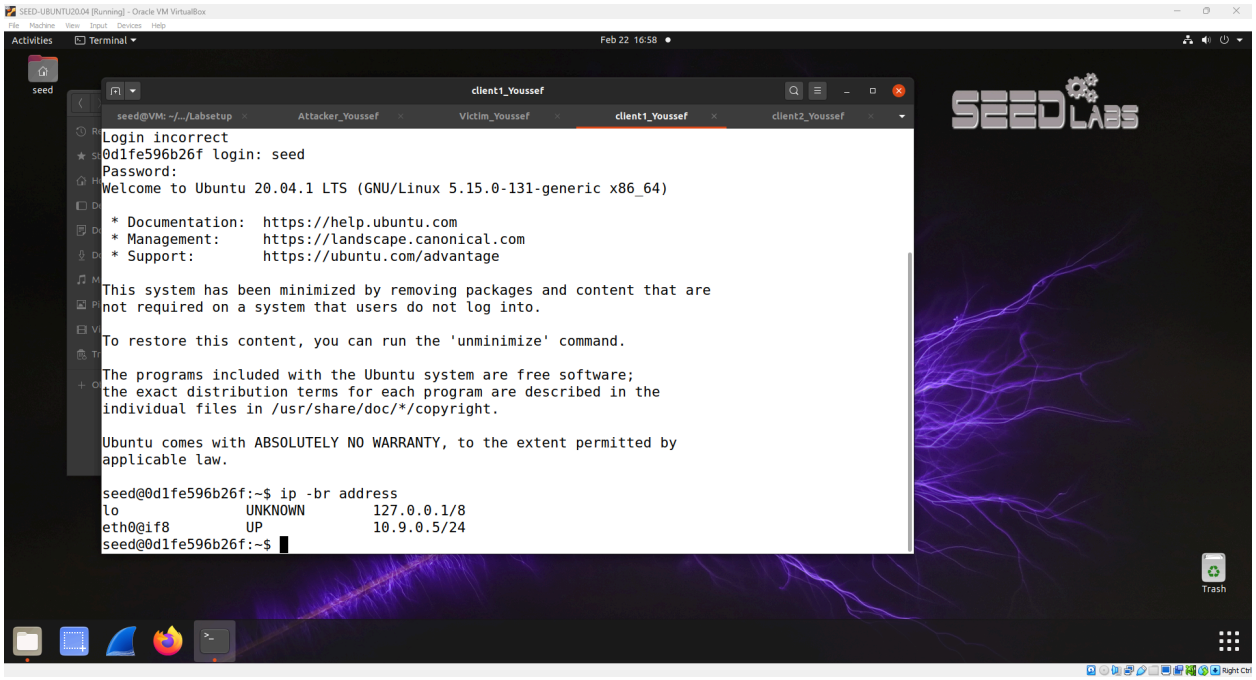
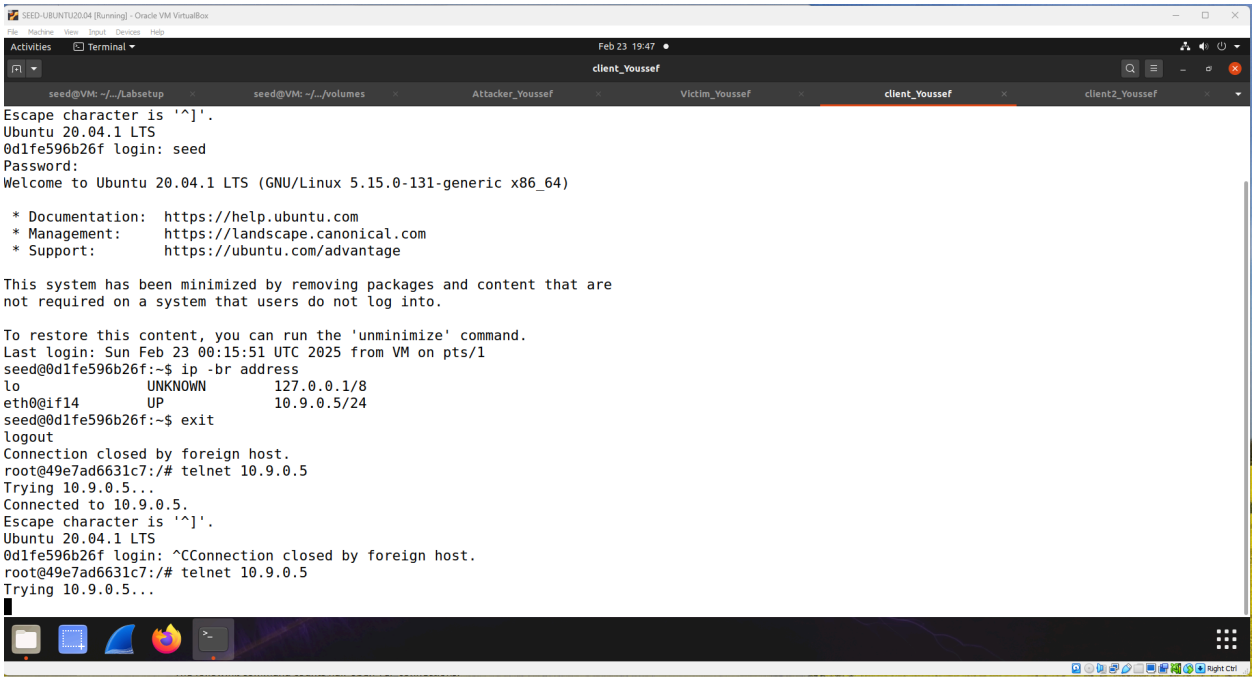
Screenshot 1:



Question 1:

The attack failed because the victim machine had protections like SYN cookies or a firewall that prevented the connection backlog from filling up. These protections allow the system to handle a large number of SYN requests without being overwhelmed. If these defenses were removed or bypassed, the SYN flood could have succeeded in making the system unresponsive.

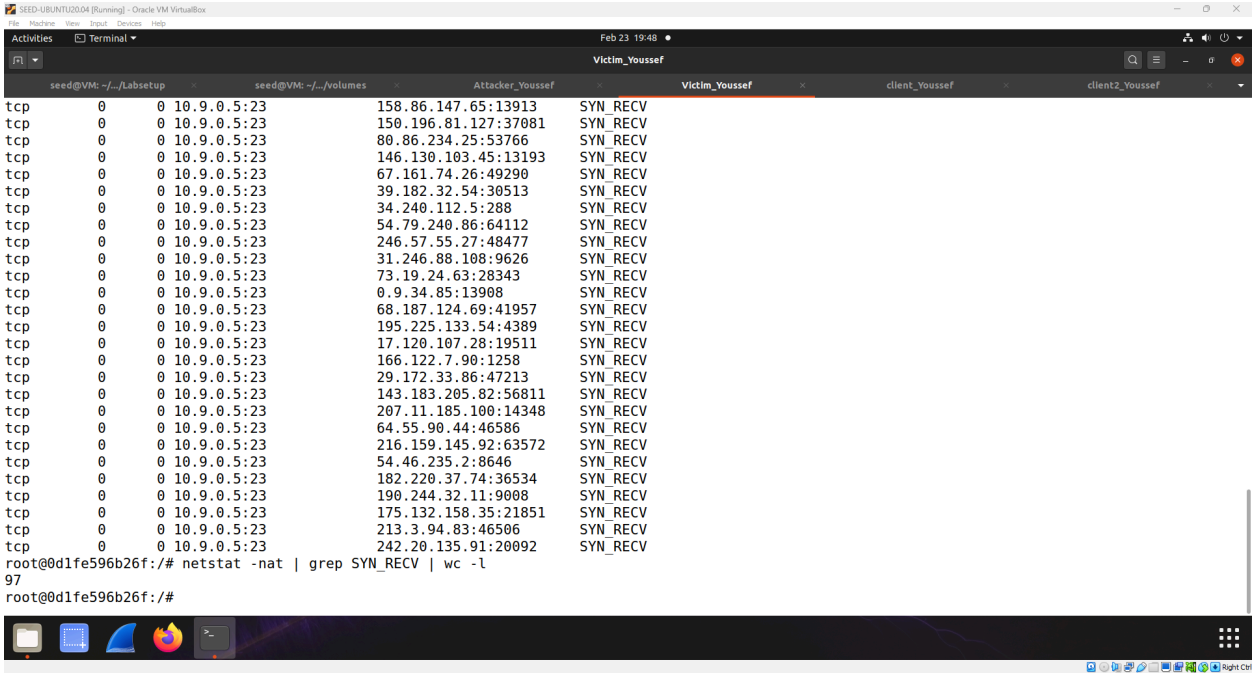
Screenshot 2:



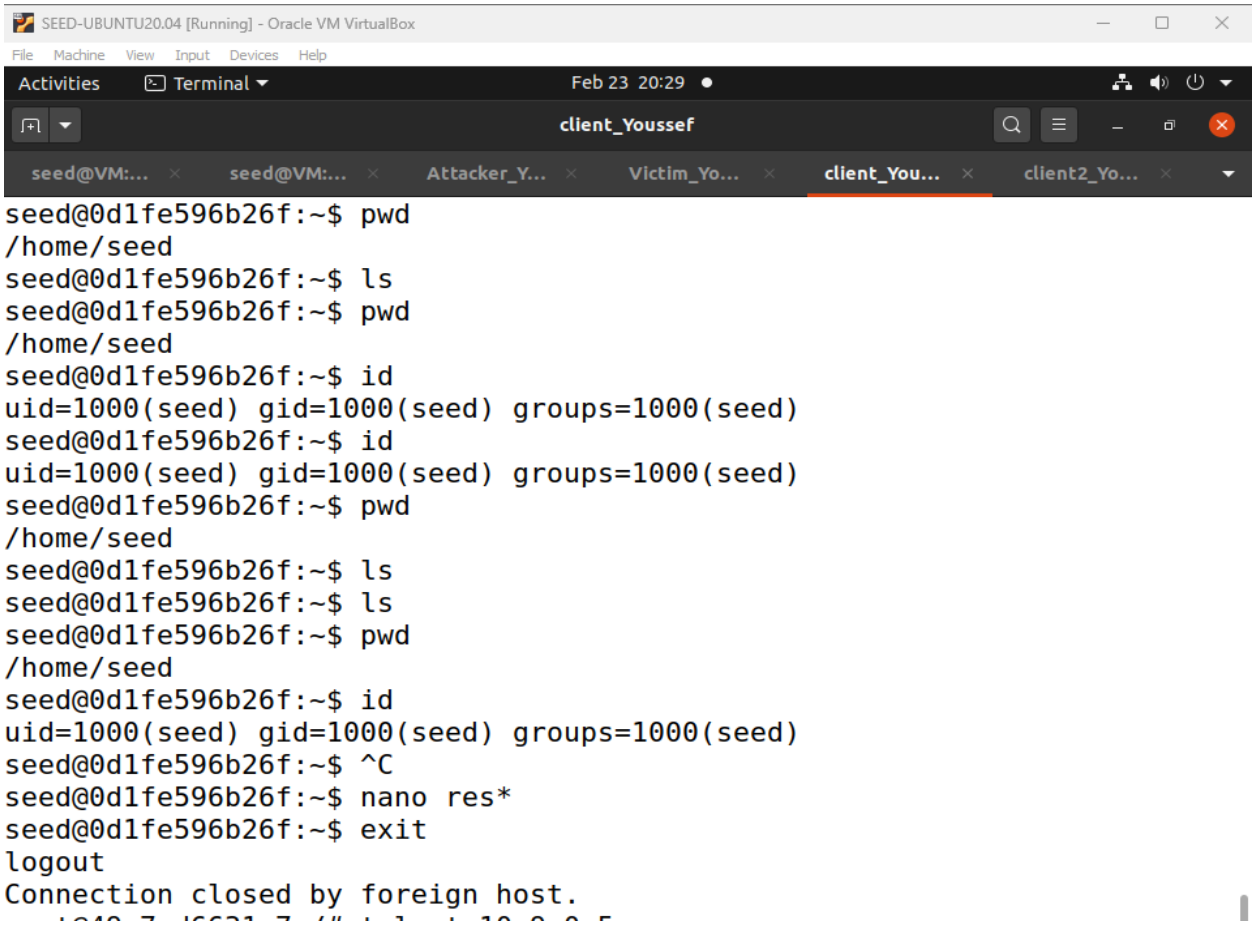
Question 2:
The attack likely failed because one Python script alone didn’t send enough SYN packets to overload the victim machine’s connection queue. The system was still able to process incoming requests because the script wasn’t sending packets fast enough to fill up the backlog. As a result, the victim machine could still accept new connections.

Question 3:
In my testing, I only needed to run one attack window for the attack to work. I think my system was able to send packets quickly enough to fill the connection queue before the victim machine could recover. Because of this, I didn’t need to open multiple attack windows like in the original test.

Screenshot 3:



Screenshot 4:

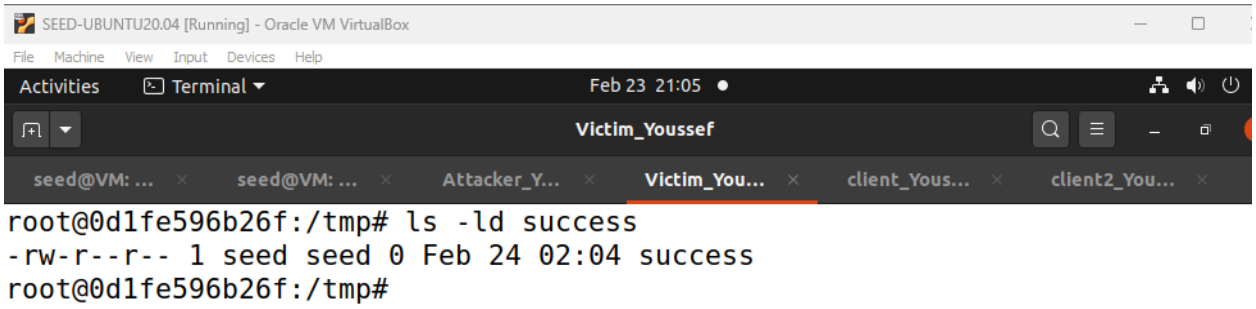


Question 4:
After I hit the spacebar 3 times, the connection tore down.

Question 5:
Three RST packets were spoofed. The output shows three RST flags with different sequence numbers, meaning three reset packets were sent. This caused the connection to close immediately.

Question 6:
The attacker runs reset_auto.py to monitor TCP packets between the client and server on port 23. The script sends fake RST packets that appear to come from the client. When the server receives one, it assumes the client wants to end the connection and shuts it down. The client is suddenly disconnected and can no longer communicate. The TCP session is immediately cut off, skipping the usual FIN-ACK process.

Picture 5:



Question 7:

After the screen freezes, the TCP connection between the client and server is terminated because of the spoofed RST packets. This means the client can no longer send or receive data through Telnet. On the victim machine, a "success" file is created. This file indicates that the server detected the connection was disrupted and automatically ran a command to generate the file as a response to the attack.

Question 8:

The commands start with a new line to make sure they run as separate commands in an interactive shell session. This prevents them from being combined with previous commands, ensuring they execute properly. It also helps maintain compatibility with how the shell processes input, avoiding errors or unexpected behavior. Additionally, it reduces the risk of corrupted input, making sure the commands are correctly interpreted and executed.

Screenshot 6: