CSC 347

# A4 Report

Arshia Gharai and Michael De Lisio

4th December, 2019

## Part A

**1.** The nmap Unix command line tool can be used to scan all ports on a host/server machine. By default nmap uses the TCP SYN scan technique along with a scanning range of 1,000 of the most common ports on the machine. Since we require nmap to scan every port (1 through $2^{16}$ -1). The port specification option `-p-` scans ports 1 to 65535 which is sufficient given it only leaves out port 0 (the wildcard port) - which isn't of much interest. The option `-sS` enforces a TCP SYN scan, while the output option `--open` only outputs scanned open ports. Since TCP SYN requires root priviledges we also need to run the command with sudo.

The command used: `sudo nmap -sS -p- --open localhost`
**The output from the command above is located in part_a/q1.txt):**

**2.** The script in subdirectory PartA under the name **q2_script.sh** takes two arguments, nmap output file and a message file, respectively. The bash script parses the nmap output and for every open port it finds, it executes the command: `netcat localhost $port < $message_file`, where `$port` is the port number of an open TCP port and `$message_file` is the file path containing the message to send to the specified port. The script emits the netcat output on success and "`Netcat unsuccessful for port: $port`" when netcat exits with return code not equal to 0. In the example below "input.txt" contains the sentence "Hello World!".

Run: `./q2_script.sh q1.txt info.txt`
**The output from the command above is located in part_a/q2.txt.**

**3.** To meet the criterion that tcpdump should collect packets from all interfaces we need to include the option `-i any.` Also, to meet the criterion that tcpdump should intercept packets from a service served on one of the ports generated from the question 1's nmap command we must use an expression with type qualifiers. The type qualifiers we used were "host" and "port". In the example below we used port 10093 since this was the first port that produced a message after netcat initiated a connection.

Tcpdump command: `sudo tcpdump -i any host localhost and port 10093`
In another shell run: `sudo netcat localhost 10093 < input.txt`
**The output from the first command above is located in part_a/q3.txt.**

**4.** The only difference in this command between the command used in question 3 and the one used here in the use of the `-x` option. This option prints the headers and data packets in hex as per the handout. We also use -w to print the raw binary data to a file

Tcpdump command: `sudo tcpdump -i any -x -w q4.pcap host localhost and port 10093`
In another shell run: `sudo netcat localhost 10093 < input.txt`
`Tcpdump print command: tcpdump -x -r q4.pcap >> q4.txt`
**The human readable tcpdump output from the third command above is located in part_a/q4.txt.**

**5.**

ACK SCAN Script **(part_a/q5.txt)**:

```
sudo tcpdump -i any -x -w q5_ACK.pcap host localhost and port
10466
# In another shell run: sudo nmap -sA -p- localhost
tcpdump -x -r q5_ACK.pcap >> q5.txt # append output to this file
```

*Excerpt from* ***`part_a/q5.txt`****:*

```
00:31:16.252277 IP localhost.48384 > localhost.10466: Flags [.], ack 3459124212, win 1024, length
0
        0x0000:  4500 0028 8103 0000 3906 02cb 7f00 0001
        0x0010:  7f00 0001 bd00 28e2 0000 0000 ce2e 0bf4
        0x0020:  5010 0400 edcc 0000
00:31:16.252318 IP localhost.10466 > localhost.48384: Flags [R], seq 3459124212, win 0, length 0
        0x0000:  4500 0028 f2f2 4000 4006 49db 7f00 0001
        0x0010:  7f00 0001 28e2 bd00 ce2e 0bf4 0000 0000
        0x0020:  5004 0000 f1d8 0000
```

The excerpt above shows an exchange between ports 48384 and 10466, where port 48384 sends packet with the “acknowledgment” bit set high (the “.” indicates the ACK bit is set high) to the receiver. Ports that respond to ACK requests are either open or closed (unfiltered) while a port that doesn’t respond is a filtered port hosted on a machine with, most probably, a firewall enabled. From the excerpt, we deduce that the receiver responds with a packet that has its RST (reset) bit set high. Therefore, we can assume the receiving machine (really just our local machine) does not have a firewall enabled. From the nmap manpage “The TCP ACK scan is used to map out firewall rulesets”, which confirms this.

FIN SCAN Script **(part_a/q5.txt):**

```
sudo tcpdump -i any -x -w q5_FIN.pcap host localhost and port
10093
# In another shell run: sudo nmap -sF -p- localhost
tcpdump -x -r q5_FIN.pcap >> q5.txt # append output to file
```

*Excerpt from **part_a/q5.txt**:*

```
00:37:39.573928 IP localhost.42543 > localhost.10093: Flags [F], seq 1867125843, win 1024,
length 0
        0x0000:  4500 0028 c9b6 0000 2806 cb17 7f00 0001
        0x0010:  7f00 0001 a62e 459d 6f4b 1452 0000 0000
        0x0020:  5001 0400 3e78 0000
```

The excerpt above reveals that nmaps TCP FIN scan sets the FIN TCP bit high when interacting with ports on the receiving machine and sets the ACK, SYN and RST bits low. This scan attempts to detect firewalls that reject TCP packets with the SYN bit set high and ACK set low, such ports have an enabled firewall in place or are closed. From the excerpt the port issuing the request (port 42542) fails to receive a response packet from the receiving port, indicating either the receiving port is open or is filtered (as per nmap’s manpage). Couple this result with the previous result from the ACK scan and we can deduce that this port is in fact open since there does not seem to be a firewall enabled on the receiving machine.

**6.**

SYN SCAN Script **(part_a/q6.txt):**

```
sudo tcpdump -i any -x -w q6_SYN.pcap host localhost and portrange 10093-10095
# In another shell run: sudo nmap -sS -p- localhost
tcpdump -x -r q6_SYN.pcap >> q6.txt # append output to this file as stated in the handout
```

*Excerpt from **part_a/q6.txt**:*

```
02:11:58.550947 IP localhost.45458 > localhost.10093: Flags [S], seq 2090535651, win 1024, options [mss 1460], length 0
	0x0000:  4500 002c 5216 0000 2606 44b4 7f00 0001
	0x0010:  7f00 0001 b192 276d 7c9b 0ae3 0000 0000
	0x0020:  6002 0400 35a6 0000 0204 05b4
02:11:58.550992 IP localhost.10093 > localhost.45458: Flags [S.], seq 1375004364, ack 2090535652, win 32792, options [mss 16396], length 0
	0x0000:  4500 002c 0000 4000 4006 3cca 7f00 0001
	0x0010:  7f00 0001 276d b192 51f4 e6cc 7c9b 0ae4
	0x0020:  6012 8018 fe20 0000 0204 400c
```

As you might infer from the excerpt above SYN scans are useful for detecting open/closed ports. The issuing port () sends a TCP packet with the SYN bit high (a SYN packet always initiates a connection between two machines), a response of a SYN and ACK TCP packet indicates the port is open and a RST TCP packet implies the non-scanning port is closed. In the excerpt we see that the receiving port (10093) issues a packet with the ACK and SYN bit set, therefore it can interpreted as an open port. This scan is particularly fast as it doesn't need to set up a connection with the client to perform the scan.

CONNECT SCAN

```
sudo tcpdump -i any -x -w q6_CNT.pcap host localhost and portrange 10093-10095
# In another shell run: sudo nmap -sT -p- localhost
tcpdump -x -r q6_CNT.pcap >> q6.txt # append output to this file as stated in the handout
```

*Excerpt from* **`part_a/q6.txt`**

```
02:32:59.390095 IP localhost.37230 > localhost.10093: Flags [S], seq 3527888044, win 32792,
options [mss 16396,sackOK,TS val 8605862 ecr 0,nop,wscale 3], length 0
        0x0000:  4500 003c 1c0a 4000 4006 20b0 7f00 0001
        0x0010:  7f00 0001 916e 276d d247 4cac 0000 0000
        0x0020:  a002 8018 fe30 0000 0204 400c 0402 080a
        0x0030:  0083 50a6 0000 0000 0103 0303
02:32:59.390148  IP  localhost.10093  >  localhost.37230:  Flags  [S.],  seq  3900343730,  ack
3527888045, win 32768, options [mss 16396,sackOK,TS val 8605862 ecr 8605862,nop,wscale 3],
length 0
        0x0000:  4500 003c 0000 4000 4006 3cba 7f00 0001
        0x0010:  7f00 0001 276d 916e e87a 85b2 d247 4cad
        0x0020:  a012 8000 fe30 0000 0204 400c 0402 080a
        0x0030:  0083 50a6 0083 50a6 0103 0303
02:32:59.390190 IP localhost.37230 > localhost.10093: Flags [.], ack 1, win 4099, options
[nop,nop,TS val 8605862 ecr 8605862], length 0
        0x0000:  4500 0034 1c0b 4000 4006 20b7 7f00 0001
        0x0010:  7f00 0001 916e 276d d247 4cad e87a 85b3
        0x0020:  8010 1003 fe28 0000 0101 080a 0083 50a6
        0x0030:  0083 50a6
02:32:59.399594 IP localhost.37230 > localhost.10093: Flags [R.], seq 1, ack 1, win 4099,
options [nop,nop,TS val 8605864 ecr 8605862], length 0
        0x0000:  4500 0034 1c0c 4000 4006 20b6 7f00 0001
        0x0010:  7f00 0001 916e 276d d247 4cad e87a 85b3
        0x0020:  8014 1003 fe28 0000 0101 080a 0083 50a8
        0x0030:  0083 50a6
```

TCP connect scan is used when the user running nmap does not have the privileges to open a raw socket (root privileges). This scan is more inefficient than SYN while providing the same purpose, since it calls the system call connect (therefore requiring to establish a connection with the client to determine information). This can be seen through the excerpt, as instead of only requiring 2 packets to be sent, now we require a connection setup costing us 2 more packets to be sent. The example above shows what happens when a client port is open. **`part_a/q6.txt`** also contains an example of a closed client port (for ports 10094 and 10095), where there are only two packets sent in total. Only two packets are sent with a closed client port, since after the client receives a SYN packet it immediately will send a RST packet (to indicate it didn't expect the first packet).

## <u>7a.</u>

The command used: `sudo nmap -sn 142.1.46.*`

**The output from the command above is located in part_a/q7.txt):**

## <u>7b.</u>

*Excerpt from* ***part_a/q7.txt:***

```
Nmap scan report for dh2026pc00.erin.utoronto.ca (142.1.46.34)
Host is up (0.0027s latency).
Nmap scan report for dh2020pc01.erin.utoronto.ca (142.1.46.35)
Host is up (0.0026s latency).
```

Nmap provides the -sn option to be used for host discovery scans without performing a port scan. This option also determines whether the host of one of the ip addresses (indicated by range expression 142.1.8.*) is running. From the excerpt above these hosts generated by nmap seem to be the DeerField Hall computers along with other servers running under the "erin.utoronto.ca" network.

## 8.

The command used: `sudo nmap -sA -p- 142.1.44.135`
**The output from the command above is located in part_a/q8.txt):**

*Excerpt from* ***q8.txt****:*

```
PORT      STATE SERVICE
902/tcp   open  iss-realsecure
2222/tcp  open  EtherNet/IP-1
25565/tcp open  minecraft
26000/tcp open  quake
```

Nmap uses the "nmap-service table" when classifying services under well-known/registered ports (ports in range 0-1023 and 1024-49151 repestively). We checked the reference at https://svn.nmap.org/nmap/nmap-services and found that unsurprisingly the ports and services match the table. Though, this system is not perfect as clients sometimes use well-known or registered ports for uncommon services to evade firewalls or hide from attackers (stated in nmap's manpage).

To test whether these services are correctly being identified we used the same script from question 2 (except it now connects to ip address: 142.1.44.135 instead of locahost - script located in **part_a/prt8_script.sh**).

Script command used: ./q8_script.sh q8.txt input.txt
**Note: all excerpts/output referenced below can be found in directory part_a under file name q8_script.txt.**

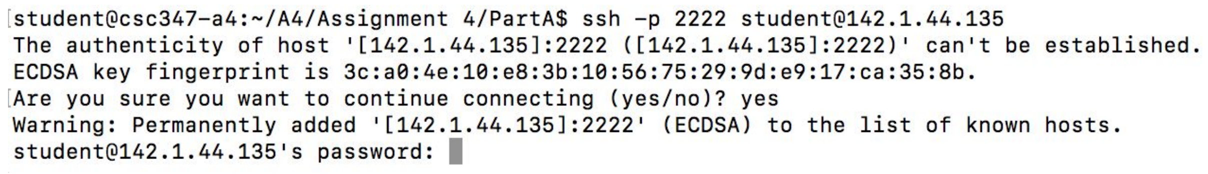
Investigating Port: 902

Upon sending a TCP packet via netcat to port 902 we receive a message containing: “VMware Authentication Daemon Version 1.10”**.** From this excerpt it seems this port serves a VMWare Authentication Daemon (rather than iss-realsecure application) from its response header.

Investigating Port: 2222

Upon sending a TCP packet via netcat to port 902 we receive a message containing: “SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8”.From the excerpt above it seems this port serves a VMWare Authentication Daemon (rather than iss-realsecure application) from its response header.

Screenshot of 142.1.42.135: 2222 ssh login in terminal:

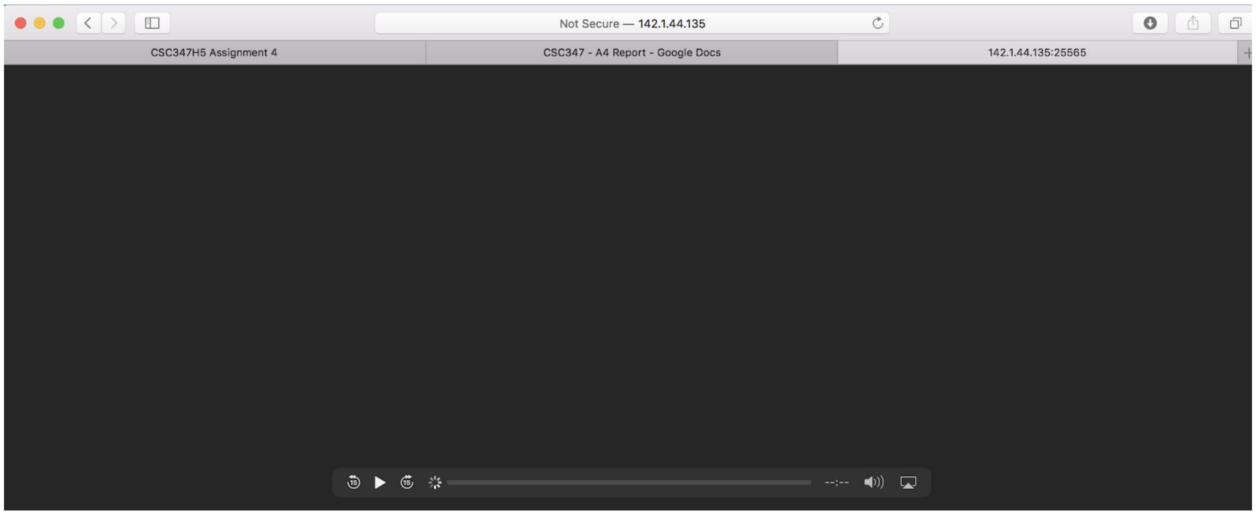


```
student@csc347-a4:~/A4/Assignment 4/PartA$ ssh -p 2222 student@142.1.44.135
The authenticity of host '[142.1.44.135]:2222 ([142.1.44.135]:2222)' can't be established.
ECDSA key fingerprint is 3c:a0:4e:10:e8:3b:10:56:75:29:9d:e9:17:ca:35:8b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[142.1.44.135]:2222' (ECDSA) to the list of known hosts.
student@142.1.44.135's password: 
```

The screenshot confirms that port 2222 serves ssh since we complete a successful key exchange and add the host to my local machines hosts file.

Investigating Port: 25565

Screenshot of 142.1.42.135: 25565 webpage:

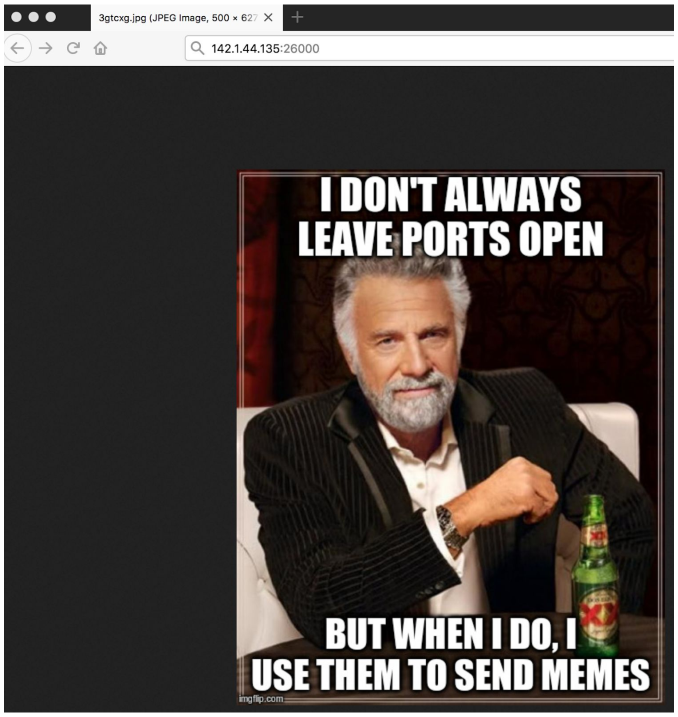


When entering the address into a safari browser we get a media player. This is contrary to nmap's determination that this port serves minecraft.

<u>Investigating Port: 26000</u>

Upon sending a TCP packet via netcat to port 26000 we receive a message containing: "<p>Message: Bad HTTP/0.9 request type ('Hello')". The excerpt containing HTML above reveals that port 26000 is serving a HTTP webpage. After entering the ip address and port of this service into a web browser we obtained the screenshot below.

Screenshot of 142.1.42.135:26000 webpage:


3gtcxg.jpg (JPEG Image, 500 × 627
142.1.44.135:26000
I DON'T ALWAYS LEAVE PORTS OPEN
BUT WHEN I DO, I USE THEM TO SEND MEMES
imgflip.com

## Part B

1) List the active rules in the INPUT, OUTPUT, and FORWARD chains of iptables:
    - **Command ran:** sudo iptables -L
    - **Output:**

```
Chain INPUT (policy ACCEPT)
target   prot opt source          destination


Chain FORWARD (policy ACCEPT)
target   prot opt source          destination


Chain OUTPUT (policy ACCEPT)
target   prot opt source          destination
```

2) In general, the rules follow a common format with some things being different according to the instructions. The firewall script file in its entirety can be found on markus under `part_b/q2.sh.`

3) The list of active firewall rules (including their counter values) as generated by `iptables-save`, as well as the results of the nmap scan can be found on markus under `part_b/q3.txt`

4) Just changed the reject to drop in the last rule. The difference between the drop and reject is that when a packet is dropped, there is no response sent back to the sender of the packet. On the other hand, when a packet is rejected, an ICMP destination-unreachable packet is sent back to the host letting them know that it was rejected. In an actual attack, using drop or reject will not matter since they both advantages and disadvantages. Moreover, using reject is more time consuming since ICMP packet needs to be sent back to the sender whereas for drop, no extra signal is needed.

5) A shell transcript containing the new firewall rule as well as the output of the testing, can be found in `part_b/q5.txt`

   Rule added (can also be found in q5.txt under the iptables-save): Sudo iptables -A INPUT -p tcp --tcp-flags SYN SYN --dport 10095 -m state --state NEW -s 192.168.122.27/32 -j ACCEPT

6) A shell transcript including the new firewall rule as well as the command used to test the firewall rule, along with the corresponding outrrput can be found in `part_b/q6.txt`

Rule added (can also be found under the iptables-save output in q6.txt): Sudo iptables -A OUTPUT -p tcp --tcp-flags --dport 993 -j DROP