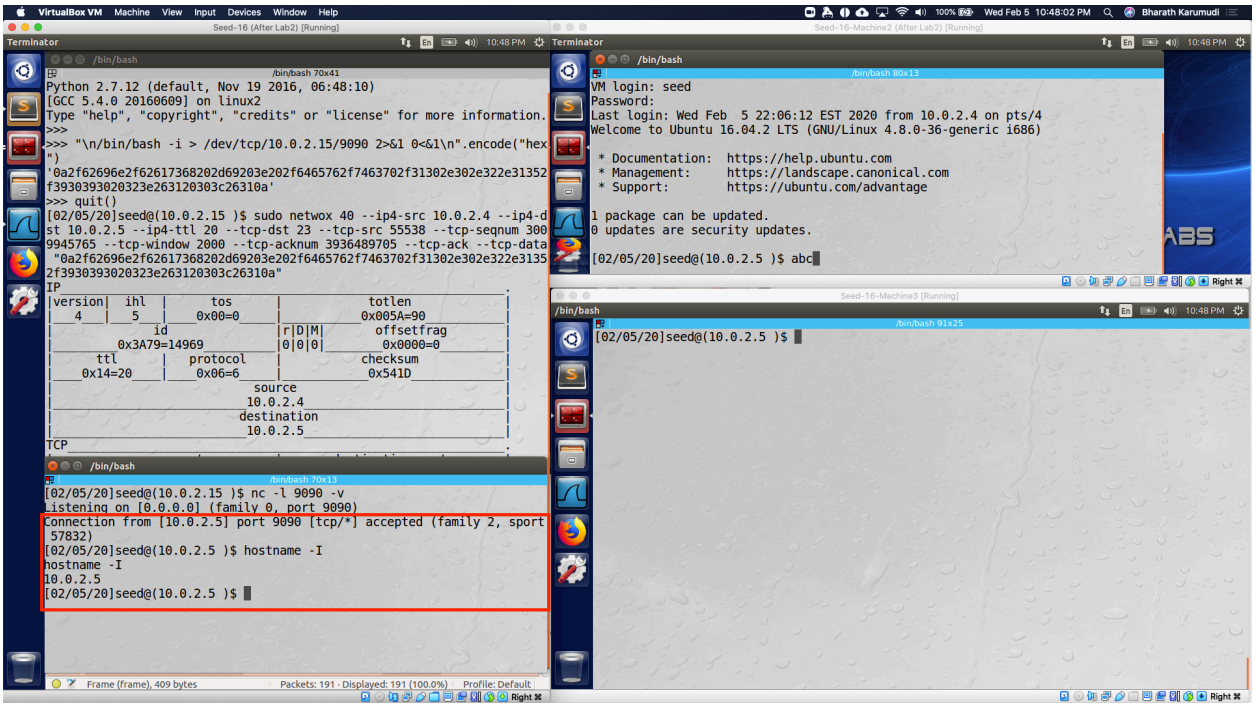

Fig4: Got the reverse shell (Left side)

Observation: The user established the telnet connection with server and as an attacker spoofed a TCP packet and hijacked the TCP session and got the reverse shell.

Explanation:

1. The user from his machine established a telnet connection with the server.
2. As attacker sniffed the network communications between user and server using Wireshark and gathered the required information like Source and Destination IP and Port and the Sequence number (3009945762) as shown in Fig1.
3. Using the gathered information and created the data for TCP packet in hex for the reverse shell and with a new sequence number (3009945765) sent a spoofed TCP packet.
4. As an attacker, opened the netcat (nc) to listen the traffic on port 9090 for reverse shell.
5. When the user is initiating the new TCP packets the sequence number increments and reached the one that we spoofed. Once reached, the spoofed packet was taken by the server and created the reverse shell that sends out on 9090 from the server.
6. As shown in Fig4, we got the reverse shell and able to execute the commands.