

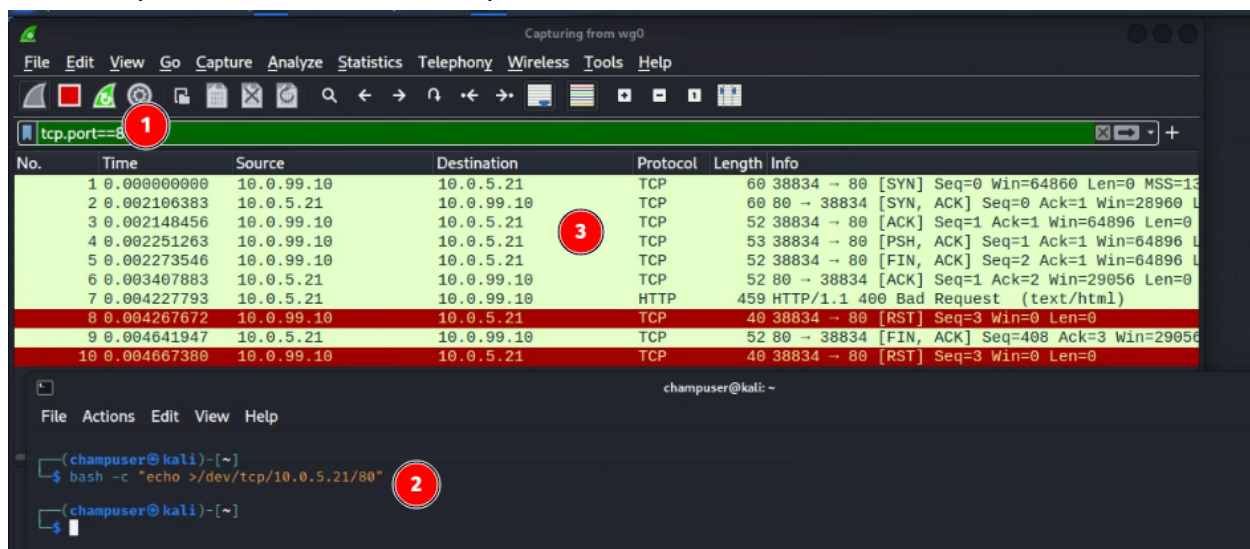
Lab 2.1 - Port Scanning 1

with bash - single host single port

💡 You will not always have tools on target, so a few other methods using native commands will also be introduced.

Try the following

- Open up wireshark on wg0 with a display filter on tcp port 80
- Execute the echo shown in 2
- Capture the TCP handshake, push and then tear down as shown in 3



Deliverable 1. Provide a screenshot similar to the one above, make sure to take a look at the interplay of TCP flags during setup and teardown of the tcp connection.

Multi host and multi port

Let's extend the example above by

- Creating a list of targets (you can use sweep.txt should you wish) shown in 1.
- Create a short list of popular tcp ports (exhaustive scans do take time)
- Recreate the script shown in 3. Collaborate with your teammates and add **enhancements** such as error checking, parameter and option checking and output enhancement (in its current state it provides csv output). Make sure to give credits in your code with a few comments.

```
/home/champuser/sec335/week2
```

```
(champuser@kali)-[~/sec335/week2]
```

```
$ cat mytargets.txt
```

```
10.0.5.20
```

```
10.0.5.22
```

1

```
(champuser@kali)-[~/sec335/week2]
```

```
$ cat mytcpports.txt
```

```
22
```

```
80
```

```
443
```

```
8080
```

```
139
```

```
445
```

```
3389
```

2

```
(champuser@kali)-[~/sec335/week2]
```

```
$ cat portscanner.sh
```

```
#!/bin/bash
```

```
hostfile=$1
```

```
portfile=$2
```

```
echo "host,port"
```

```
for host in $(cat $hostfile); do
```

```
    for port in $(cat $portfile); do
```

```
        timeout .1 bash -c "echo >/dev/tcp/$host/$port" 2>/dev/null &&
```

```
        echo "$host,$port"
```

```
    done
```

```
done
```

3

Deliverable 2. Execute your script (demo your enhancements as well), provide a source code listing (also upload this to your technical journal). Capture a screenshot of your program run similar to the one below. (Note, the ports may be different at the time of this lab)

```
(champuser@kali)-[~/sec335/week2]
```

```
$ ./portscanner.sh mytargets.txt mytcpports.txt
```

```
host,port
```

```
10.0.5.20,22
```

```
10.0.5.20,80
```

```
10.0.5.22,139
```

```
10.0.5.22,445
```

```
10.0.5.22,3389
```

Deliverable 3. So, you notice we target the file /dev/tcp/thehostip/thetcpport. Can you find this file in kali? Break out our friend google and see if you can find out what is going on. Briefly explain what you discover.

Nmap

💡 If you can access the target with nmap, use it. In some cases when tunneled knee deep in a network it is easier to simply use the tools on target than to proxy or tunnel traffic from kali to the target and back again. This is known as "Living-off-the-Land"

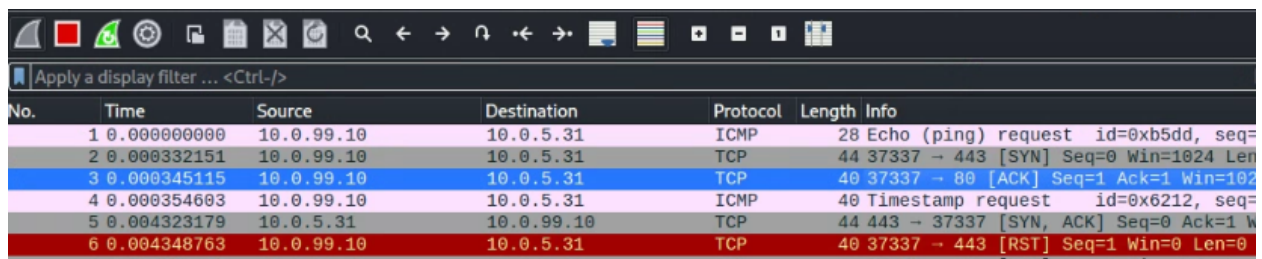
Nmap default scan

Begin a new wireshark session on wg0. Run a default scan against 10.0.5.31

```
sudo nmap 10.0.5.31
```

Deliverable 4. Provide a screenshot showing your nmap output

You will notice right away that the default nmap scan begins by an ICMP echo request, a SYN to 80 and 443 and an ICMP timestamp request. It then SYN scans 1000 popular ports. To see how it behaves against a given port, find one reported to be open (different from the example and observe the TCP FLAGS being invoked by client and server. The following image shows the server responding with a SYN/ACK (meaning the port is open) similar to the one below.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.99.10	10.0.5.31	ICMP	28	Echo (ping) request id=0xb5dd, seq=
2	0.000332151	10.0.99.10	10.0.5.31	TCP	44	37337 → 443 [SYN] Seq=0 Win=1024 Len
3	0.000345115	10.0.99.10	10.0.5.31	TCP	40	37337 → 80 [ACK] Seq=1 Ack=1 Win=102
4	0.000354603	10.0.99.10	10.0.5.31	ICMP	40	Timestamp request id=0x6212, seq=
5	0.004323179	10.0.5.31	10.0.99.10	TCP	44	443 → 37337 [SYN, ACK] Seq=0 Ack=1 W
6	0.004348763	10.0.99.10	10.0.5.31	TCP	40	37337 → 443 [RST] Seq=1 Win=0 Len=0

Deliverable 5, find another open port, create the appropriate display filter and submit a screenshot similar to the example (but with another port).

Nmap a single port

Restart a new wireshark capture and clear any display filters. Run the following command.

```
sudo nmap 10.0.5.31 -p 3389
```

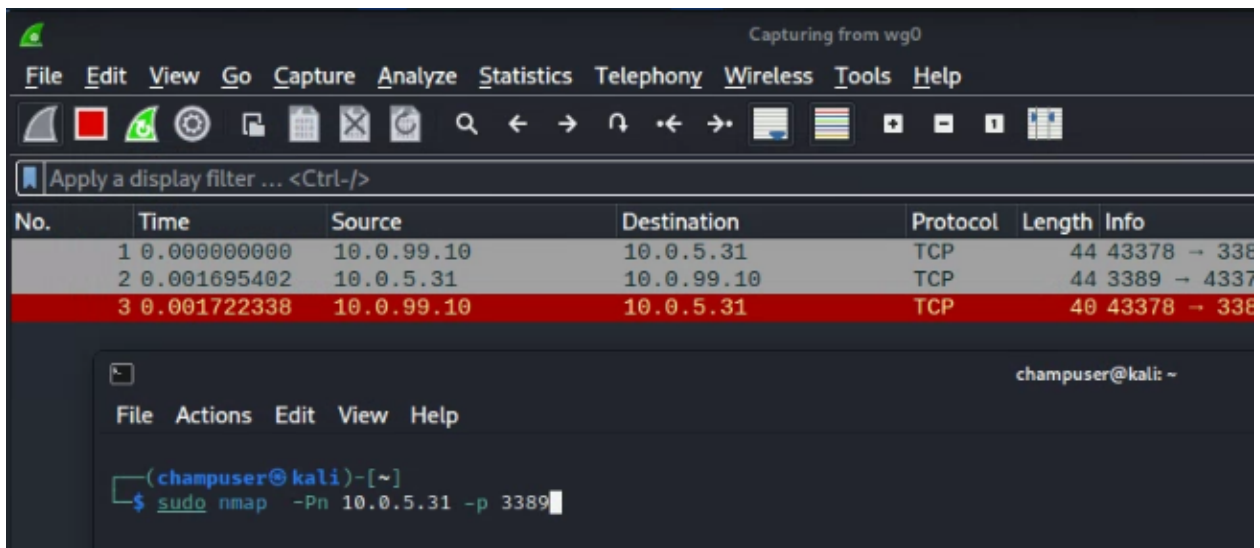
- Look for traffic to and from port 3389. Anything else is part of the default host discovery process.
- Run the same command without sudo

```
nmap 10.0.5.31 -p 3389
```

Deliverable 6. Describe the difference in the two wireshark captures

Limiting nmap's host discovery with -Pn

Deliverable 7. Add the -Pn flag and provide a wireshark display. With no display filter, you should have a total of 3 packets and evidence of a simple SYN scan similar to the one below.



The image shows a Wireshark network capture interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The packet list table below shows three packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.99.10	10.0.5.31	TCP	44	43378 → 3389
2	0.001695402	10.0.5.31	10.0.99.10	TCP	44	3389 → 43378
3	0.001722338	10.0.99.10	10.0.5.31	TCP	40	43378 → 3389

The third packet is highlighted in red. Below the packet list is a terminal window titled 'champuser@kali: ~' with a menu bar 'File Actions Edit View Help'. The terminal shows the command:

```
(champuser@kali)-[~]
$ sudo nmap -Pn 10.0.5.31 -p 3389
```

Deliverable 8. Provide links to any source code written in accomplishing this lab's objectives (remember, you can collaborate with your teammates on this). If you were asked to write a script (more than a line), make sure this is an actual file uploaded to the source part of github as opposed to a wiki entry (though you can certainly link to this file in your wiki).