Nmap

Arshad Ali

24/03/2024

Chapter 1

Nmap Basic Port Scan

1.1 Introduction

This chapter involves checking which ports are open and listening and which ports are closed. This chapter and subsequent chapter focus on port scanning and the different types of port scans used by nmap. This chapter explains the following:

1. TCP connect port scan
2. TCP SYN port scan
3. UDP port scan

Moreover, this chapter discusses the different options to specify the ports, the scan rate, and the number of parallel probes.

1.2 TCP and UDP Ports

J
J

ust like an IP address specifies a host on a network among many others, a TCP or UDP port is used to identify a network service running on that host. A server provides the network service, and it adheres to a specific network protocol. For example, providing time, responding to DNS queries, and serving web pages. A port is usually linked to a service using that specific port number. For instance, an HTTP server would bind to TCP port 80 by default; moreover, if the HTTP server supports SSL/TLS, it would listen on TCP port 443. (TCP ports 80 and 443 are the default ports for HTTP and HTTPS; however, the webserver administrator might choose other port numbers if necessary.) Furthermore, no more than one service can listen on any TCP or UDP port (on the same IP address). To keep things simple, we can classify ports in two states:

1. Open port indicates that there is some service listening on that port.
2. Closed port indicates that there is no service listening on that port.

However, in practical situations, we need to consider the impact of firewalls. For instance, a port might be open, but a firewall might be blocking the packets. Therefore, Nmap considers the following six states:

1. **Open**: indicates that a service is listening on the specified port.
2. **Closed:** indicates that no service is listening on the specified port, although the port is accessible. By accessible, we mean that it is reachable and is not blocked by a firewall or other security appliances/programs.
3. **Filtered:**means that Nmap cannot determine if the port is open or closed because the port is not accessible. This state is usually due to a firewall preventing Nmap from reaching that port. Nmap's packets may be blocked from reaching the port; alternatively, the responses are blocked from reaching Nmap's host.
4. **Unfiltered:**means that Nmap cannot determine if the port is open or closed, although the port is accessible. This state is encountered when using an ACK scan **-sA**.
5. **OpneFiltered:**This means that Nmap cannot determine whether the port is open or filtered.
6. **ClosedFiltered:**This means that Nmap cannot decide whether a port is closed or filtered.

1.3 TCP Flags

N
N

map supports different types of TCP port scans. To understand the difference between these port scans, we need to review the TCP header. The TCP header is the first 20 bytes of a TCP segment. First, we have the source TCP port number and the destination port number which are allocated 16 bits each (2 bytes). Then, we have the sequence number and the acknowledgement number. Each has 32 bits (4 bytes) allocated. In particular, we need to focus on the flags that Nmap can set or unset. Setting a flag bit means setting its value to 1. TCP header flags are:

1. **URG**: Urgent flag indicates that the urgent pointer filed is significant. The urgent pointer indicates that the incoming data is urgent, and that a TCP segment with the URG flag set is processed immediately without consideration of having to wait on previously sent TCP segments..

2. **ACK:** Acknowledgement flag indicates that the acknowledgement number is significant. It is used to acknowledge the receipt of a TCP segment.

3. **PSH:**Push flag asking TCP to pass the data to the application promptly.

4. **RST:**Reset flag is used to reset the connection. Another device, such as a firewall, might send it to tear a TCP connection. This flag is also used when data is sent to a host and there is no service on the receiving end to answer.

5. **SYN:**Synchronize flag is used to initiate a TCP 3-way handshake and synchronize sequence numbers with the other host. The sequence number should be set randomly during TCP connection establishment.

6. **FIN:**The sender has no more data to send.

1.4 TCP Connect Scan

T
T

CP connect scan works by completing the TCP 3-way handshake. In standard TCP connection establishment, the client sends a TCP packet with SYN flag set, and the server responds with SYN/ACK if the port is open; finally, the client completes the 3-way handshake by sending an ACK as shown in Figure 1.1.

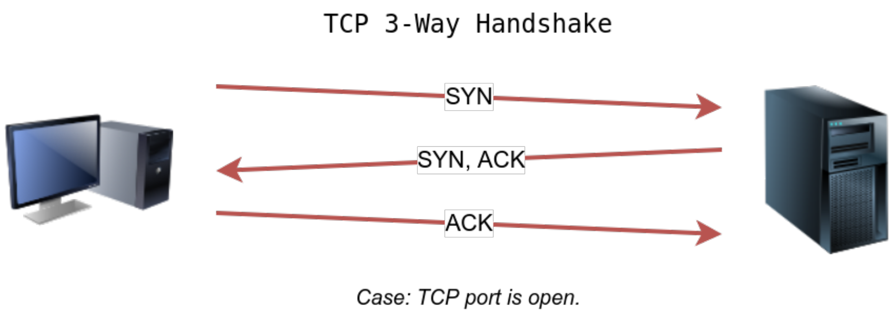


Figure 1.1: TCP 3-way handshake

We are interested in learning whether the TCP port is open, not establishing a TCP connection. Hence the connection is torn as soon as its state is confirmed by sending a RST/ACK. You can choose to run TCP connect scan using **-sT**. Case of TCP open port scan is shown in Figure 1.2.

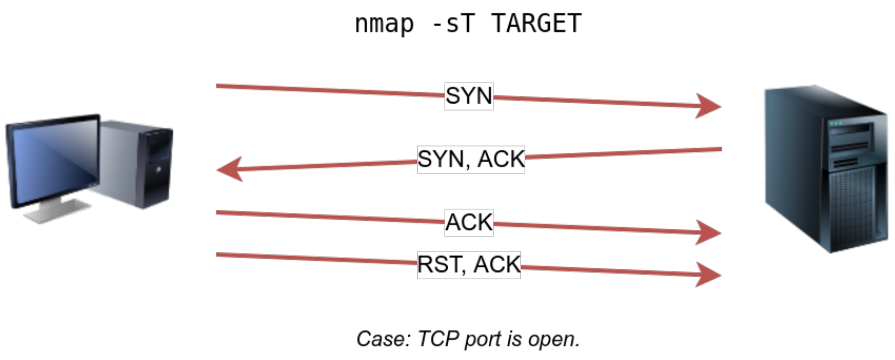
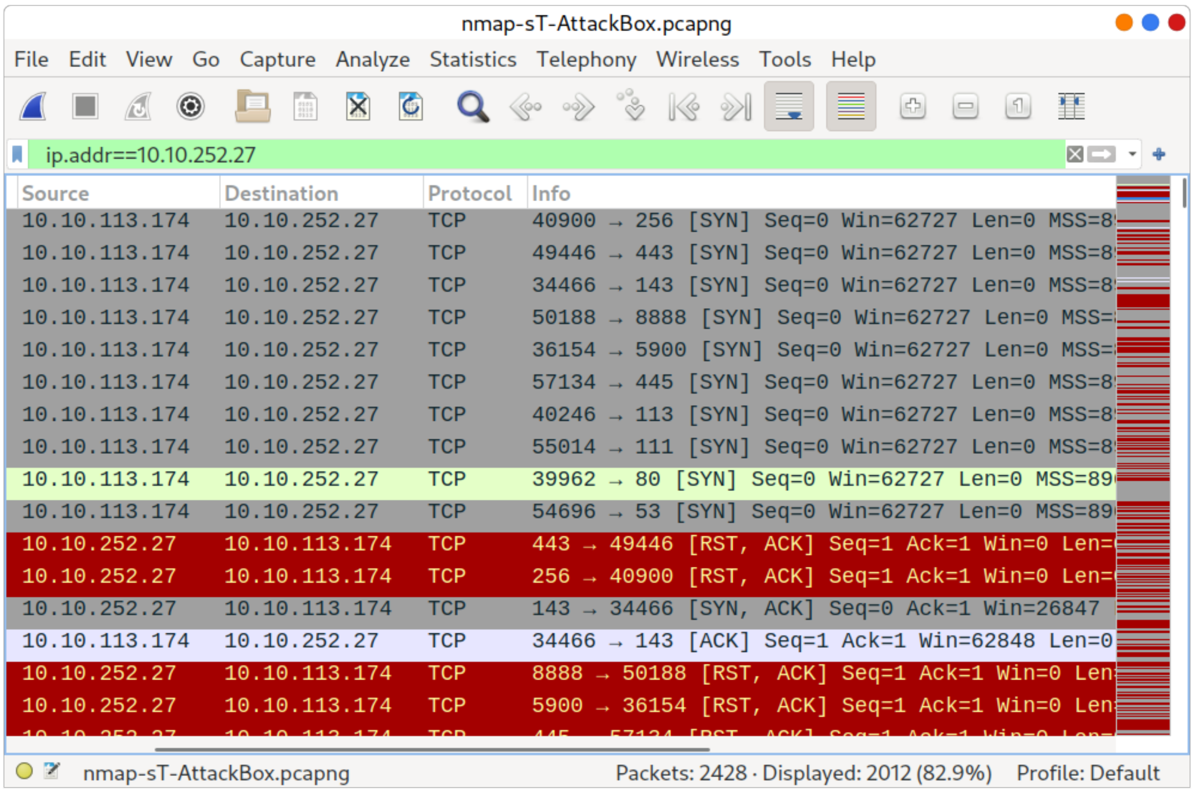


Figure 1.2: TCP open port scan

It is important to note that if you are not a privileged user (root or sudoer), a TCP connect scan is the only possible option to discover open TCP ports.
Figure Figure 1.3 shows a Wireshark packet capture window. we see Nmap sending TCP packets with SYN flag set to various ports, 256, 443, 143, and so on. By default, Nmap will attempt to connect to the 1000 most common ports. A closed TCP port responds to a SYN packet with RST/ACK to indicate that it is not open. This pattern will repeat for all the closed ports as we attempt to initiate a TCP 3-way handshake with them.
It is observed that port 143 is open, so it replied with a SYN/ACK, and Nmap completed the 3-way handshake by sending an ACK. The Figure 1.4 shows all the packets exchanged between our Nmap host and the target system's port 143. The first three packets are the TCP 3-way handshake being completed. Then, the fourth packet tears it down with an RST/ACK packet. To illustrate the **-sT** (TCP connect scan), the command example shown in Figure 1.5returned a detailed list of the open ports. Note that we can use **-F** to enable fast mode and decrease the number of scanned ports from

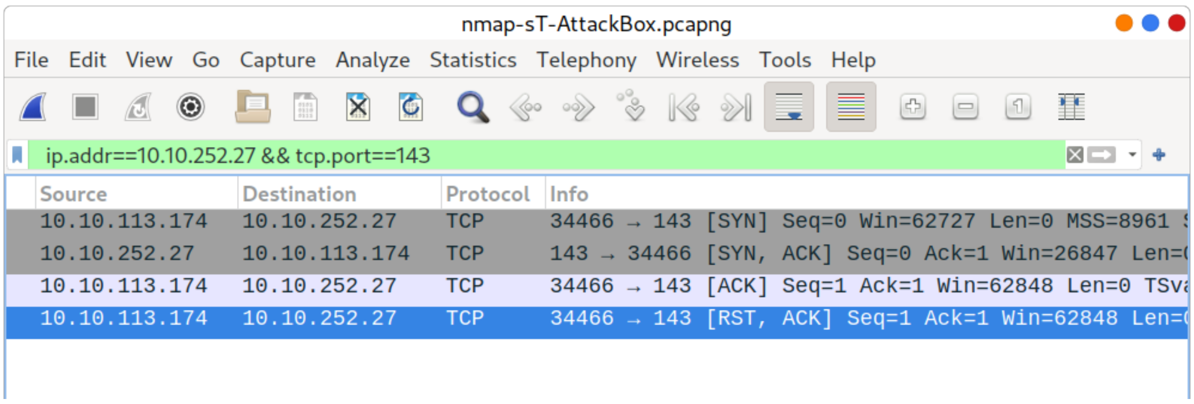

nmap-sT-AttackBox.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr==10.10.252.27

Source         Destination    Protocol  Info
10.10.113.174  10.10.252.27   TCP       40900 → 256 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       49446 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       34466 → 143 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       50188 → 8888 [SYN] Seq=0 Win=62727 Len=0 MSS=
10.10.113.174  10.10.252.27   TCP       36154 → 5900 [SYN] Seq=0 Win=62727 Len=0 MSS=
10.10.113.174  10.10.252.27   TCP       57134 → 445 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       40246 → 113 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       55014 → 111 [SYN] Seq=0 Win=62727 Len=0 MSS=8
10.10.113.174  10.10.252.27   TCP       39962 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=89
10.10.113.174  10.10.252.27   TCP       54696 → 53 [SYN] Seq=0 Win=62727 Len=0 MSS=89
10.10.252.27   10.10.113.174  TCP       443 → 49446 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27   10.10.113.174  TCP       256 → 40900 [RST, ACK] Seq=1 Ack=1 Win=0 Len=
10.10.252.27   10.10.113.174  TCP       143 → 34466 [SYN, ACK] Seq=0 Ack=1 Win=26847
10.10.113.174  10.10.252.27   TCP       34466 → 143 [ACK] Seq=1 Ack=1 Win=62848 Len=0
10.10.252.27   10.10.113.174  TCP       8888 → 50188 [RST, ACK] Seq=1 Ack=1 Win=0 Len
10.10.252.27   10.10.113.174  TCP       5900 → 36154 [RST, ACK] Seq=1 Ack=1 Win=0 Len
10.10.252.27   10.10.113.174  TCP       445 → 57134 [RST, ACK] Seq=1 Ack=1 Win=0 Len=

nmap-sT-AttackBox.pcapng          Packets: 2428 · Displayed: 2012 (82.9%)    Profile: Default
```

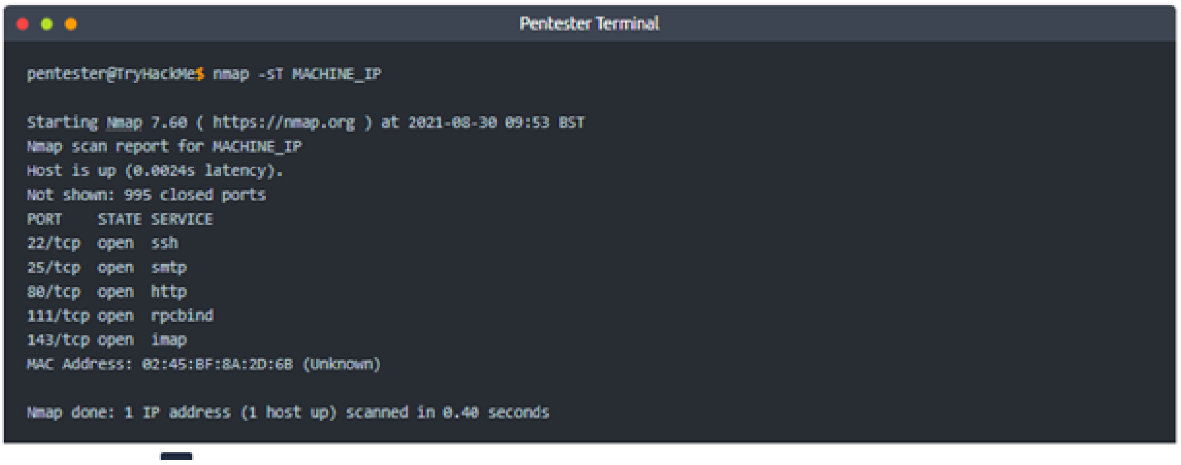
Figure 1.3: TCP 3-way handshake


nmap-sT-AttackBox.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr==10.10.252.27 && tcp.port==143

Source         Destination    Protocol  Info
10.10.113.174  10.10.252.27   TCP       34466 → 143 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 S
10.10.252.27   10.10.113.174  TCP       143 → 34466 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0
10.10.113.174  10.10.252.27   TCP       34466 → 143 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSva
10.10.113.174  10.10.252.27   TCP       34466 → 143 [RST, ACK] Seq=1 Ack=1 Win=62848 Len=0
```

Figure 1.4: Wireshark View



```
pentester@TryHackMe$ nmap -sT MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for MACHINE_IP
Host is up (0.0024s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
111/tcp open  rpcbind
143/tcp open  imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.40 seconds
```

Figure 1.5: Wireshark view: Detailed list of open ports

1000 to 100 most common ports. It is worth mentioning that the **-r** option can also be added to scan the ports in consecutive order instead of random order. This option is useful when testing whether ports open in a consistent manner, for instance, when a target boots up.

1.5 TCP SYN Scan

U
U

nprivileged users are limited to connect scan. However, the default scan mode is SYN scan, and it requires a privileged (root or sudoer) user to run it. SYN scan does not need to complete the TCP 3-way handshake; instead, it tears down the connection once it receives a response from the server. Because we didn't establish a TCP connection, this decreases the chances of the scan being logged. We can select this scan type by using the -sS option. The Figure **??** shows how the TCP SYN scan works without completing the TCP 3-way handshake. The screenshot shown in Figure 1.7 shows a

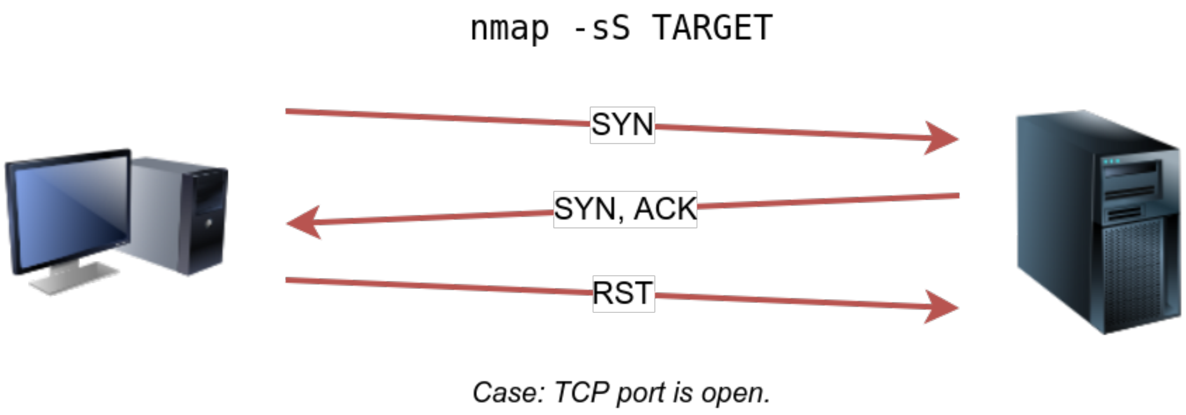
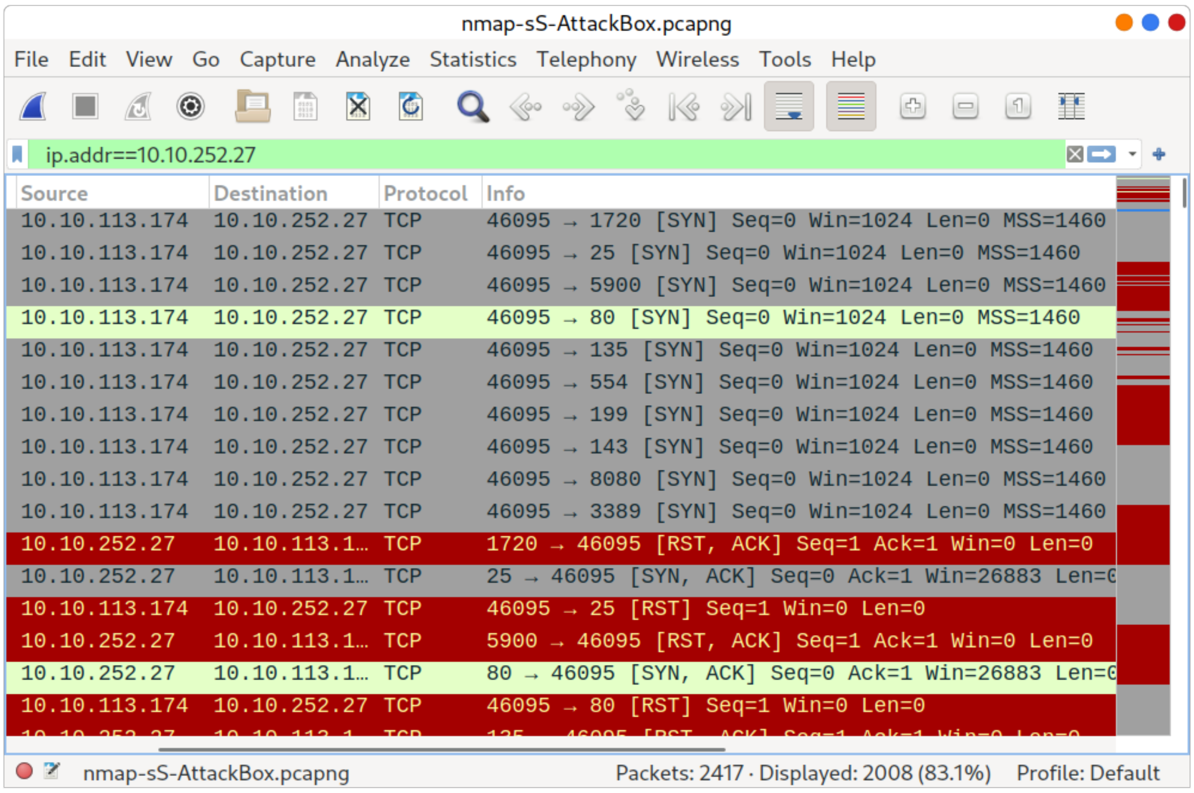


Figure 1.6: TCP SYN scan

TCP SYN scanfrom Wireshark. The behaviour in the case of closed TCP ports is similar to that of the TCP connect scan.
To better see the difference between the two scans, consider the screenshot shown in Figure 1.8. In the upper half of the following figure, we can see a TCP connect scan **-sT** traffic. Any open TCP port will require Nmap to complete the TCP 3-way handshake before closing the connection. In the lower half of the following figure, we see how a SYN scan **-sS** does not need to complete the TCP 3-way handshake; instead, Nmap sends an RST packet once a SYN/ACK packet is received. TCP SYN scan is the default scan mode when running Nmap as a privileged user, running as root or using sudo, and it is a very reliable choice. It has successfully discovered the open ports you found earlier with the TCP connect scan, yet no TCP connection was fully established with the target.Figure 1.9 shows terminal view of the command.

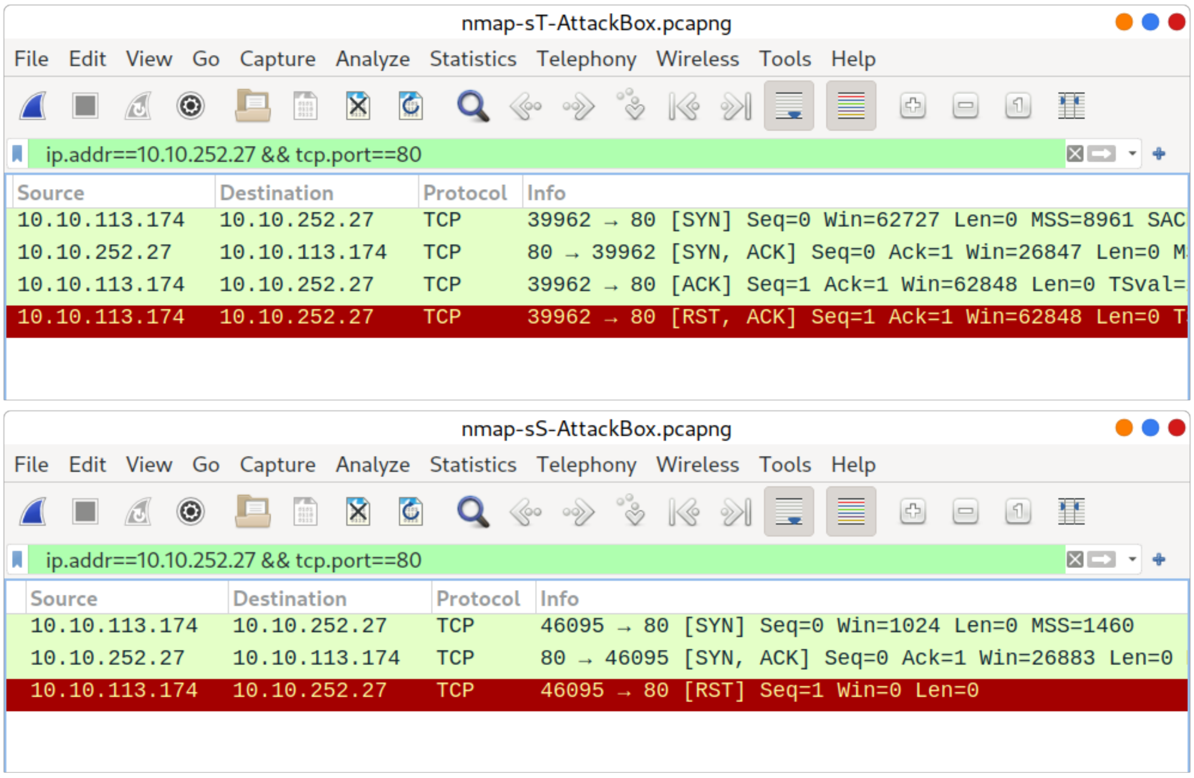


```
ip.addr==10.10.252.27

Source         Destination    Protocol  Info
10.10.113.174  10.10.252.27   TCP       46095 → 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 554 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 199 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174  10.10.252.27   TCP       46095 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.252.27   10.10.113.1…   TCP       1720 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27   10.10.113.1…   TCP       25 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174  10.10.252.27   TCP       46095 → 25 [RST] Seq=1 Win=0 Len=0
10.10.252.27   10.10.113.1…   TCP       5900 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27   10.10.113.1…   TCP       80 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174  10.10.252.27   TCP       46095 → 80 [RST] Seq=1 Win=0 Len=0
10.10.252.27   10.10.113.1…   TCP       135 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

nmap-sS-AttackBox.pcapng     Packets: 2417 · Displayed: 2008 (83.1%)     Profile: Default
```

Figure 1.7: TCP SYN scan Wireshark view



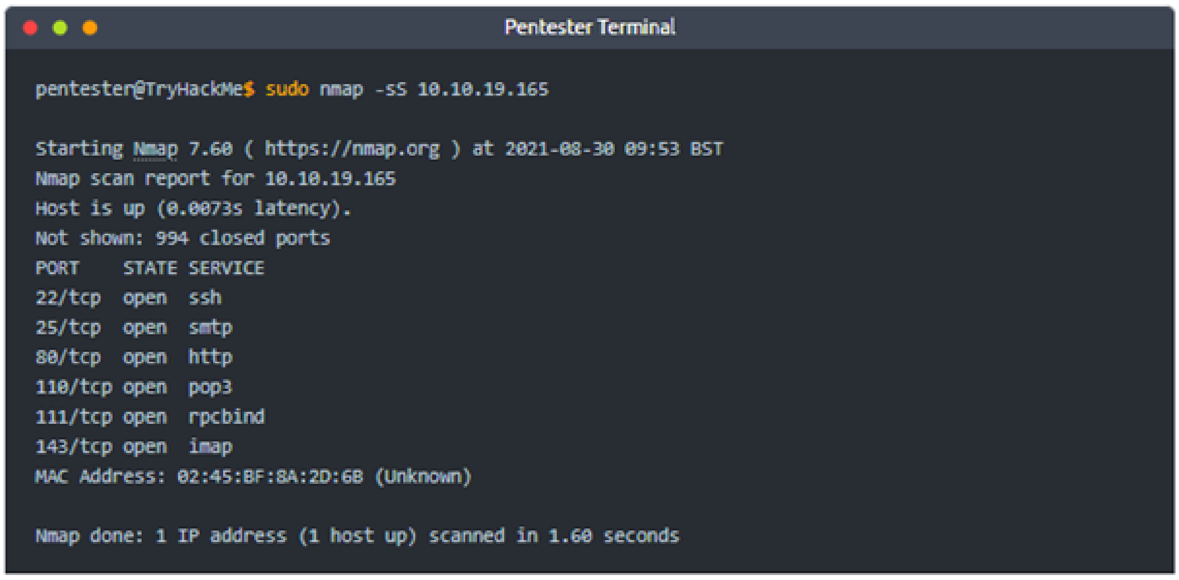
```
ip.addr==10.10.252.27 && tcp.port==80

Source         Destination    Protocol  Info
10.10.113.174  10.10.252.27   TCP       39962 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SAC
10.10.252.27   10.10.113.174  TCP       80 → 39962 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 M
10.10.113.174  10.10.252.27   TCP       39962 → 80 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=
10.10.113.174  10.10.252.27   TCP       39962 → 80 [RST, ACK] Seq=1 Ack=1 Win=62848 Len=0 T
```

```
ip.addr==10.10.252.27 && tcp.port==80

Source         Destination    Protocol  Info
10.10.113.174  10.10.252.27   TCP       46095 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.252.27   10.10.113.174  TCP       80 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174  10.10.252.27   TCP       46095 → 80 [RST] Seq=1 Win=0 Len=0
```

Figure 1.8: TCP open port scan



```
pentester@TryHackMe$ sudo nmap -sS 10.10.19.165

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for 10.10.19.165
Host is up (0.0073s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.60 seconds
```

Figure 1.9: Pentester terminal view of TCP SYN scan

1.6 UDP Scan

U
U

DP is a connectionless protocol, and hence it does not require any handshake for connection establishment. We cannot guarantee that a service listening on a UDP port would respond to our packets. However, if a UDP packet is sent to a closed port, an ICMP port unreachable error (type 3, code 3) is returned. You can select UDP scan using the -sU option; moreover, you can combine it with another TCP scan. The Figure 1.10shows that if we send a UDP packet to an open UDP port, we cannot expect any reply in return. Therefore, sending a UDP packet to an open port won't tell us anything.

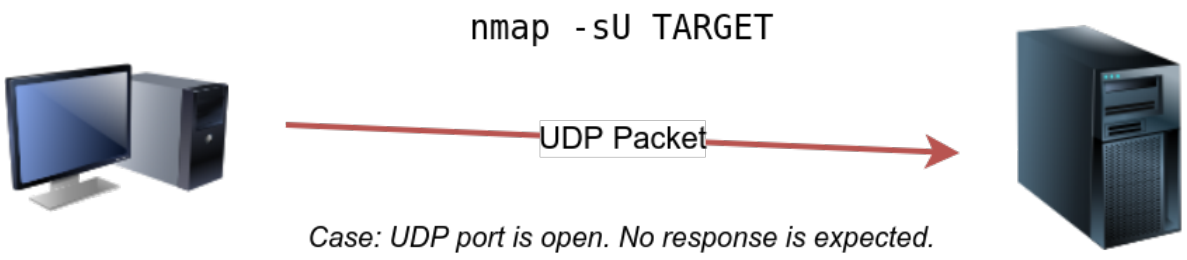


Figure 1.10: UDP open port

However, as shown in the Figure 1.11, we expect to get an ICMP packet of type 3, destination unreachable, and code 3, port unreachable. In other words, the UDP ports that don't generate any response are the ones that Nmap will state as open.

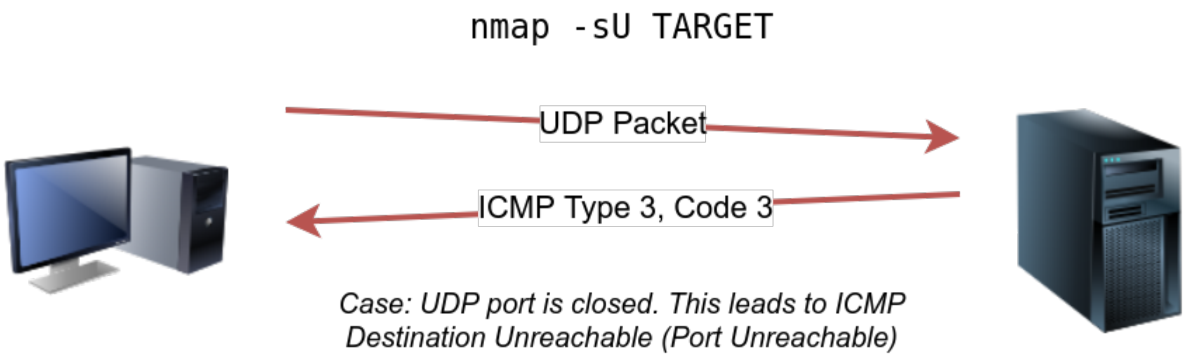
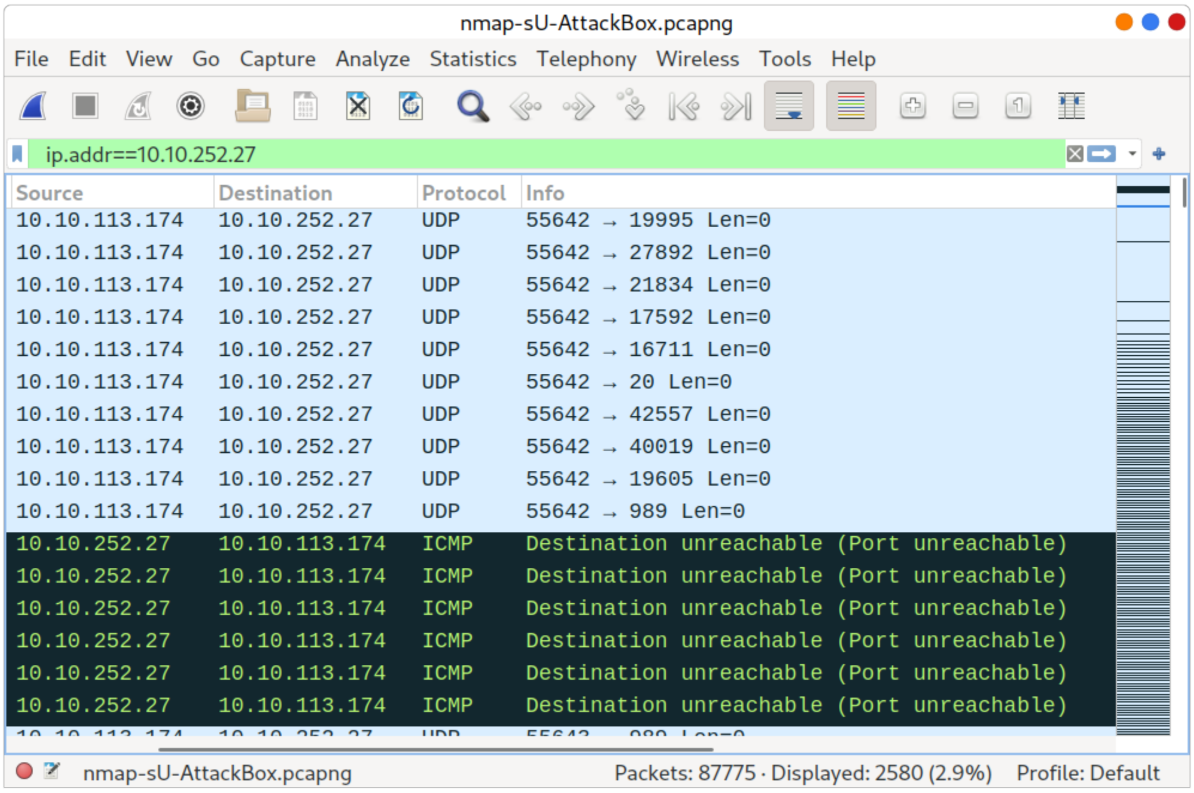

nmap -sU TARGET
UDP Packet
ICMP Type 3, Code 3
Case: UDP port is closed. This leads to ICMP Destination Unreachable (Port Unreachable)

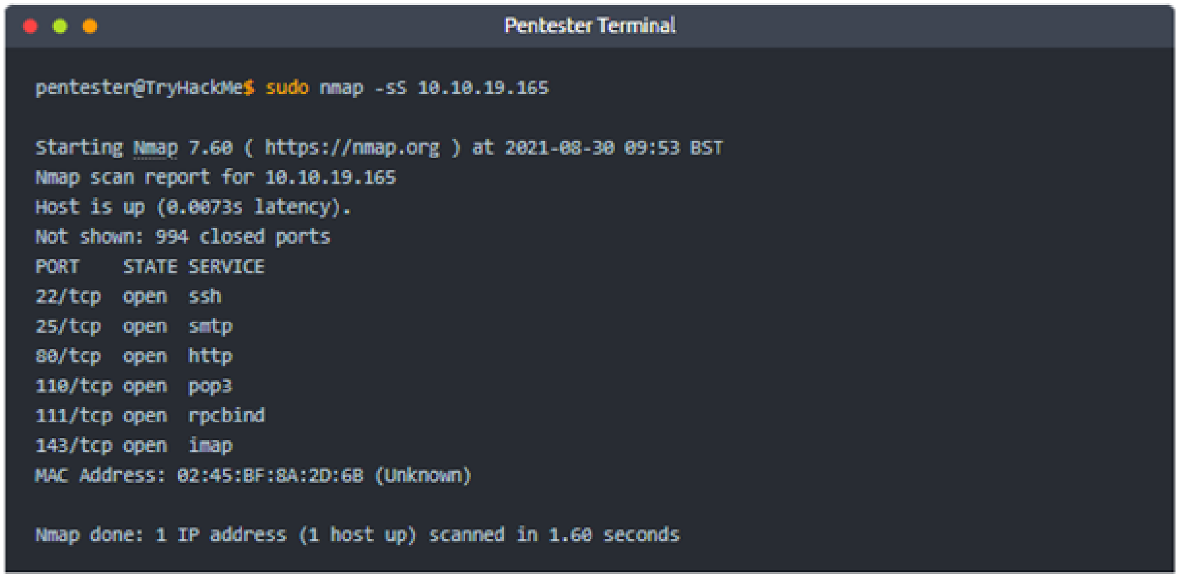
Figure 1.11: UDP close port

In the Wireshark capture shown in Figure 1.12, we can see that every closed port will generate an ICMP packet destination unreachable (port unreachable).
Launching a UDP scan against this Linux server proved valuable, and indeed, we learned that port 111 is open. On the other hand, Nmap cannot determine whether UDP port 68 is open or filtered. Terinal view is shown in Figure 1.13


nmap-sU-AttackBox.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.addr==10.10.252.27
nmap-sU-AttackBox.pcapng
Packets: 87775 · Displayed: 2580 (2.9%)
Profile: Default

```
Source          Destination     Protocol  Info
10.10.113.174   10.10.252.27    UDP       55642 → 19995 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 27892 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 21834 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 17592 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 16711 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 20 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 42557 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 40019 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 19605 Len=0
10.10.113.174   10.10.252.27    UDP       55642 → 989 Len=0
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
10.10.252.27    10.10.113.174   ICMP      Destination unreachable (Port unreachable)
```

Figure 1.12: UDP port scan wireshark view



```
pentester@TryHackMe$ sudo nmap -sS 10.10.19.165

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for 10.10.19.165
Host is up (0.0073s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.60 seconds
```

Figure 1.13: UDP port scan terminal view

Option	Purpose
-p-	all ports
-p1-1023	scan ports 1 to 1023
-F	100 most common ports
-r	scan ports in consecutive order
-T¡0-5¿	-T0 being the slowest and T5 the fastest
–max-rate 50	rate ¡= 50 packets/sec
–min-rate 15	rate ¿= 15 packets/sec
–min-parallelism 100	at least 100 probes in parallel

Table 1.1: Commands and purposes

1.7 Fine Tuning Scope and Performance

Y

Y

ou can specify the ports you want to scan instead of the default 1000 ports. Specifying the ports is intuitive by now. Let's see some examples:

1. port list: -p22,80,443 will scan ports 22, 80 and 443..
2. port range: -p1-1023 will scan all ports between 1 and 1023 inclusive, while -p20-25 will scan ports between 20 and 25 inclusive.

You can request the scan of all ports by using -p-, which will scan all 65535 ports. If you want to scan the most common 100 ports, add -F. Using will check the ten most common ports. You can control the scan timing using -T¡0-5¿. -T0 is the slowest (paranoid), while -T5 is the fastest. According to Nmap manual page, there are six templates:

1. paranoid (0)
2. sneaky (1)
3. polite (2)
4. normal (3)
5. aggressive (4)
6. insane (5)

To avoid IDS alerts, you might consider -T0 or -T1. For instance, -T0 scans one port at a time and waits 5 minutes between sending each probe, so you can guess how long scanning one target would take to finish. If you don't specify any timing, Nmap uses normal -T3. Note that -T5 is the most aggressive in terms of speed; however, this can affect the accuracy of the scan results due to the increased likelihood of packet loss. Note that -T4 is often used during CTFs and when learning to scan on practice targets, whereas -T1 is often used during real engagements where stealth is more important. Table 1.1 shows various commands and their purposes.
Alternatively, you can choose to control the packet rate using –min-rate ¡number¿ and –max-rate ¡number¿. For example, –max-rate 10 or –max-rate=10 ensures that your scanner is not sending more than ten packets per second. Moreover, you can control probing parallelization using –min-parallelism ¡numprobes¿ and –max-parallelism ¡numprobes¿. Nmap probes the targets to discover which hosts are live and which ports are open; probing parallelization specifies the number of such probes that can be run in parallel. For instance, –min-parallelism=512 pushes Nmap to maintain at least 512 probes in parallel; these 512 probes are related to host discovery and open ports.

Port Scan Type	Example Command
TCP Connect Scan	nmap -sT 10.10.19.165
TCP SYN Scan	sudo nmap -sS 10.10.19.165
UDP Scan	sudo nmap -sU 10.10.19.165

Table 1.2: Port scan types and example commands

1.8 Summary

This Chapter covered the three types of scans as shown in Table 1.2:

Chapter 2

Nmap Advanced Port Scans

2.1 Introduction

This chapter explains advanced types of scans and scan options. Some of these scan types can be useful against specific systems, while others are useful in particular network setups. The following types of port scans are covered:

1. Null Scan
2. FIN Scan
3. Xmas Scan
4. ACK Scan
5. Window Scan
6. Custom Scan

Moreover, it covers the following:

1. Spoofing IP
2. Spoofing MAC
3. Decoy Scan
4. Fragmented Packets
5. Idle/Zombie Scan

Options and techniques to evade firewalls and IDS systems are also discussed.

2.2 TCP Null Scan, Full Scan and Xmas Scan

This section covers three types of scanes namely Null, Fin and Xmas.

2.2.1 Null Scan

The null scan does not set any flag; all six flag bits are set to zero. You can choose this scan using the **-sN** option. A TCP packet with no flags set will not trigger any response when it reaches an open port. Therefore, from Nmap's perspective, a lack of reply in a null scan indicates that either the port is open or a firewall is blocking the packet.
However, we expect the target server to respond with an RST packet if the port is closed. Consequently, we can use the lack of RST response to figure out the ports that are not closed: open or filtered.
Because the null scan relies on the lack of a response to infer that the port is not closed, it cannot indicate with certainty that these ports are open; there is a possibility that the ports are not responding due to a firewall rule.

2.2.2 Fin Scan

The FIN scan sends a TCP packet with the FIN flag set. You can choose this scan type using the **-sF** option. Similarly, no response will be sent if the TCP port is open. Again, Nmap cannot be sure if the port is open or if a firewall is blocking the traffic related to this TCP port.
However, the target system should respond with an RST if the port is closed. Consequently, we will be able to know which ports are closed and use this knowledge to infer the ports that are open or filtered. It's worth noting some firewalls will 'silently' drop the traffic without sending an RST.

2.2.3 Xmas Scan

The Xmas scan gets its name after Christmas tree lights. An Xmas scan sets the FIN, PSH, and URG flags simultaneously. You can select Xmas scan with the option **-sX**.
Like the Null scan and FIN scan, if an RST packet is received, it means that the port is closed. Otherwise, it will be reported as open—filtered.
One scenario where these three scan types can be efficient is when scanning a target behind a stateless (non-stateful) firewall. A stateless firewall will check if the incoming packet has the SYN flag set to detect a connection attempt. Using a flag combination that does not match the SYN packet makes it possible to deceive the firewall and reach the system behind it. However, a stateful firewall will practically block all such crafted packets and render this kind of scan useless.

2.3 TCP ACK, Window, Custom Scan

2.3.1 TCP ACK Scan

An ACK scan will send a TCP packet with the ACK flag set. Use the **-sA** option to choose this scan. The target would respond to the ACK with RST regardless of the state of the port. This behaviour happens because a TCP packet with the ACK flag set should be sent only in response to a received TCP packet to acknowledge the receipt of some data. Hence, this scan won't tell us whether the target port is open in a simple setup.
This kind of scan would be helpful if there is a firewall in front of the target. Consequently, based on which ACK packets resulted in responses, you will learn which ports were not blocked by the firewall. In other words, this type of scan is more suitable to discover firewall rule sets and configuration.
After setting up the target VM with a firewall, you may get some interesting results. There may be some open ports that aren't being blocked by the firewall. This indicates that the firewall is blocking all other ports except a few ones.

2.3.2 TCP Window Scan

The TCP window scan is almost the same as the ACK scan; however, it examines the TCP Window field of the RST packets returned. On specific systems, this can reveal that the port is open. You can select this scan type with the option **-sW**. We expect to get an RST packet in reply to our "uninvited" ACK packets, regardless of whether the port is open or closed.
Without firewall, this scan gives results similar to TCP ACK Scan. However, if we perform TCP window scan against a server behind a firewall, we expect to get more satisfying results. The TCP window scan points towards ports detected as closed (in contrast with the ACK scan that labelled the same ports as unfiltered).

2.3.3 TCP Custom Scan

If you want to experiment with a new TCP flag combination beyond the built-in TCP scan types, you can do so using **–scanflags**. For instance, if you want to set SYN, RST, and FIN simultaneously, you can do so using **–scanflags RSTSYNFIN**. If you develop your custom scan, you need to know how the different ports will behave to interpret the results in different scenarios correctly.
Finally, it is essential to note that the ACK scan and the window scan were very efficient at helping us map out the firewall rules. However, it is vital to remember that just because a firewall is not blocking a specific port, it does not necessarily mean that a service is listening on that port. For example, there is a possibility that the firewall rules need to be updated to reflect recent service changes. Hence, ACK and window scans are exposing the firewall rules, not the services.

2.4 Spoofing and Decoys

In some network setups, you will be able to scan a target system using a spoofed IP address and even a spoofed MAC address. Such a scan is only beneficial in a situation where you can guarantee to capture the response. If you try to scan a target from some random network using a spoofed IP address, chances are you won't have any response routed to you, and the scan results could be unreliable.
The Figure shows 2.1 the attacker launching the command **nmap -S SPOOFED_IP MACHINE_IP**. Consequently, Nmap will craft all the packets using the provided source IP address **SPOOFED_IP**. The target machine will respond to the incoming packets sending the replies to the destination IP address **SPOOFED_IP**. For this scan to work and give accurate results, the attacker needs to monitor the network traffic to analyze the replies. In brief, scanning with a spoofed IP address is three steps:

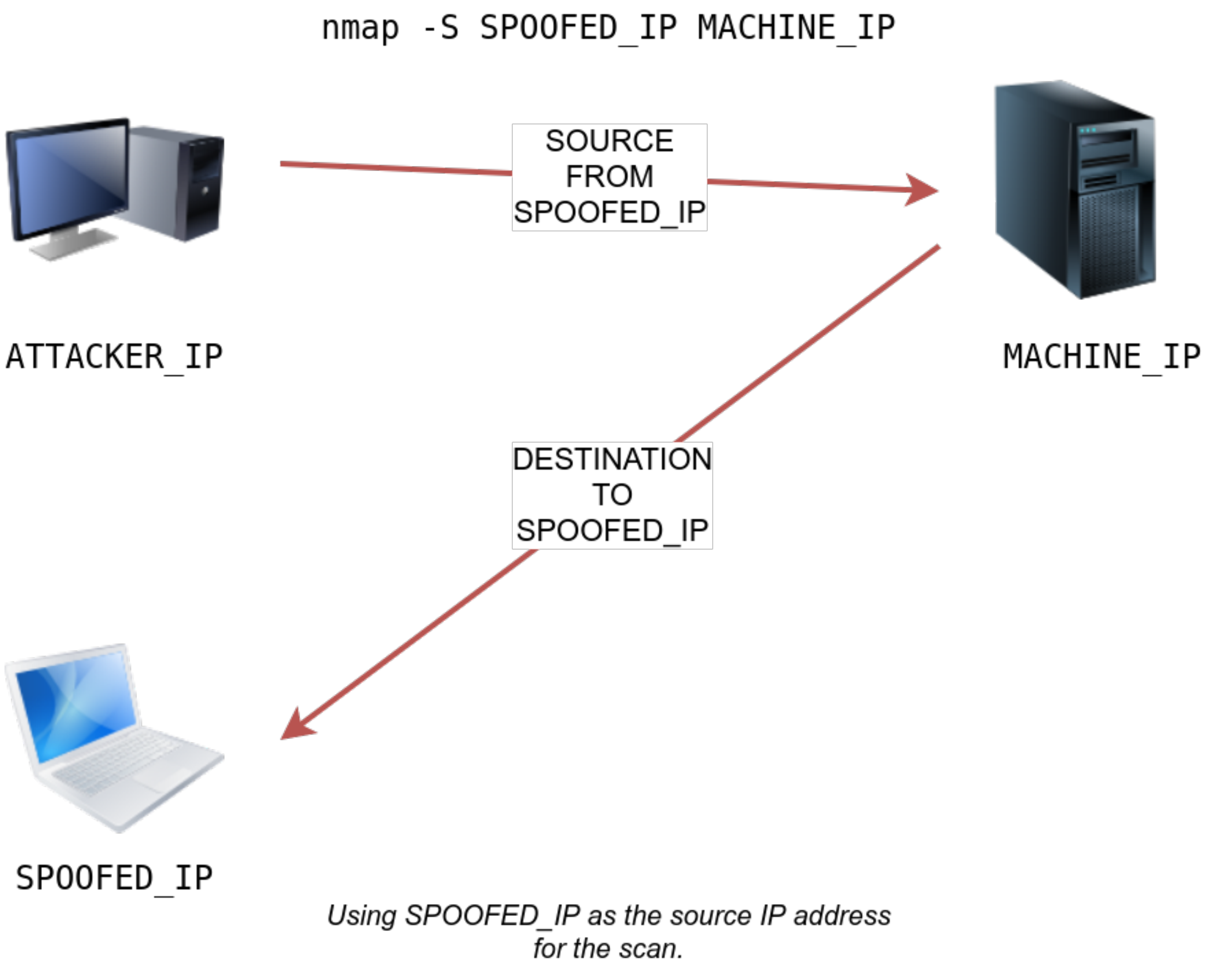

nmap -S SPOOFED_IP MACHINE_IP
SOURCE FROM SPOOFED_IP
ATTACKER_IP
MACHINE_IP
DESTINATION TO SPOOFED_IP
SPOOFED_IP
Using SPOOFED_IP as the source IP address for the scan.

Figure 2.1: Using SPOOFED IP as the source IP address

1. Attacker sends a packet with a spoofed source IP address to the target machine.
2. Target machine replies to the spoofed IP address as the destination.
3. Attacker captures the replies to figure out open ports.

In general, you expect to specify the network interface using **-e** and to explicitly disable ping scan **-Pn**. Therefore, instead of **nmap -S SPOOFED_IP MACHINE_IP** , you will need to issue **nmap -e NET_INTERFACE -Pn -S SPOOFED_IP MACHINE_IP** to tell Nmap explicitly which network interface to use and not to expect to receive a ping reply. It is worth repeating that this scan will be useless if the attacker system cannot monitor the network for responses.
When you are on the same subnet as the target machine, you would be able to spoof your MAC address as well. You can specify the source MAC address using **–spoof-mac SPOOFED_MAC**. This address spoofing is only possible if the attacker and the target machine are on the same Ethernet (802.3) network or same WiFi (802.11).
Spoofing only works in a minimal number of cases where certain conditions are met. Therefore, the attacker might resort to using decoys to make it more challenging to be pinpointed. The concept is simple, make the scan appear to be coming from many IP addresses so that the attacker's IP address would be lost among them. As shown in Figure 2.2, the scan of the target machine will appear to be coming from 3 different sources, and consequently, the replies will go the decoys as well. You can launch a decoy scan by specifying a specific or random IP address after **-D**. For ex-

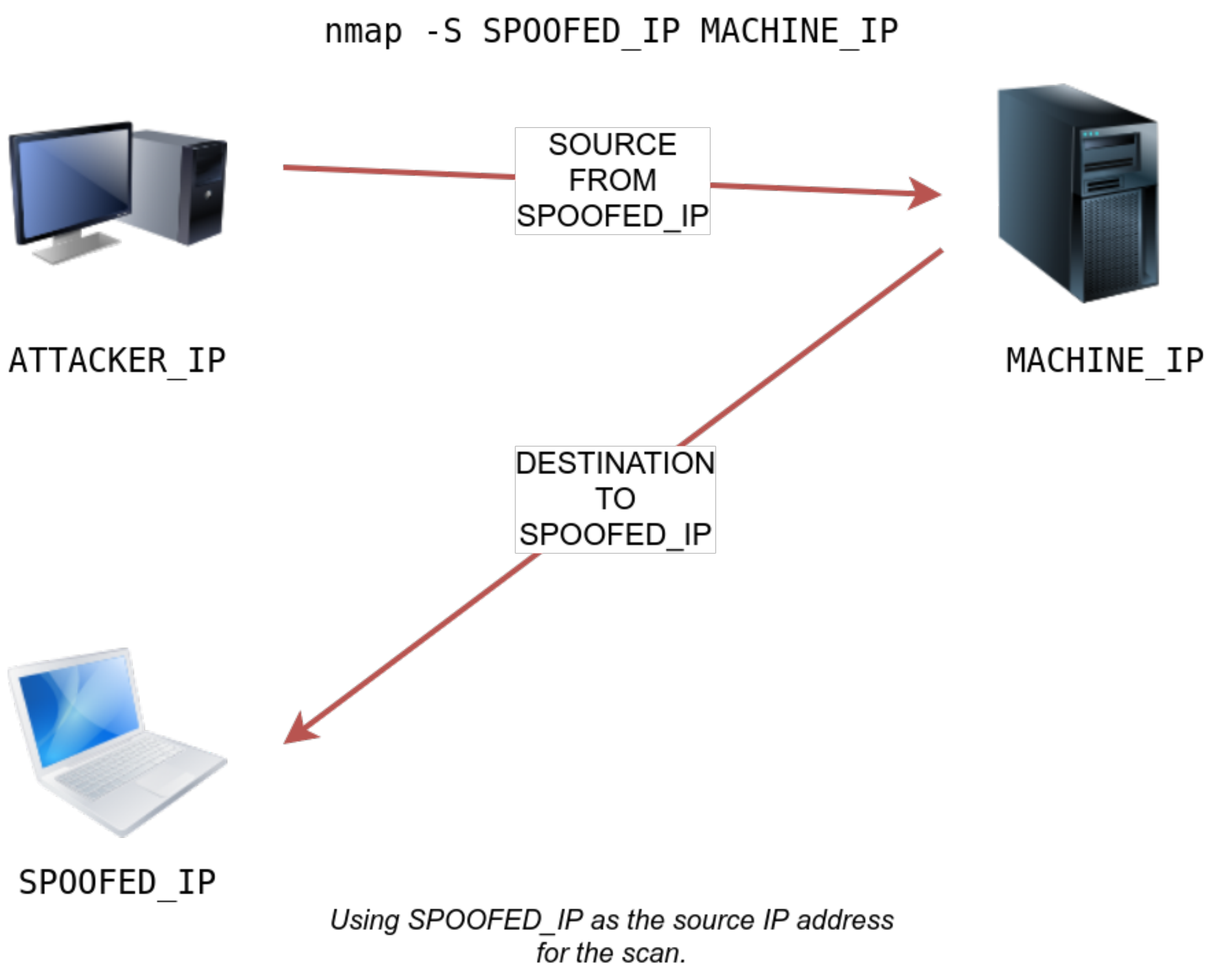

nmap -S SPOOFED_IP MACHINE_IP
SOURCE FROM SPOOFED_IP
ATTACKER_IP
MACHINE_IP
DESTINATION TO SPOOFED_IP
SPOOFED_IP
Using SPOOFED_IP as the source IP address for the scan.

Figure 2.2: Using DECOY1, ATTACKER_IP, DECOY2 as the source IP for the scan

ample, **nmap -D 10.10.0.1,10.10.0.2,ME MACHINE_IP** will make the scan of MACHINE_IP appear as coming from the IP addresses 10.10.0.1, 10.10.0.2, and then **ME** to indicate that your IP address should appear in the third order. Another example command would be **nmap -D 10.10.0.1,10.10.0.2,RND,RND,ME MACHINE_IP** where the third and fourth source IP addresses are assigned randomly, while the fifth source is going to be the attacker's IP address. In other words, each time you execute the latter command, you would expect two new random IP addresses to be the third and fourth decoy sources.

2.5 Fragmented Packets

2.5.1 Firewall

A firewall is a piece of software or hardware that permits packets to pass through or blocks them. It functions based on firewall rules, summarized as blocking all traffic with exceptions or allowing all traffic with exceptions. For instance, you might block all traffic to your server except those coming to your web server. A traditional firewall inspects, at least, the IP header and the transport layer header. A more sophisticated firewall would also try to examine the data carried by the transport layer.

2.5.2 IDS

An intrusion detection system (IDS) inspects network packets for select behavioural patterns or specific content signatures. It raises an alert whenever a malicious rule is met. In addition to the IP header and transport layer header, an IDS would inspect the data contents in the transport layer and check if it matches any malicious patterns. How can you make it less likely for a traditional firewall/IDS to detect your Nmap activity? It is not easy to answer this; however, depending on the type of firewall/IDS, you might benefit from dividing the packet into smaller packets.

2.5.3 Fragmented Packets

Nmap provides the option **-f** to fragment packets. Once chosen, the IP data will be divided into 8 bytes or less. Adding another **-f (-f -f or -ff)** will split the data into 16 byte-fragments instead of 8. You can change the default value by using the **–mtu**–mtu; however, you should always choose a multiple of 8.
To properly understand fragmentation, we need to look at the IP header in the Figure 2.3. It might look complicated at first, but we notice that we know most of its fields. In particular, notice the source address taking 32 bits (4 bytes) on the fourth row, while the destination address is taking another 4 bytes on the fifth row. The data that we will fragment across multiple packets is highlighted in red. To aid in the reassembly on the recipient side, IP uses the identification (ID) and fragment offset, shown on the second row of the 2.3.
Let's compare running **sudo nmap -sS -p80 10.20.30.144** and **udo nmap -sS -p80 -f 10.20.30.144**. First one uses stealth TCP SYN scan on port 80; however, in the second command, we are requesting Nmap to fragment the IP packets.
In the first two lines of Figure 2.4, we can see an ARP query and response. Nmap issued an ARP query because the target is on the same Ethernet. The second two lines show a TCP SYN ping and a reply. The fifth line is the beginning of the port scan; Nmap sends a TCP SYN packet to port 80. In this case, the IP header is 20 bytes, and the TCP header is 24 bytes. Note that the minimum size of the TCP header is 20 bytes. 2.3.
With fragmentation requested via -f, the 24 bytes of the TCP header will be divided into multiples of 8 bytes, with the last fragment containing 8 bytes or less of the TCP header. Since 24 is divisible by 8, we got 3 IP fragments; each has 20 bytes of IP header and 8 bytes of TCP header. We can see the three fragments in between as shown in Figure 2.5. 2.3. Note that if you added **-ff** (or **-f -f**), the fragmentation of the data will be multiples of 16. In other words, the 24 bytes of the TCP header, in this case, would be divided over two IP fragments, the first containing 16 bytes and the second containing 8 bytes of the TCP header.
On the other hand, if you prefer to increase the size of your packets to make them look innocuous, you can use the option **–data-length NUM**, where num specifies the number of bytes you want to

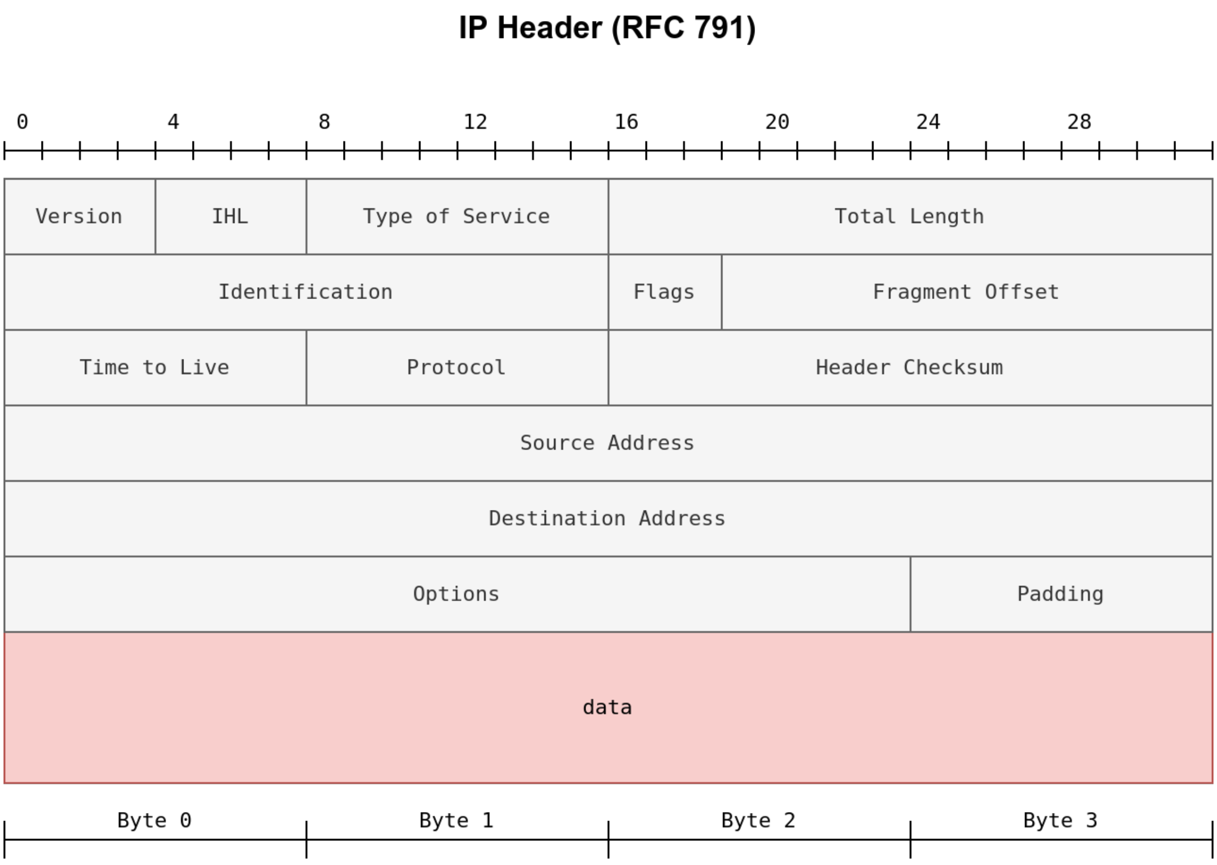
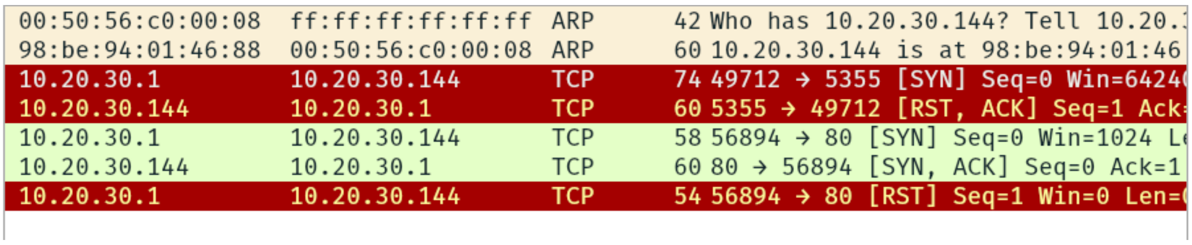
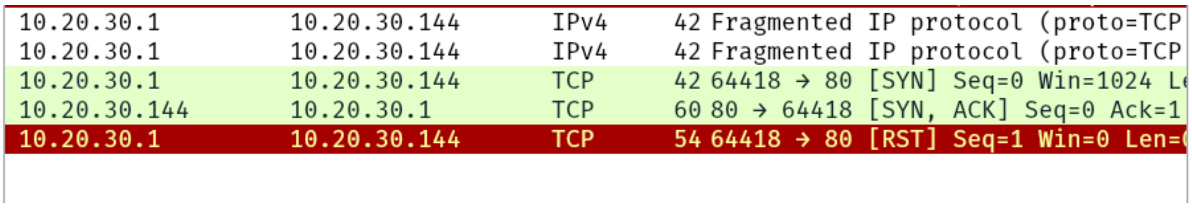


Figure 2.3: IP Header



```
00:50:56:c0:00:08  ff:ff:ff:ff:ff:ff  ARP      42 Who has 10.20.30.144? Tell 10.20.
98:be:94:01:46:88  00:50:56:c0:00:08  ARP      60 10.20.30.144 is at 98:be:94:01:46
10.20.30.1         10.20.30.144       TCP      74 49712 → 5355 [SYN] Seq=0 Win=6424
10.20.30.144       10.20.30.1         TCP      60 5355 → 49712 [RST, ACK] Seq=1 Ack
10.20.30.1         10.20.30.144       TCP      58 56894 → 80 [SYN] Seq=0 Win=1024 L
10.20.30.144       10.20.30.1         TCP      60 80 → 56894 [SYN, ACK] Seq=0 Ack=1
10.20.30.1         10.20.30.144       TCP      54 56894 → 80 [RST] Seq=1 Win=0 Len=
```

Figure 2.4: ARP Query and Response



```
10.20.30.1         10.20.30.144       IPv4     42 Fragmented IP protocol (proto=TCP
10.20.30.1         10.20.30.144       IPv4     42 Fragmented IP protocol (proto=TCP
10.20.30.1         10.20.30.144       TCP      42 64418 → 80 [SYN] Seq=0 Win=1024 L
10.20.30.144       10.20.30.1         TCP      60 80 → 64418 [SYN, ACK] Seq=0 Ack=1
10.20.30.1         10.20.30.144       TCP      54 64418 → 80 [RST] Seq=1 Win=0 Len=
```

Figure 2.5: ARP Query and Response

append to your packets.

Port Scan Type	Example Command
TCP Null Scan	sudo nmap -sN MACHINE_IP
TCP FIN Scan	sudo nmap -sF MACHINE_IP
TCP Xmas Scan	sudo nmap -sX MACHINE_IP
TCP ACK Scan	sudo nmap -sA MACHINE_IP
TCP Window Scan	sudo nmap -sW MACHINE_IP
Custom TCP Scan	sudo nmap –scanflags URGACKPSHRSTSYNFIN MACHINE_IP
Spoofed Source IP	sudo nmap -S SPOOFED_IP MACHINE_IP
Spoofed MAC Address	–spoof-mac SPOOFED_MAC
Decoy Scan	nmap -D DECOY_IP,ME MACHINE_IP
Fragment IP data into 8 bytes	-f
Fragment IP data into 16 bytes	-ff

Table 2.1: Caption

2.6 Summary

S
S

ummary of commands is provided in Table **??**.