**COMP 4108 Assignment 5**

Shane Bishop

Student #: 101030053

# Part A

**Question 1**

In the nmap man page, under the “OPTIONS SUMMARY” section, it says

SCAN TECHNIQUES:

-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans

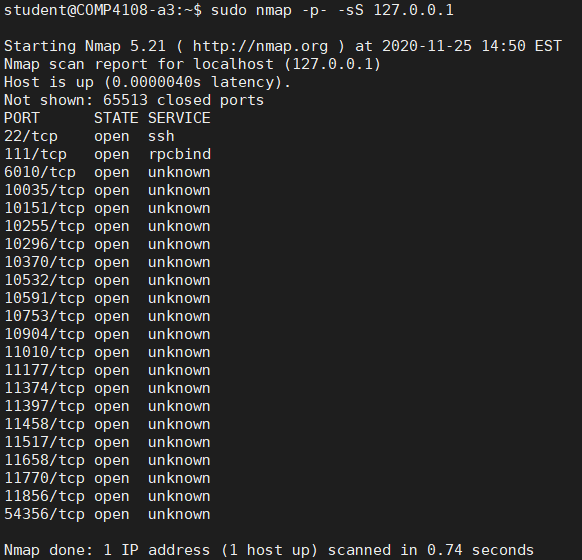
This means to do a TCP SYN scan, the -sS option should be specified.

The -p option of nmap is used to specify a port range. The man page says, “The beginning and/or end values of a range may be omitted, causing Nmap to use 1 and 65535, respectively. So you can specify -p- to scan ports from 1 through 65535.” This means I can scan all ports with the option -p-.

Therefore, the command I want to run is

sudo nmap -p- -sS 127.0.0.1

Here is a screenshot of the output of this command:



**Question 2**

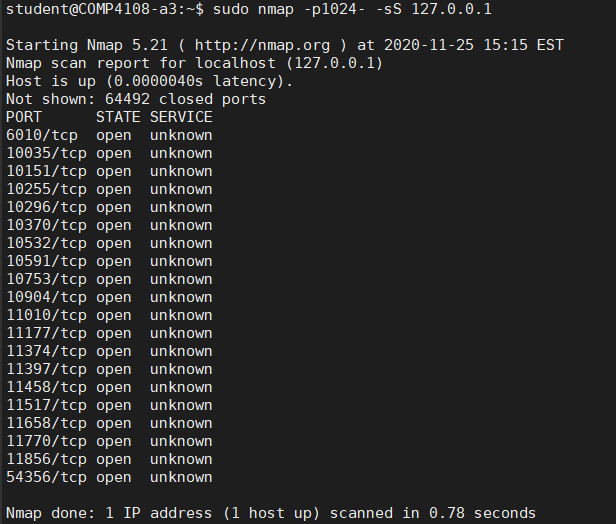
Ports below 1024 are reserved system ports that I do not want to attempt to transmit files over. For that reason, I will only scan ports above this. (I confirmed with TA Chris Bennett that this is okay for this question.)

For this reason, I will use the -p option of nmap to restrict the port scan. I therefore edited the command I ran in question 1 to

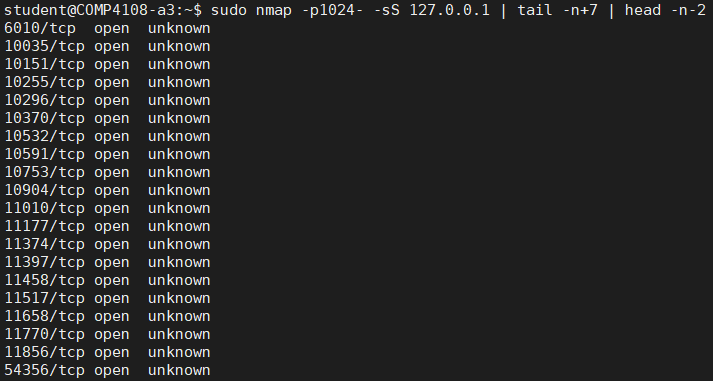
sudo nmap -p1024- -sS 127.0.0.1

Since I omitted an end port for the range, nmap will automatically scan all ports through to 65535, as indicated in my quote from the man page (quoted in my answer to question 1).

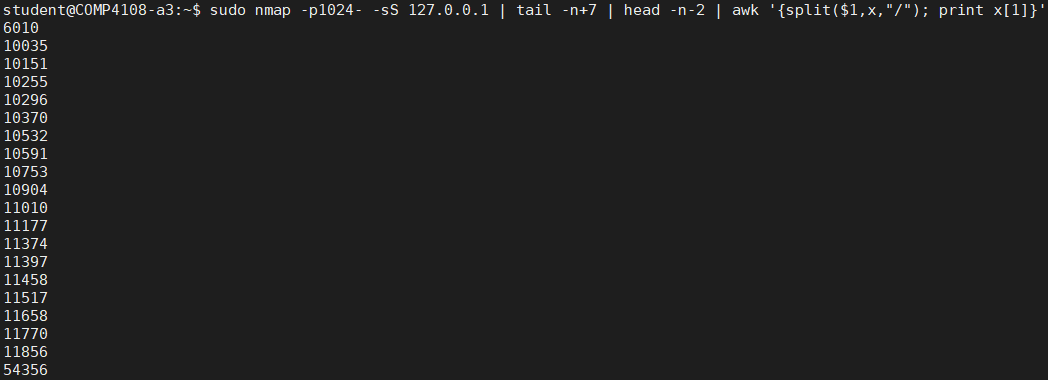
When I run this command, this is what I get:



­I can use tail and head to chop off lines that do not contain port numbers:



Then I can use awk to extract just the port numbers, in such a way that I get all the ports delimited by newline characters:



Let’s breakdown the awk command I used.

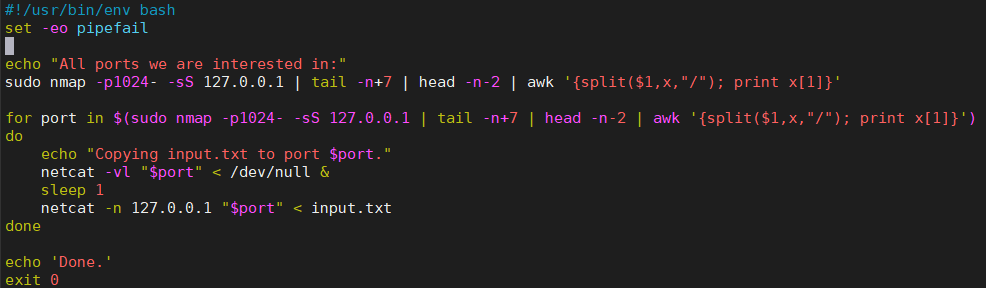
The full awk command is awk '{split($1,x,"/"); print x[1]}'

The split() function first takes each line of input and splits it, using the “/” character as the delimiter. The result of the split() is stored in the variable x as an array.

Then, after the semi-colon, I print the port. The awk split() function has a weird property that the first value of the split is stored at index 1, rather than at index 0. So that is why I use print x[1] at the end of the awk command to print the port.

Now that I have all the ports, delimited by newlines, I can loop over these ports with a bash for loop.

Here is the bash script I wrote (the source code for this script is attached as code/a\_q2.sh):



On line 5, I print the ports I’m going to loop over. Then I loop over each port. On line 9 I log that I am about to copy input.txt over the port.

On line 10, I open a netcat connection on the port for the current iteration of the loop. -l tells netcat to listen for an incoming connection, and -v enables verbose output. I redirect /dev/null to stdin so netcat won’t try to read anything from stdin. Finally, I tell bash to run netcat in the background (using &) so that the script can proceed to run the next command.

On line 11, I cause the script to pause for 1 second to let netcat complete initialization to listen to incoming connections.

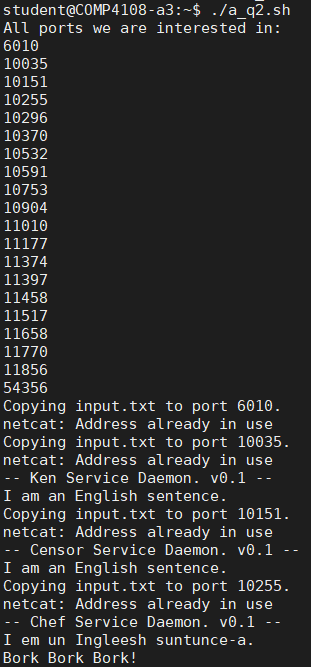
On line 12, I start netcat on the port. The -n flag tells netcat to not do any DNS or service lookups on any specified addresses, hostnames, or ports. I read from input.txt to transmit the file.

Next, I created my input.txt file:



The output of running my script is quite long. For this reason, I have a short snip of the output as a screenshot, and then a copy-paste of the full output below.

Here is the snipped screenshot:



Here is the full output of my script:

student@COMP4108-a3:~$ ./a\_q2.sh

All ports we are interested in:

6010

10035

10151

10255

10296

10370

10532

10591

10753

10904

11010

11177

11374

11397

11458

11517

11658

11770

11856

54356

Copying input.txt to port 6010.

netcat: Address already in use

Copying input.txt to port 10035.

netcat: Address already in use

-- Ken Service Daemon. v0.1 --

I am an English sentence.

Copying input.txt to port 10151.

netcat: Address already in use

-- Censor Service Daemon. v0.1 --

I am an English sentence.

Copying input.txt to port 10255.

netcat: Address already in use

-- Chef Service Daemon. v0.1 --

I em un Ingleesh suntunce-a.

Bork Bork Bork!

Copying input.txt to port 10296.

netcat: Address already in use

-- Cockney Service Daemon. v0.1 --

I am an English sentence.

Copying input.txt to port 10370.

netcat: Address already in use

-- Eleet Service Daemon. v0.1 --

1 4m 4n 3ngl15h 53nt3nc3.

Copying input.txt to port 10532.

netcat: Address already in use

-- Fanboy Service Daemon. v0.1 --

Copying input.txt to port 10591.

netcat: Address already in use

-- Fudd Service Daemon. v0.1 --

I am an Engwish sentence.

Copying input.txt to port 10753.

netcat: Address already in use

-- Jethro Service Daemon. v0.1 --

I wuz ah English sentence.

Copying input.txt to port 10904.

netcat: Address already in use

-- Jibberish Service Daemon. v0.1 --

Deerr

Frryund, 'a)ueffooes ys. Ye-a DEVGyes!!! Syeg 'iyl. Ye-a DEVGyes!!!76ooa ue-a fe-a . Ye-a DEVGyes!!!

Dys yst eh iyn \*cetnep breek!\* tyme-a oofffooh. Ye-a DEVGyes!!! Syeg 'iyl. Ye-a DEVGyes!!!

---

Tooh rr-rr-rreceyffe-a neyn foordooh na'yces oon dys meoolynkt lyst sund eh imevl meoott eh soobject ufff "rremevffe-a". Ye-a DEVGyes!!!{%R-rr-rrEND%}

Copying input.txt to port 11010.

netcat: Address already in use

-- Jive Service Daemon. v0.1 --

I's gots'ta be an English sentence. What it is, Mama!

Copying input.txt to port 11177.

netcat: Address already in use

-- Kraut Service Daemon. v0.1 --

I am an Englisch sentence.

Copying input.txt to port 11374.

netcat: Address already in use

-- Newspeak Service Daemon. v0.1 --

I am an English sentence.

Hail Big Brother!

Copying input.txt to port 11397.

netcat: Address already in use

-- Nyc Service Daemon. v0.1 --

I am an English sentence. Okay?

Copying input.txt to port 11458.

netcat: Address already in use

-- Pirate Service Daemon. v0.1 --

I be an English sentence.

Copying input.txt to port 11517.

netcat: Address already in use

-- Rasterman Service Daemon. v0.1 --

i am a nenglish sentence...

Copying input.txt to port 11658.

netcat: Address already in use

-- Scottish Service Daemon. v0.1 --

I am an 'Nglish sentence.

Copying input.txt to port 11770.

netcat: Address already in use

-- Spammer Service Daemon. v0.1 --

I am an Engrish sentence. WINNER! CALL NOW! This is a 1 time e-mail.

If you did not opt in for these mails please write back with a subject of REMOVE.

SSPLTM

Copying input.txt to port 11856.

netcat: Address already in use

-- Cowsay Service Daemon. v0.1 --

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

< I am an English sentence. >

---------------------------

\ ^\_\_^

\ (oo)\\_\_\_\_\_\_\_

(\_\_)\ )\/\

||----w |

|| ||

Copying input.txt to port 54356.

netcat: Address already in use

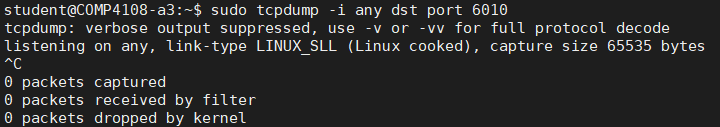
Done.

**Question 3**

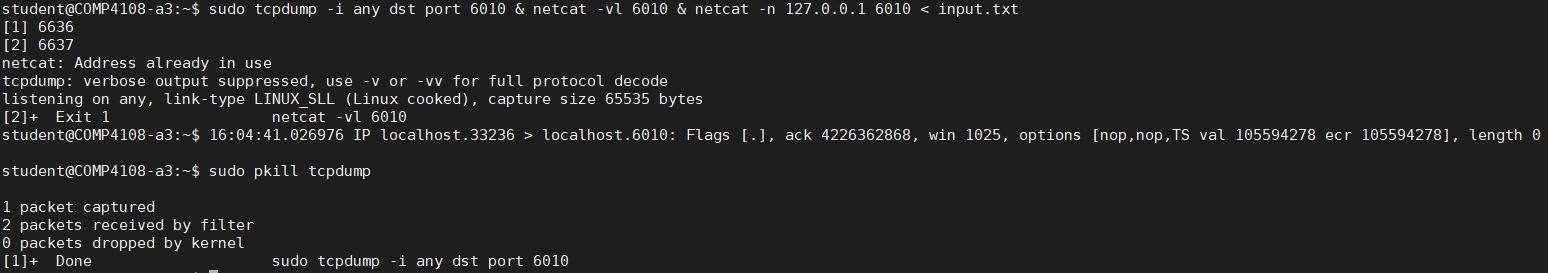
The tcpdump command for this question is sudo tcpdump -i any dst port 6010

The -i any tells tcpdump to match packets received on any interface, the dst port 6010 tells tcpdump to only match packets with 6010 as the destination port (6010 was one of the ports discovered in question 1).

If I run the command without sending anything, and then I interrupt it, this is what I see:



Here is a screenshot of what happens when I run tcpdump and netcat together:



The text in the screenshot might be a bit hard to see, so here is the output in text as well (since this is a PDF, you can also zoom in on the screenshot to better examine the output):

student@COMP4108-a3:~$ sudo tcpdump -i any dst port 6010 & netcat -vl 6010 & netcat -n 127.0.0.1 6010 < input.txt

[1] 6636

[2] 6637

netcat: Address already in use

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on any, link-type LINUX\_SLL (Linux cooked), capture size 65535 bytes

[2]+ Exit 1 netcat -vl 6010

student@COMP4108-a3:~$ 16:04:41.026976 IP localhost.33236 > localhost.6010: Flags [.], ack 4226362868, win 1025, options [nop,nop,TS val 105594278 ecr 105594278], length 0

student@COMP4108-a3:~$ sudo pkill tcpdump

1 packet captured

2 packets received by filter

0 packets dropped by kernel

[1]+ Done sudo tcpdump -i any dst port 6010

Let’s breakdown the command I ran.

I ran sudo tcpdump -i any dst port 6010 & netcat -vl 6010 & netcat -n 127.0.0.1 6010 < input.txt

The first command (sudo tcpdump -i any dst port 6010 &) was to run tcpdump in the background, listening for traffic from any protocol with 6010 as the destination port.

The second command (netcat -vl 6010 &) was to have netcat listen on port 6010, and run in the background.

The third command (netcat -n 127.0.0.1 6010 < input.txt) was to send input.txt to port 6010.

We can see the line

16:04:41.026976 IP localhost.33236 > localhost.6010: Flags [.], ack 4226362868, win 1025, options [nop,nop,TS val 105594278 ecr 105594278], length 0

was output from tcpdump regarding a captured packet.

Next, I ran sudo pkill tcpdump to kill the tcpdump process. (I had to use sudo to kill it, since I started it with sudo.)

Next, we see summary output from tcpdump, since tcpdump always prints a summary when it terminates. These lines were the summary output:

1 packet captured

2 packets received by filter

0 packets dropped by kernel

This means tcpdump captured one packet (which is the one that was displayed earlier in the output) out of the 2 packets that were received by the filter. tcpdump detected that no packets were dropped by the kernel.

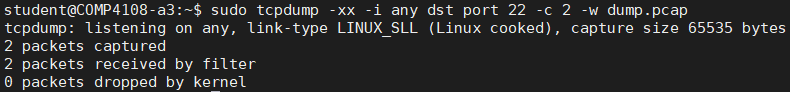
**Question 4**

From the tcpdump man page:

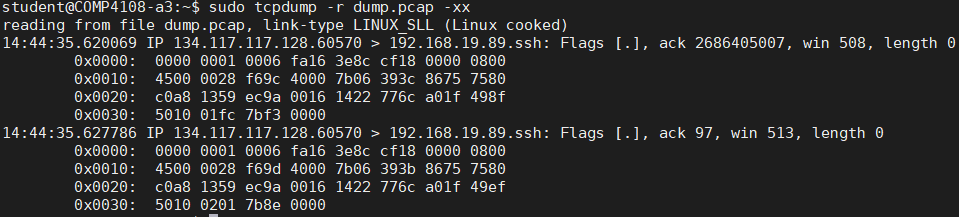


So I can use the -xx option to get the link-level headers and data of the packet in hex. Also from reading the man page, I learned that I can output to a file with -w.

Updated tcpdump command from previous question, with output:



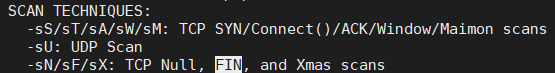
I can then read everything back out of the dump.pcap file like this:



The output file I created for this question is attached as data/dump.pcap.

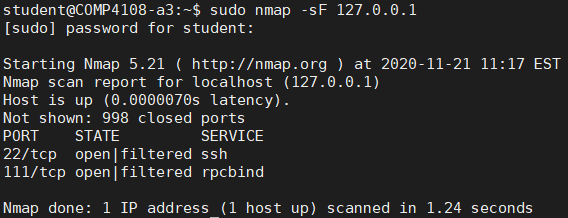
**Question 5**

On the nmap man page, it says the following:

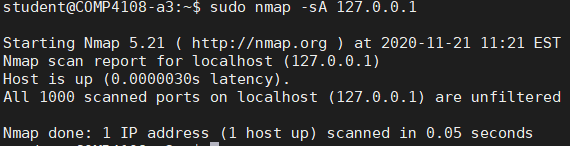


So I can use -sF to perform a TCP FIN scan, and -sA to perform a TCP ACK scan.

I ran an nmap TCP FIN scan like below:

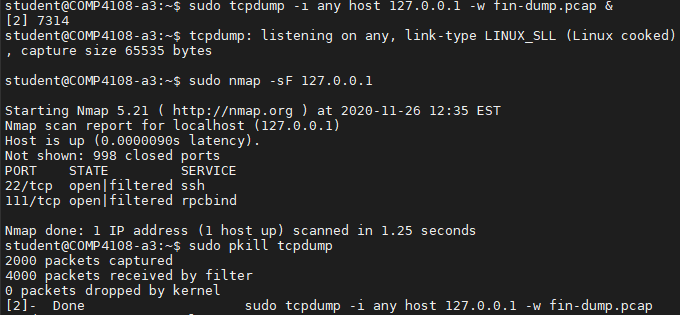


I ran an nmap TCP ACK scan like below:



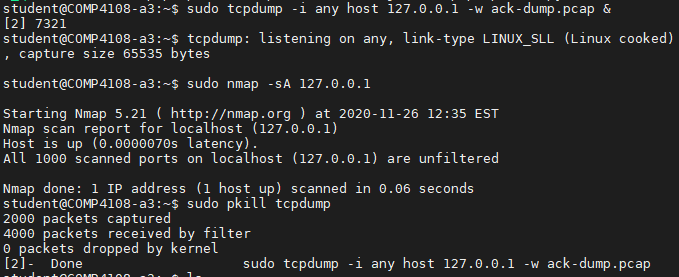
I can capture all incoming and outgoing packets for localhost by including host 127.0.0.1 in my tcpdump command, and I can specify the output file with the -w option. I can run my tcpdump command in the background (using bash’s & syntax), and then run my two nmap scans to capture all packets from both scans in separate files.

First, I captured the data for TCP FIN:

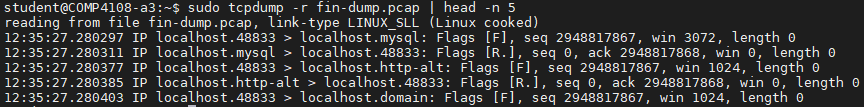


First, I ran tcpdump in the background, with fin-dump.pcap as the output file. Then I ran the TCP FIN scan. Finally, I used the pkill command kill the tcpdump process, which caused tcpdump to write to the fin-dump.pcap file, and exit.

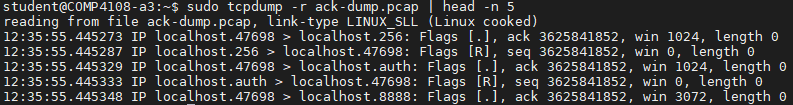
I repeated this for the TCP ACK scan, except using ack-dump.pcap as the output file:



Here is some of the output for the TCP FIN dump:



And here is some of the output for the TCP ACK dump:



In the TCP FIN dump snippet above, there is a pattern, where the first packet has [F] flags and a seq of 2948817867, and the second packet has [R.] flags, a seq of 0, and an ack of 2948817868. This repeats every two packets.

In the TCP ACK dump snippet above, there is a different pattern. The first packet has [.] flags and a seq of 3625841852, and the second packet has [R] flags and an ack of 3625841852 (the same number as for the seq of the previous packet). This also repeats every two packets.

A TCP FIN packet is used to terminate a TCP connection. A TCP ACK packet is used to acknowledge packets. TCP ACK packets are used to acknowledge TCP FIN packets (a TCP connection is not entirely terminated until both sides have sent a TCP FIN and received a TCP ACK).

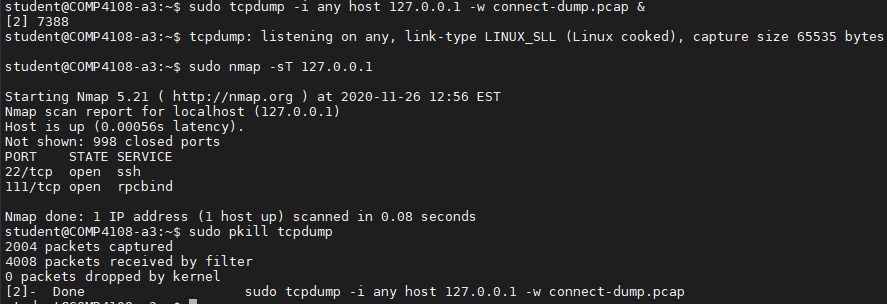
This means a TCP FIN scan is useful to monitor which applications are closing connections, and a TCP ACK scan is useful to monitor which applications are receiving close connection requests.

**Question 6**

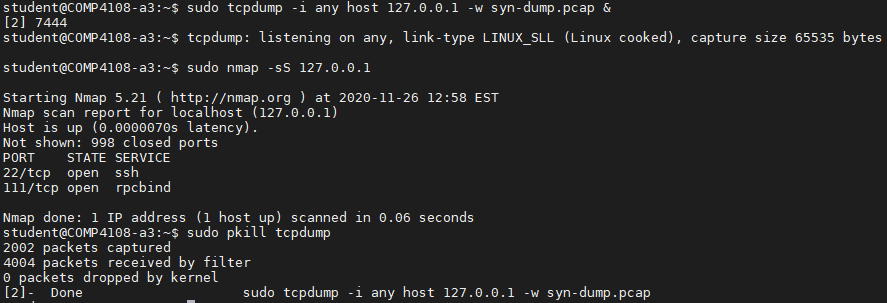
From reading the nmap man page, I learned I can scan for TCP connect()s with -sT, and I already knew from question 1 that I can scan for TCP SYNs with -sS.

I simply modified my commands from question 6 to scan for TCP connect() and TCP SYN.

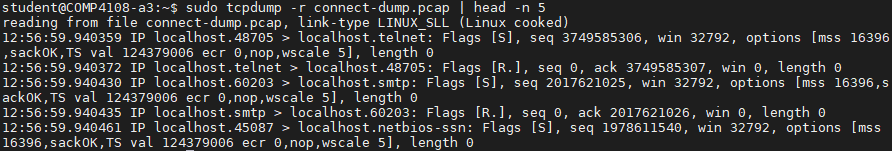
First I captured packets for TCP connect():



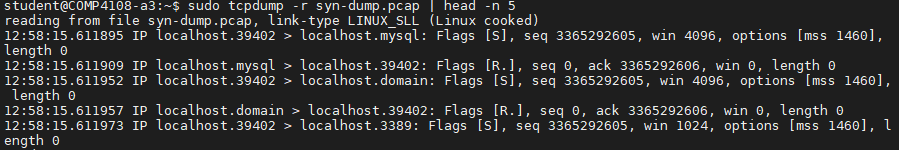
Then I captured packets for TCP SYN:



Here is some output from the TCP connect() dump:



And here is some output from the TCP SYN dump:



The pattern that repeats every two packets for the TCP connect() dump is the first packet has [S] flags, a seq of 3749585306, and a win of 32792; and the second packet has [R.] flags, a seq of 0, an ack, and a win of 0.

The pattern that repeats every two packets for the TCP SYN dump is the first packet has [S] flags, a seq of 3365292605, and a win; and the second packet has [R.] flags, a seq of 0, and an ack of 3365292606.

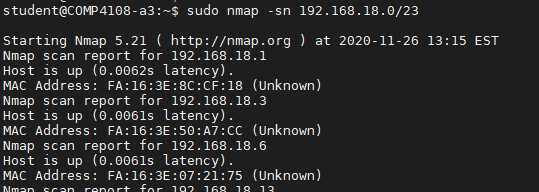
A TCP SYN packet is sent as part of the process of establishing a TCP connection. Once a SYN packet is received, the connection is considered “half open”. The host that received the SYN, if it is open, sends back a SYN/ACK packet, and then the first host would send an ACK packet to complete the TCP handshake, thus fully establishing a connection. A TCP connect(), then, is when the ACK packet is received, and the TCP connection is fully established.

Therefore, a TCP SYN scan can be used to monitor when TCP handshakes are initialized, and a TCP connect() scan can be used to monitor when TCP connections are established. The two can be used together to avoid something like TCP SYN flooding, since then half-open connections can be monitored (i.e., all cases where a TCP SYN was received but no TCP ACK is ever received to complete the TCP handshake).

**Question 7**

Question 7 Part A

Here is a truncated screenshot of what I ran and the output generated:



The -sn option for nmap is to perform a ping scan, and disable port scan.

Here is the full output:

student@COMP4108-a3:~$ sudo nmap -sn 192.168.18.0/23

Starting Nmap 5.21 ( http://nmap.org ) at 2020-11-26 13:15 EST

Nmap scan report for 192.168.18.1

Host is up (0.0062s latency).

MAC Address: FA:16:3E:8C:CF:18 (Unknown)

Nmap scan report for 192.168.18.3

Host is up (0.0061s latency).

MAC Address: FA:16:3E:50:A7:CC (Unknown)

Nmap scan report for 192.168.18.6

Host is up (0.0061s latency).

MAC Address: FA:16:3E:07:21:75 (Unknown)

Nmap scan report for 192.168.18.13

Host is up (0.0018s latency).

MAC Address: FA:16:3E:3B:F6:B3 (Unknown)

Nmap scan report for 192.168.18.14

Host is up (0.0018s latency).

MAC Address: FA:16:3E:74:61:75 (Unknown)

Nmap scan report for 192.168.18.26

Host is up (0.0030s latency).

MAC Address: FA:16:3E:21:51:72 (Unknown)

Nmap scan report for 192.168.18.38

Host is up (0.0028s latency).

MAC Address: FA:16:3E:FB:58:16 (Unknown)

Nmap scan report for 192.168.18.41

Host is up (0.0013s latency).

MAC Address: FA:16:3E:D5:C1:6A (Unknown)

Nmap scan report for 192.168.18.44

Host is up (0.0016s latency).

MAC Address: FA:16:3E:6A:DB:C7 (Unknown)

Nmap scan report for 192.168.18.50

Host is up (0.0024s latency).

MAC Address: FA:16:3E:33:D2:FE (Unknown)

Nmap scan report for 192.168.18.56

Host is up (0.0024s latency).

MAC Address: FA:16:3E:80:5D:BA (Unknown)

Nmap scan report for 192.168.18.59

Host is up (0.0029s latency).

MAC Address: FA:16:3E:13:31:AA (Unknown)

Nmap scan report for 192.168.18.77

Host is up (0.0048s latency).

MAC Address: FA:16:3E:1D:6A:A3 (Unknown)

Nmap scan report for 192.168.18.87

Host is up (0.0012s latency).

MAC Address: FA:16:3E:C0:B6:1B (Unknown)

Nmap scan report for 192.168.18.107

Host is up (0.0063s latency).

MAC Address: FA:16:3E:57:F0:F3 (Unknown)

Nmap scan report for 192.168.18.110

Host is up (0.0063s latency).

MAC Address: FA:16:3E:EC:1D:71 (Unknown)

Nmap scan report for 192.168.18.112

Host is up (0.0031s latency).

MAC Address: FA:16:3E:98:61:F6 (Unknown)

Nmap scan report for 192.168.18.114

Host is up (0.0030s latency).

MAC Address: FA:16:3E:AE:04:FE (Unknown)

Nmap scan report for 192.168.18.123

Host is up (0.0074s latency).

MAC Address: FA:16:3E:91:5C:88 (Unknown)

Nmap scan report for 192.168.18.124

Host is up (0.0074s latency).

MAC Address: FA:16:3E:91:83:6C (Unknown)

Nmap scan report for 192.168.18.128

Host is up (0.0074s latency).

MAC Address: FA:16:3E:45:87:2D (Unknown)

Nmap scan report for 192.168.18.132

Host is up (0.0074s latency).

MAC Address: FA:16:3E:24:E8:93 (Unknown)

Nmap scan report for 192.168.18.134

Host is up (0.0074s latency).

MAC Address: FA:16:3E:17:50:94 (Unknown)

Nmap scan report for 192.168.18.141

Host is up (0.0074s latency).

MAC Address: FA:16:3E:BB:E3:1A (Unknown)

Nmap scan report for 192.168.18.143

Host is up (0.0074s latency).

MAC Address: FA:16:3E:42:5C:6E (Unknown)

Nmap scan report for 192.168.18.147

Host is up (0.0040s latency).

MAC Address: FA:16:3E:74:8B:AD (Unknown)

Nmap scan report for 192.168.18.166

Host is up (0.0089s latency).

MAC Address: FA:16:3E:27:7B:AC (Unknown)

Nmap scan report for 192.168.18.175

Host is up (0.0089s latency).

MAC Address: FA:16:3E:1D:6D:DA (Unknown)

Nmap scan report for 192.168.18.181

Host is up (0.0089s latency).

MAC Address: FA:16:3E:AA:07:87 (Unknown)

Nmap scan report for 192.168.18.188

Host is up (0.0088s latency).

MAC Address: FA:16:3E:C5:77:9D (Unknown)

Nmap scan report for 192.168.18.193

Host is up (0.0088s latency).

MAC Address: FA:16:3E:60:DB:99 (Unknown)

Nmap scan report for 192.168.18.201

Host is up (0.0063s latency).

MAC Address: FA:16:3E:C3:0A:0F (Unknown)

Nmap scan report for 192.168.18.202

Host is up (0.0024s latency).

MAC Address: FA:16:3E:45:05:18 (Unknown)

Nmap scan report for 192.168.18.204

Host is up (0.0022s latency).

MAC Address: FA:16:3E:28:1E:3E (Unknown)

Nmap scan report for 192.168.18.207

Host is up (0.0022s latency).

MAC Address: FA:16:3E:3B:9E:57 (Unknown)

Nmap scan report for 192.168.18.228

Host is up (0.0075s latency).

MAC Address: FA:16:3E:FD:F1:E7 (Unknown)

Nmap scan report for 192.168.18.231

Host is up (0.0075s latency).

MAC Address: FA:16:3E:C2:E2:2B (Unknown)

Nmap scan report for 192.168.18.236

Host is up (0.0019s latency).

MAC Address: FA:16:3E:FA:B5:96 (Unknown)

Nmap scan report for 192.168.18.243

Host is up (0.0063s latency).

MAC Address: FA:16:3E:5D:20:63 (Unknown)

Nmap scan report for 192.168.18.248

Host is up (0.0063s latency).

MAC Address: FA:16:3E:FB:D5:04 (Unknown)

Nmap scan report for 192.168.19.15

Host is up (0.013s latency).

MAC Address: FA:16:3E:DD:4F:03 (Unknown)

Nmap scan report for 192.168.19.17

Host is up (0.013s latency).

MAC Address: FA:16:3E:56:62:C5 (Unknown)

Nmap scan report for 192.168.19.18

Host is up (0.013s latency).

MAC Address: FA:16:3E:45:5A:A9 (Unknown)

Nmap scan report for 192.168.19.21

Host is up (0.013s latency).

MAC Address: FA:16:3E:4F:C9:09 (Unknown)

Nmap scan report for 192.168.19.25

Host is up (0.013s latency).

MAC Address: FA:16:3E:B9:37:B5 (Unknown)

Nmap scan report for 192.168.19.31

Host is up (0.013s latency).

MAC Address: FA:16:3E:BF:05:E6 (Unknown)

Nmap scan report for 192.168.19.36

Host is up (0.013s latency).

MAC Address: FA:16:3E:12:7E:03 (Unknown)

Nmap scan report for 192.168.19.42

Host is up (0.013s latency).

MAC Address: FA:16:3E:BD:F8:91 (Unknown)

Nmap scan report for 192.168.19.45

Host is up (0.013s latency).

MAC Address: FA:16:3E:13:BF:20 (Unknown)

Nmap scan report for 192.168.19.47

Host is up (0.013s latency).

MAC Address: FA:16:3E:E5:A1:22 (Unknown)

Nmap scan report for 192.168.19.52

Host is up (0.0076s latency).

MAC Address: FA:16:3E:E4:34:0E (Unknown)

Nmap scan report for 192.168.19.61

Host is up (0.0027s latency).

MAC Address: FA:16:3E:3C:B5:B6 (Unknown)

Nmap scan report for 192.168.19.70

Host is up (0.0034s latency).

MAC Address: FA:16:3E:3A:D9:17 (Unknown)

Nmap scan report for 192.168.19.80

Host is up (0.0034s latency).

MAC Address: FA:16:3E:5D:7A:7E (Unknown)

Nmap scan report for 192.168.19.82

Host is up (0.0035s latency).

MAC Address: FA:16:3E:B9:D4:99 (Unknown)

Nmap scan report for 192.168.19.89

Host is up.

Nmap scan report for 192.168.19.92

Host is up (0.0031s latency).

MAC Address: FA:16:3E:00:AA:1E (Unknown)

Nmap scan report for 192.168.19.99

Host is up (0.0036s latency).

MAC Address: FA:16:3E:E6:E1:E3 (Unknown)

Nmap scan report for 192.168.19.111

Host is up (0.0032s latency).

MAC Address: FA:16:3E:8D:9E:DC (Unknown)

Nmap scan report for 192.168.19.115

Host is up (0.0030s latency).

MAC Address: FA:16:3E:18:EC:41 (Unknown)

Nmap scan report for 192.168.19.129

Host is up (0.0022s latency).

MAC Address: FA:16:3E:12:89:E3 (Unknown)

Nmap scan report for 192.168.19.135

Host is up (0.0027s latency).

MAC Address: FA:16:3E:95:84:84 (Unknown)

Nmap scan report for 192.168.19.136

Host is up (0.0027s latency).

MAC Address: FA:16:3E:BE:28:18 (Unknown)

Nmap scan report for 192.168.19.151

Host is up (0.0032s latency).

MAC Address: FA:16:3E:EC:7E:DE (Unknown)

Nmap scan report for 192.168.19.156

Host is up (0.0032s latency).

MAC Address: FA:16:3E:5E:9E:9F (Unknown)

Nmap scan report for 192.168.19.157

Host is up (0.0032s latency).

MAC Address: FA:16:3E:91:4D:10 (Unknown)

Nmap scan report for 192.168.19.159

Host is up (0.0032s latency).

MAC Address: FA:16:3E:C7:0D:48 (Unknown)

Nmap scan report for 192.168.19.162

Host is up (0.0035s latency).

MAC Address: FA:16:3E:7C:15:4B (Unknown)

Nmap scan report for 192.168.19.168

Host is up (0.0063s latency).

MAC Address: FA:16:3E:E1:5A:3E (Unknown)

Nmap scan report for 192.168.19.175

Host is up (0.0063s latency).

MAC Address: FA:16:3E:12:52:3F (Unknown)

Nmap scan report for 192.168.19.177

Host is up (0.0063s latency).

MAC Address: FA:16:3E:A0:7E:DA (Unknown)

Nmap scan report for 192.168.19.188

Host is up (0.0063s latency).

MAC Address: FA:16:3E:B0:E1:EF (Unknown)

Nmap scan report for 192.168.19.193

Host is up (0.0062s latency).

MAC Address: FA:16:3E:D5:68:30 (Unknown)

Nmap scan report for 192.168.19.200

Host is up (0.0025s latency).

MAC Address: FA:16:3E:35:02:4D (Unknown)

Nmap scan report for 192.168.19.211

Host is up (0.0026s latency).

MAC Address: FA:16:3E:5D:52:87 (Unknown)

Nmap scan report for 192.168.19.213

Host is up (0.0030s latency).

MAC Address: FA:16:3E:88:F3:C2 (Unknown)

Nmap scan report for 192.168.19.219

Host is up (0.0052s latency).

MAC Address: FA:16:3E:32:42:D6 (Unknown)

Nmap scan report for 192.168.19.220

Host is up (0.0052s latency).

MAC Address: FA:16:3E:3C:E4:0B (Unknown)

Nmap scan report for 192.168.19.241

Host is up (0.0052s latency).

MAC Address: FA:16:3E:88:F3:71 (Unknown)

Nmap scan report for 192.168.19.242

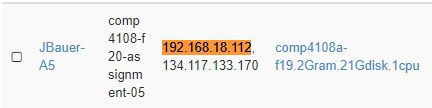
Host is up (0.0052s latency).

MAC Address: FA:16:3E:87:37:F3 (Unknown)

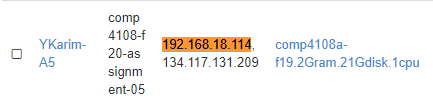
Nmap done: 512 IP addresses (80 hosts up) scanned in 1.78 seconds

Question 7 Part B

All of the IP addresses in the scan output are either 192.168.18.X or 192.168.19.X IP addresses. From looking at our OpenStack instances, all of them have internal addresses of either 192.168.18.X or 192.168.19.X. In particular, 192.168.18.112 was one of the IP addresses found in the scan, and this matches the IP address for the instance “JBAuer-A5”:



This is also true for another IP address from the scan, 192.168.18.114:

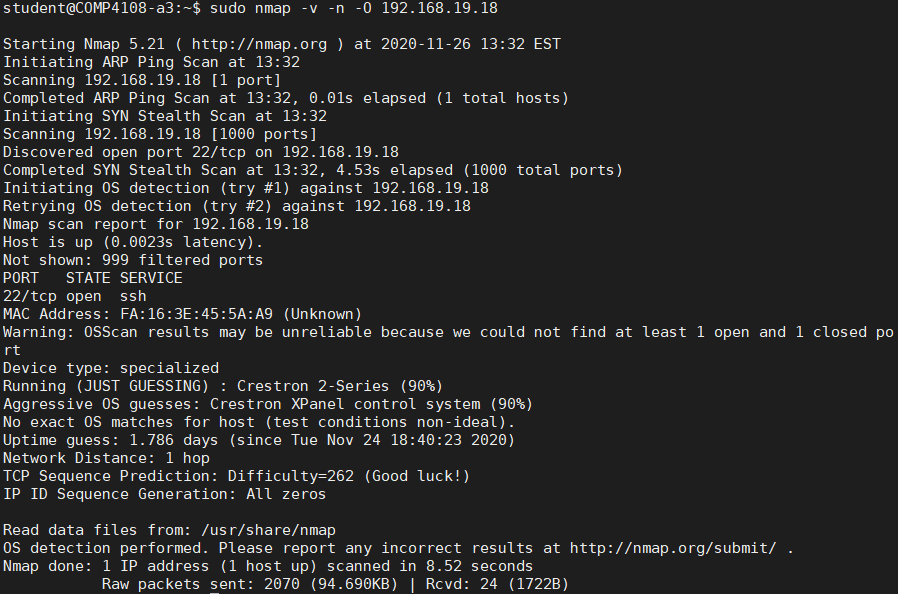


In fact, at the time of scanning, there were 79 active instances in OpenStack, and the nmap scan found 80 hosts. I did not check exhaustively, but I imagine all 80 hosts found by nmap map to the 79 hosts (plus 1 mystery host) in our OpenStack for this course.

Question 7 Part C

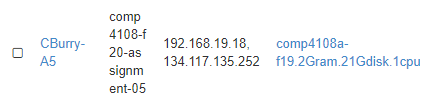
I can use -v for verbose, -n to disable DNS resolution (I read this speeds up nmap a bit), and -O to enable OS detection.

I picked 192.168.19.18 as the host to scan, and this is the command I ran and its output:



My hypothesis in part B of this question was that the IP addresses were for the OpenStack instances. This scan failed to conclusively determine an OS, as nmap could not find at least 1 open and 1 closed port. nmap guessed the OS is Crestron, but since it was only a guess, this does not confirm or deny my hypothesis (the result would have to be more conclusive to confirm or deny my hypothesis).

Nonetheless, my hypothesis is most likely correct, as the “CBurry-A5” OpenStack instance has an internal IP address of 192.168.19.18:

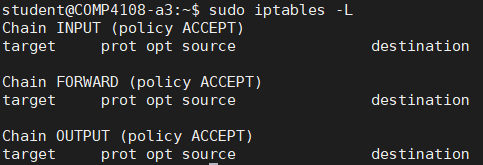


# Part B

**Question 1**

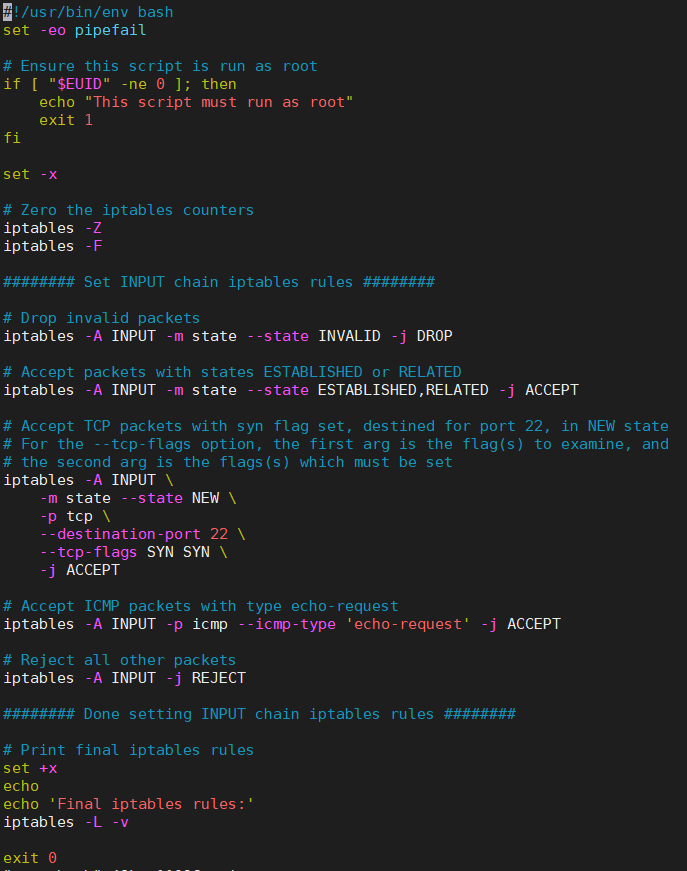
From Ubuntu’s Iptables How To wiki (<https://help.ubuntu.com/community/IptablesHowTo>) I learned that sudo iptables -L lists all current rules in iptables.

I ran this command to list all the active rules in the INPUT, OUTPUT, and FORWARD chains of iptables from your VM before you have added any rules. Submit the command you ran as well as the output:



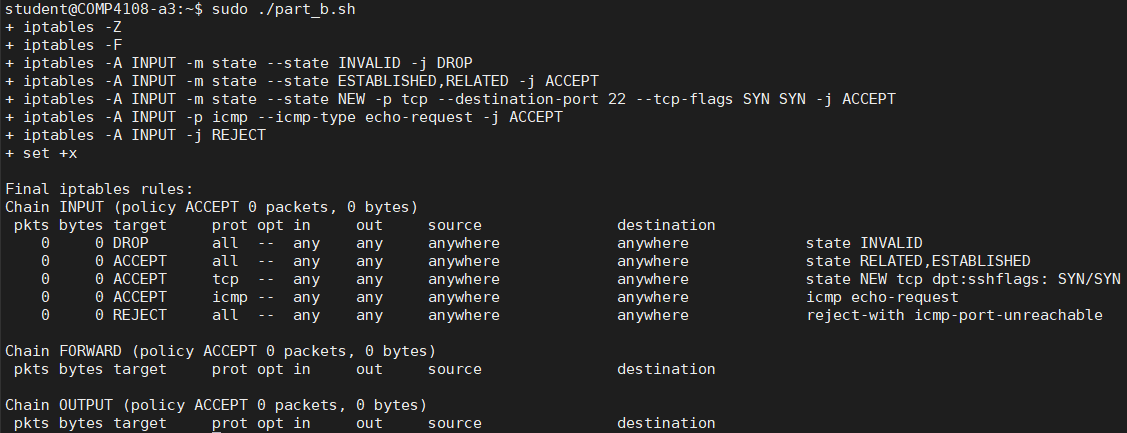
**Question 2**

See code/partb.sh for my script. Below is also a screenshot of my script:

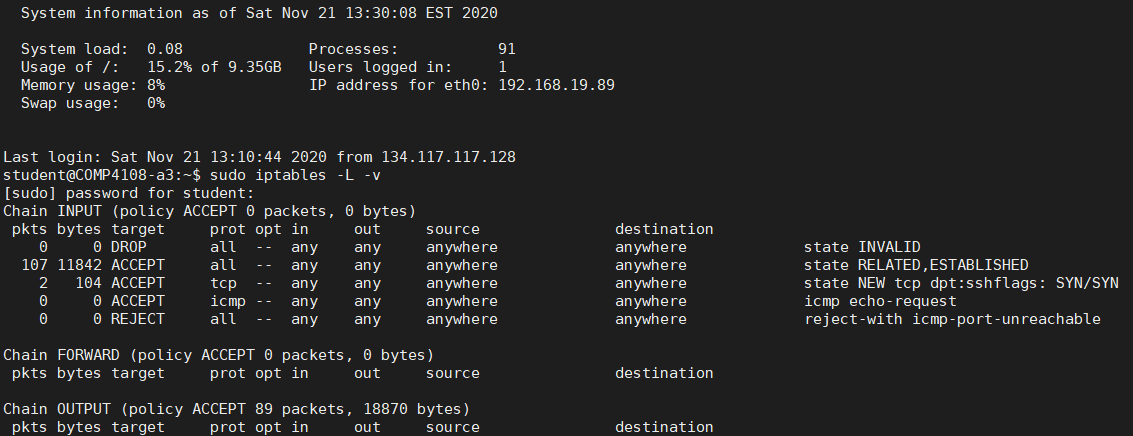


**Question 3**

I ran my script like so:

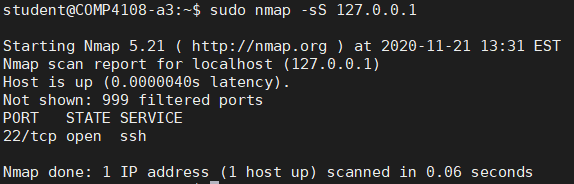


I was able to establish a second ssh session:

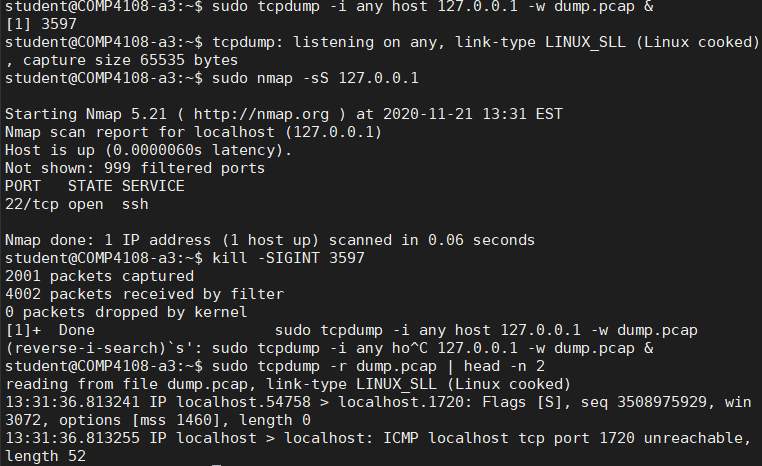


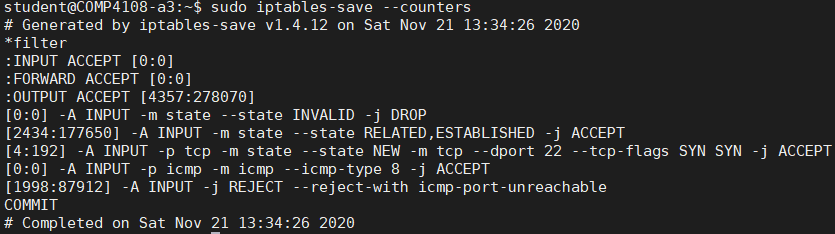
In this screenshot, I have evidence that 1 user (myself) was already logged in (see the “Users logged in” above), and I also have evidence that the iptables rules were in effect when I established the second ssh session (see the output of sudo iptables -L -v above).

I was able to run my nmap command from Part A, Q1:



I was able to capture packets from the nmap scan to a file using tcdump, as before (I used head with the final command to truncate output; ignore the “reverse-i-search” line in the output, that was an entry error I made, and is not relevant to the rest of the output):

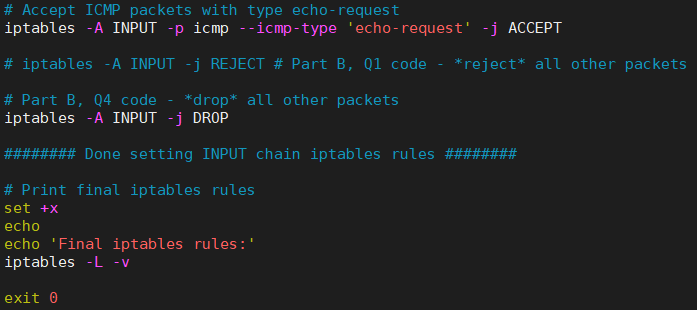


I viewed the list of active firewall rules, including their counter values, using iptables-save, as below: 

This is likely obvious, but the --counters option causes iptables-save to display the counter values.

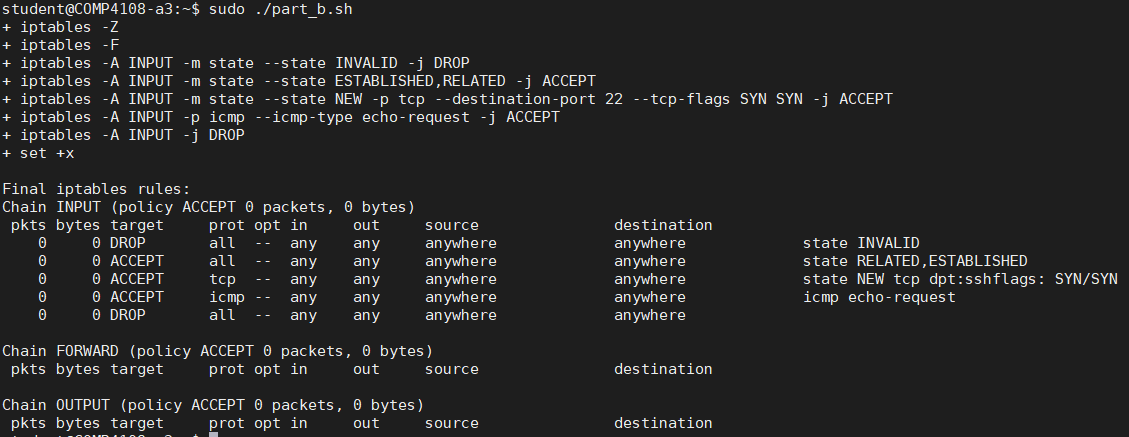
**Question 4**

My script is almost identical for this question, but I commented out the last iptables command and replaced it, as shown in the screenshot below (again, the script’s source code can be accessed at code/part\_b.sh):



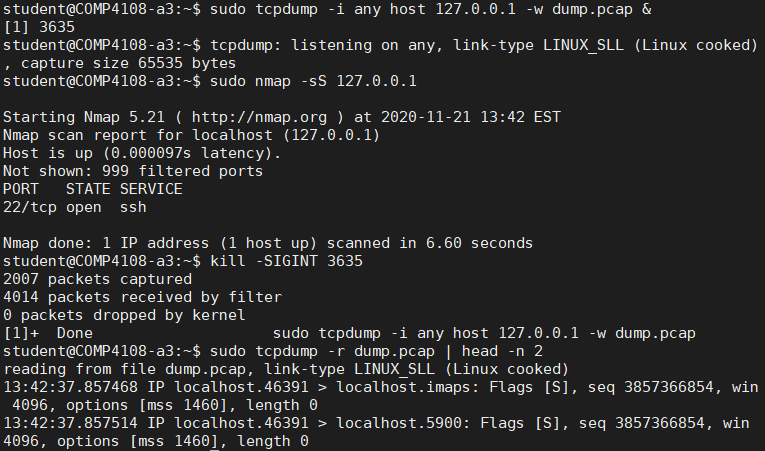
As can be seen, I changed from rejecting all packets that did not meet any other rule, to dropping all such packets.

I ran my script like so:



As can be seen, the rule is now to drop, rather than reject, all packets that do not match the first four rules.

I ran nmap and tcpdump with the updated rules:



Looking at the output of tcpdump when REJECT is set, there are several lines of the form

13:46:42.572151 IP localhost > localhost: ICMP localhost tcp port 8888 unreachable, length 52

In fact, 999 lines have the string “unreachable” (wc -l counts the number of lines fed to it):



Looking at the output of tcpdump when DROP is set, there are no lines that indicate something was unreachable:

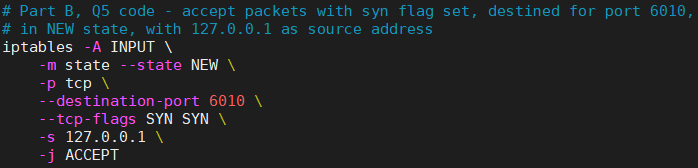


The difference between drop and reject behaviour is that, when REJECT is set, the other end receives a response indicating that the packet was rejected. Conversely, when DROP is set instead of REJECT, the packet is simply dropped, and the other end does not receive a response (hence why there are no “unreachable” messages with the drop-dump.pcap file).

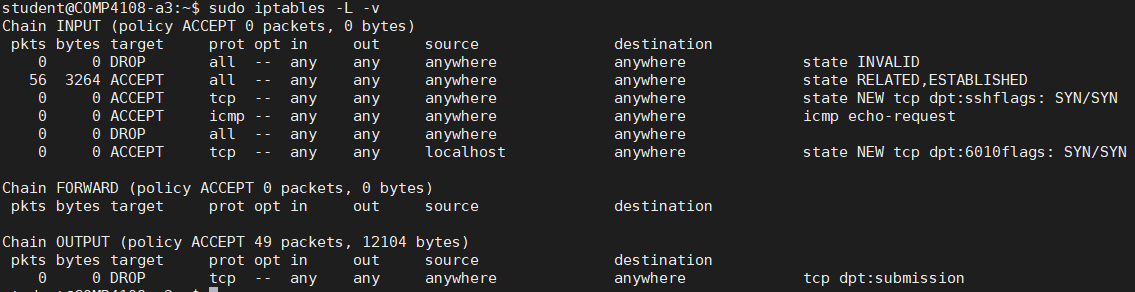
**Question 5**

Most of the options I needed to create this new rule I already knew from part D of question 2. From the iptables man page, I read that the -s option can be used to set the source address in the rule. I chose 6010 as my port of choice from the ports discovered in part A question 1.

Here is a screenshot of what I added to this script for this question:



After rerunning my script, I see the following rules are set:

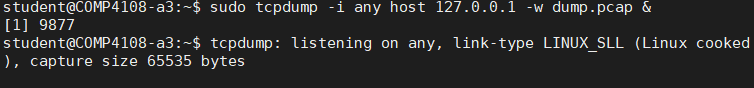


In particular, we can see TCP packets with SYN flag, etc. are set for port 6010:

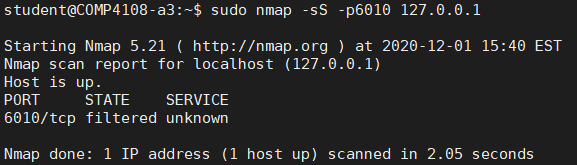


To test my new rule, I used nmap on the port, and captured the packets with tcpdump.

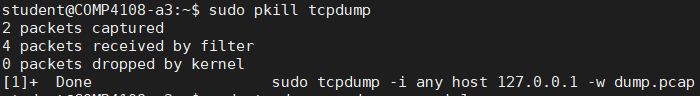
First, I started tcpdump in the background, writing output to dump.pcap:



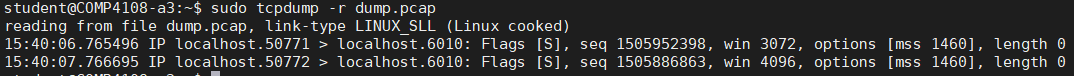
Then I ran nmap against port 6010:



Then I killed the tcpdump process with pkill (so it would flush its output and terminate):



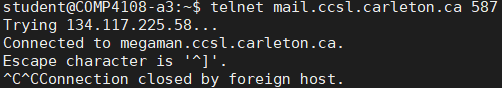
Finally, I viewed the captured packet data (this is the full, non-truncated output):



As can be seen, only the TCP SYN packets made it through. We know this because the flags is[S], and there are no packets recorded with any other flags.

**Question 6**

This is what I saw when I telnet’d mail.ccsl.carleton.ca initially, before updating my firewall script:

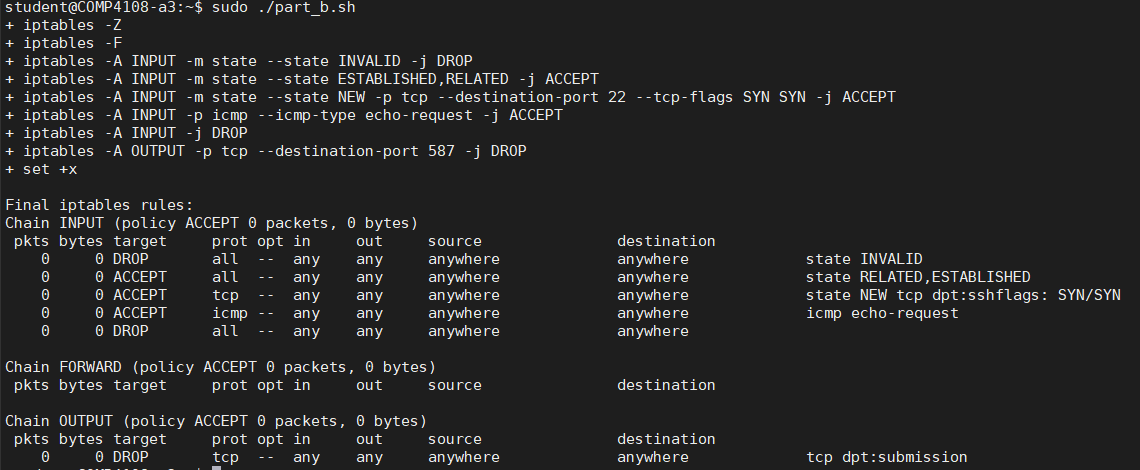


As can be seen, connection was successful, and then the connection was closed.

I added the following to my script, after all of my INPUT rules:



I ran my updated script like so:



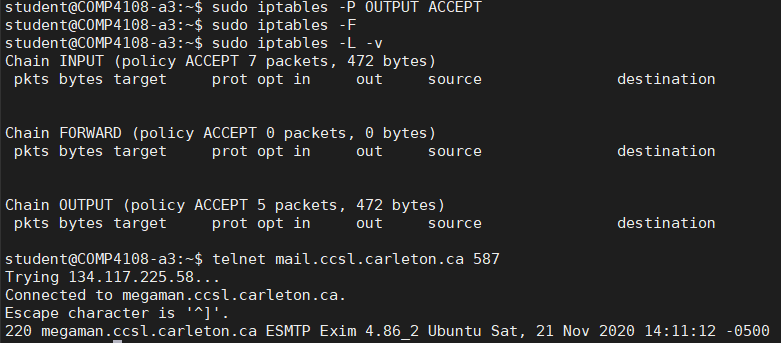
We can see from the output that now there is an entry in the OUTPUT chain.

I then attempted to telnet as before:



The telnet process just hung (since it failed to establish the connection, due to the packets being dropped), so I was forced to Ctrl+C to terminate the process.

I was then able to restore connectivity by resetting the OUTPUT chain rules:



Here I set all OUTPUT rules to ACCEPT (sudo iptables -P OUTPUT ACCEPT). Then I flush my changes (sudo iptables -F) and print the updated iptables rules (sudo iptables -L -v). Then I am able to re-establish a connection with telnet, as seen from the output of my last command.