Acknowledgment

This material is taken from /based on the forum “TryHackMe” (https://tryhackme.com/paths)

Nmap

Room 1: Nmap Advanced Port Scans

Task 1: Introduction

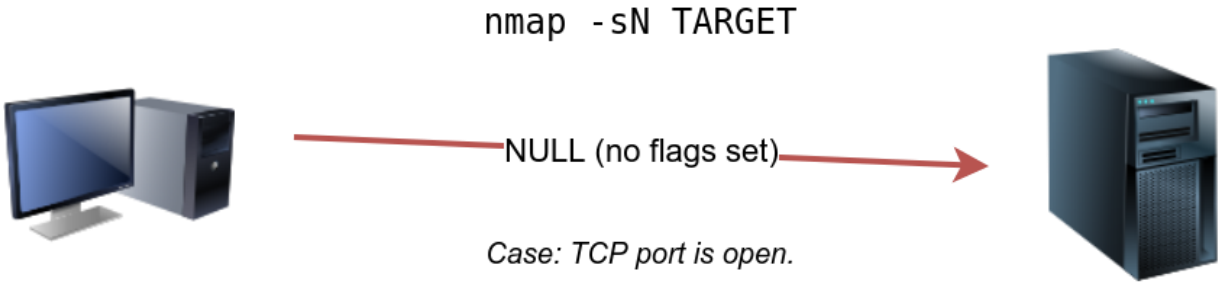
Security researchers and hackers contemplated the TCP flags, shown in the figure below and explained in the previous room, and started to experiment. They wanted to know what would happen if we send a TCP packet, which is not part of any ongoing TCP connection, with one or more flags set.
For instance, an ACK flag is set when you want to acknowledge received data. An ACK scan is like trying to acknowledge data that was neither sent nor received in the first place. Consider this simple analogy, someone coming to you out of nowhere to tell you, “yes, I hear you, please continue.” when you haven’t said anything.
Advanced types of scans and scan options are explained. Some of these scan types can be useful against specific systems, while others are useful in particular network setups. Following types of port scans are covered in this module:
Null Scan, FIN Scan, Xmas Scan, Maimon Scan, ACK Scan, Window Scan, Custom ScanThe following are also covered: Spoofing IP, Spoofing MAC, Decoy Scan, Fragmented Packets, Idle/Zombie Scan

Task2: TCP Null Scan, Full Scan and Xmas Scan

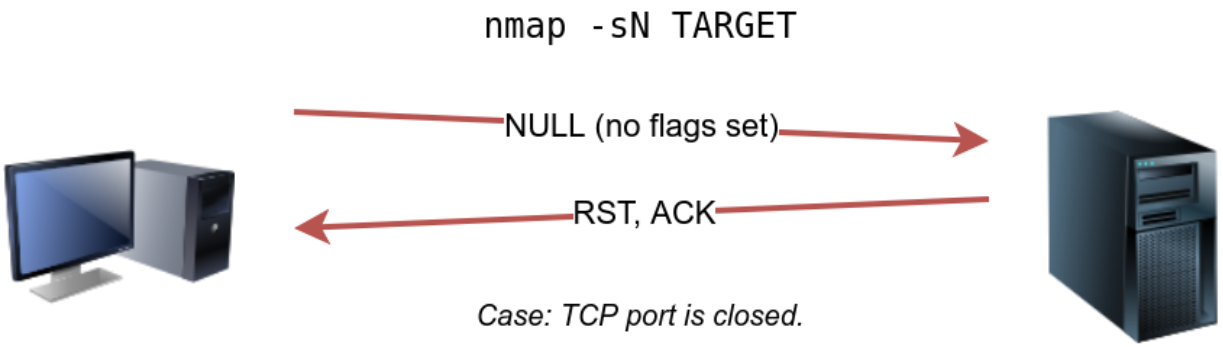
Many Nmap options require root privileges. Unless you are running Nmap as root, you need to use `sudo` as in the example above using the `-sN` option.

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, as shown in the figure below. 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.



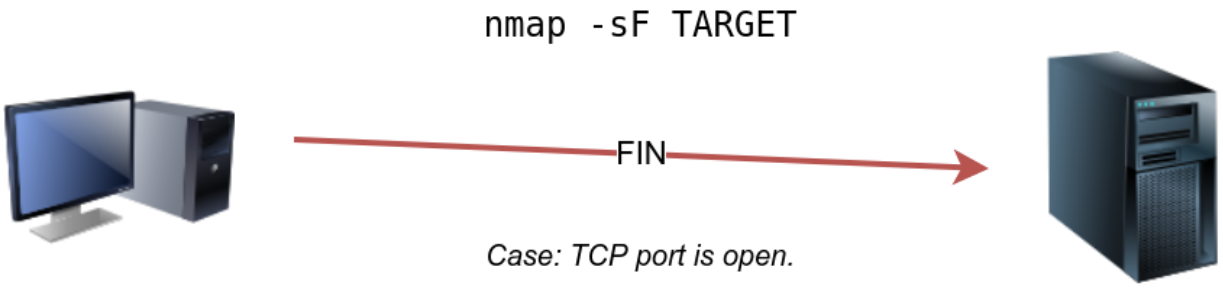
Below is an example of a null scan against a Linux server. The null scan we carried out has successfully identified the six open ports on the target system. Because the null scan relies on the lack of a response to infer that the port is not closed, it cannot indicate with certainty that these ports are open; there is a possibility that the ports are not responding due to a firewall rule.

```
Pentester Terminal

pentester@TryHackMe$ sudo nmap -sN MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:30 BST
Nmap scan report for MACHINE_IP
Host is up (0.00066s latency).
Not shown: 994 closed ports
PORT    STATE         SERVICE
22/tcp  open|filtered ssh
25/tcp  open|filtered smtp
80/tcp  open|filtered http
110/tcp open|filtered pop3
111/tcp open|filtered rpcbind
143/tcp open|filtered imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 96.50 seconds
```

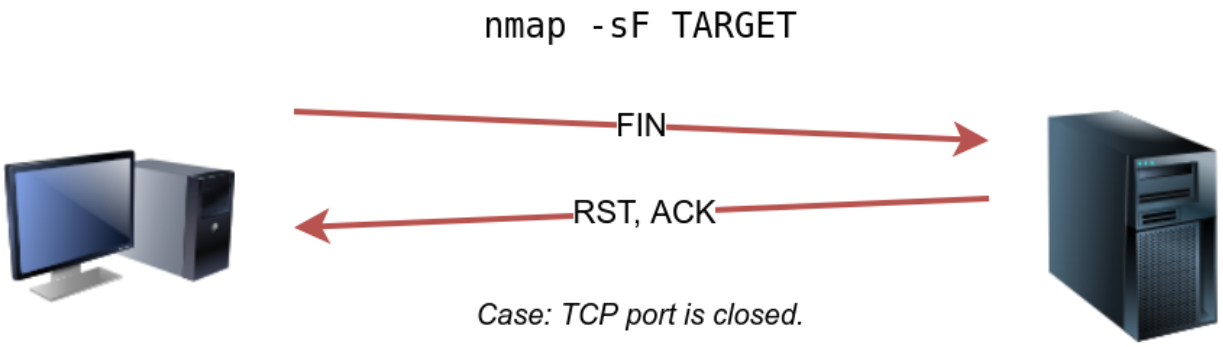
Note that many Nmap options require root privileges. Unless you are running Nmap as root, you need to use `sudo` as in the example above using the `-sN` option.

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.



Below is an example of a FIN scan against a Linux server. The result is quite similar to the result we obtained earlier using a null scan.

Pentester Terminal

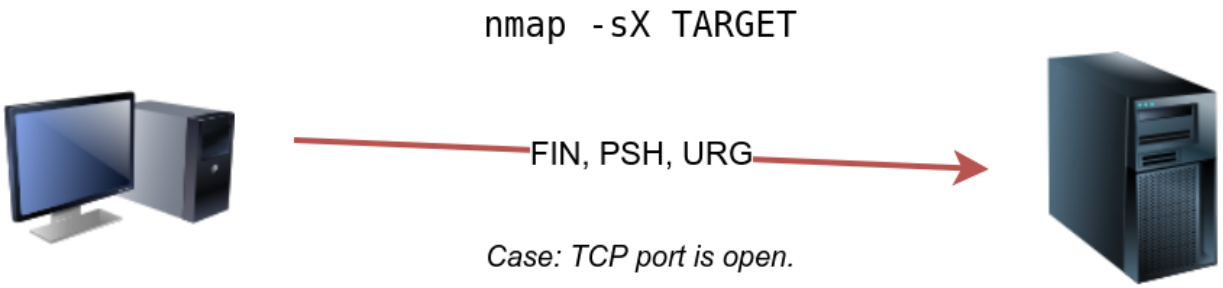
```
pentester@TryHackMe$ sudo nmap -sF MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:32 BST
Nmap scan report for MACHINE_IP
Host is up (0.0018s latency).
Not shown: 994 closed ports
PORT    STATE         SERVICE
22/tcp  open|filtered ssh
25/tcp  open|filtered smtp
80/tcp  open|filtered http
110/tcp open|filtered pop3
111/tcp open|filtered rpcbind
143/tcp open|filtered imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 96.52 seconds
```

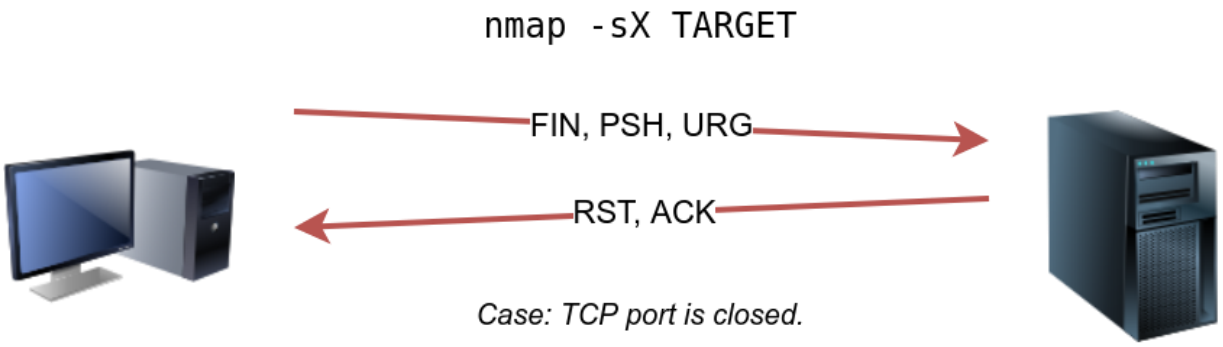
Xmas Scan

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.

The following two figures show the case when the TCP port is open and the case when the TCP port is closed.





The console output below shows an example of a Xmas scan against a Linux server. The obtained results are pretty similar to that of the null scan and the FIN scan.

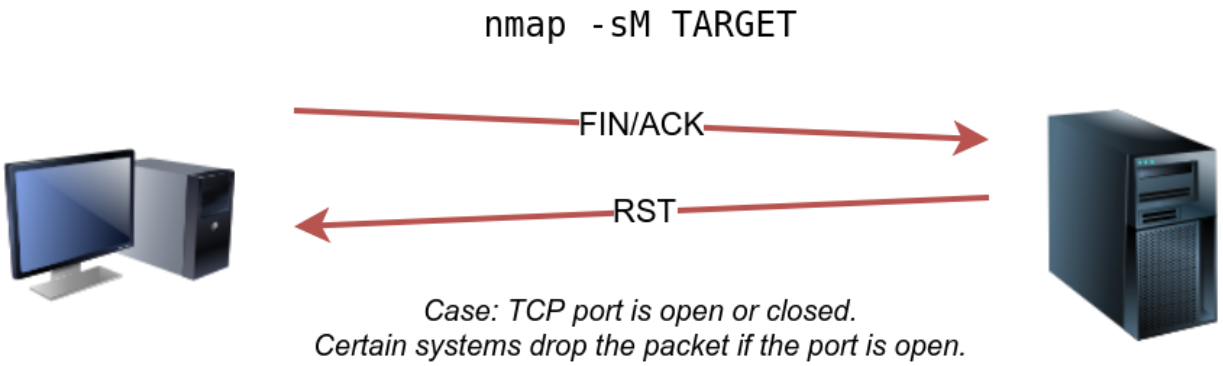
```
Pentester Terminal

pentester@TryHackMe$ sudo nmap -sX MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:34 BST
Nmap scan report for MACHINE_IP
Host is up (0.00087s latency).
Not shown: 994 closed ports
PORT    STATE         SERVICE
22/tcp  open|filtered ssh
25/tcp  open|filtered smtp
80/tcp  open|filtered http
110/tcp open|filtered pop3
111/tcp open|filtered rpcbind
143/tcp open|filtered imap
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 84.85 seconds
```

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.

Task3: TCP Maimon Scan

Uriel Maimon first described this scan in 1996. In this scan, the FIN and ACK bits are set. The target should send an RST packet as a response. However, certain BSD-derived systems drop the packet if it is an open port exposing the open ports. This scan won't work on most targets encountered in modern networks; however, we include it in this room to better understand the port scanning mechanism and the hacking mindset. To select this scan type, use the `-sM` option.

Most target systems respond with an RST packet regardless of whether the TCP port is open. In such a case, we won't be able to discover the open ports. The figure below shows the expected behaviour in the cases of both open and closed TCP ports.



The console output below is an example of a TCP Maimon scan against a Linux server. As mentioned, because open ports and closed ports are behaving the same way, the Maimon scan could not discover any open ports on the target system.

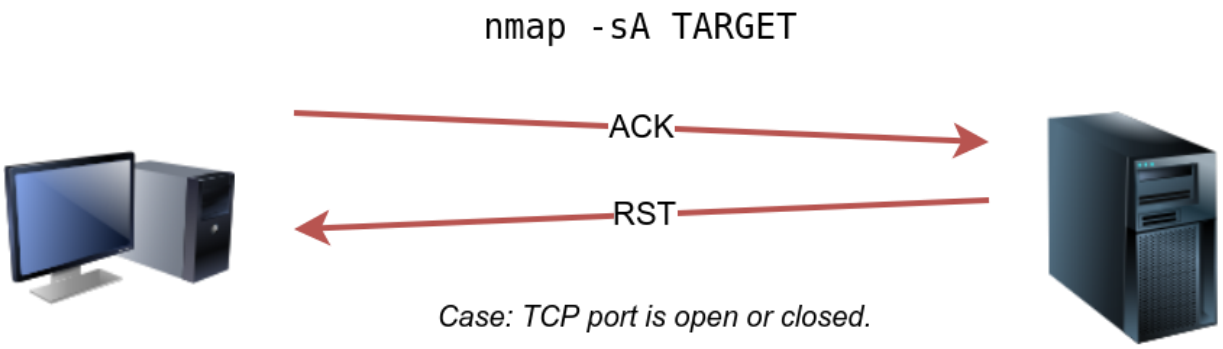
```
Pentester Terminal

pentester@TryHackMe$ sudo nmap -sM 10.10.252.27

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:36 BST
Nmap scan report for ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)
Host is up (0.00095s latency).
All 1000 scanned ports on ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27) are closed
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.61 seconds
```

Task4: TCP ACK, Window, Custom Scan

This task will cover how to perform a TCP ACK scan, a TCP window scan, and how to create your custom flag scan.

TCP ACK Scan

Let's start with the TCP ACK scan. As the name implies, an ACK scan will send a TCP packet with the ACK flag set. Use the -sA option to choose this scan. As we show in the figure below, 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, unlike our case. Hence, this scan won't tell us whether the target port is open in a simple setup.



In the following example, we scanned the target VM before installing a firewall on it. As expected, we couldn't learn which ports were open.

```
Pentester Terminal

pentester@TryHackMe$ sudo nmap -sA MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:37 BST
Nmap scan report for MACHINE_IP
Host is up (0.0013s latency).
All 1000 scanned ports on MACHINE_IP are unfiltered
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.68 seconds
```

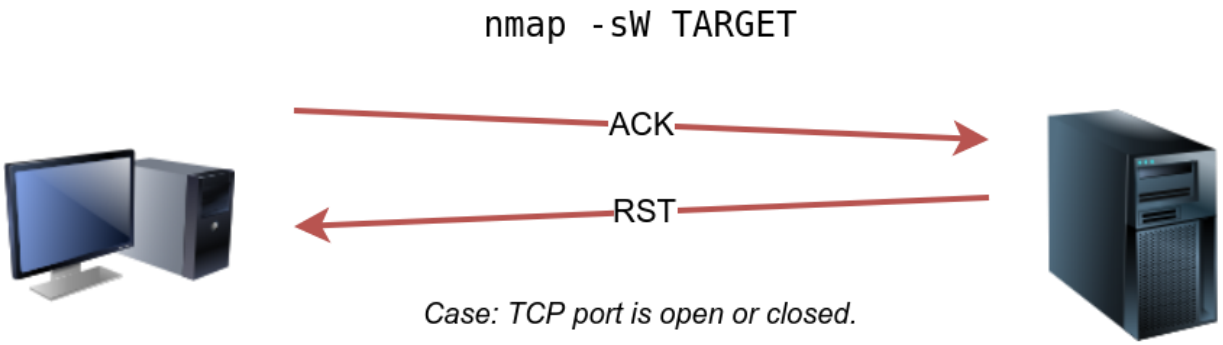
This kind of scan would be helpful if there is a firewall in front of the target. Consequently, based on which ACK packets resulted in responses, you will learn which ports were not blocked by the firewall. In other words, this type of scan is more suitable to discover firewall rule sets and configuration.

After setting up the target VM MACHINE_IP with a firewall, we repeated the ACK scan. This time, we received some interesting results. As seen in the console output below, we have three ports that aren't being blocked by the firewall. This result indicates that the firewall is blocking all other ports except for these three ports.

```
Pentester Terminal

pentester@TryHackMe$ sudo nmap -sA MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-07 11:34 BST
Nmap scan report for MACHINE_IP
Host is up (0.00046s latency).
Not shown: 997 filtered ports
PORT    STATE      SERVICE
22/tcp  unfiltered ssh
25/tcp  unfiltered smtp
80/tcp  unfiltered http
MAC Address: 02:78:C0:D0:4E:E9 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 15.45 seconds
```

Window Scan

Another similar scan is the 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. As shown in the figure below, we expect to get an RST packet in reply to our “uninvited” ACK packets, regardless of whether the port is open or closed.



Similarly, launching a TCP window scan against a Linux system with no firewall will not provide much information. As we can see in the console output below, the results of the window scan against a Linux server with no firewall didn't give any extra information compared to the ACK scan executed earlier.

Pentester Terminal

```
pentester@TryHackMe$ sudo nmap -sW MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:38 BST
Nmap scan report for MACHINE_IP
Host is up (0.0011s latency).
All 1000 scanned ports on ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27) are closed
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

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

However, as you would expect, if we repeat our TCP window scan against a server behind a firewall, we expect to get more satisfying results. In the console output shown below, the TCP window scan pointed that three ports are detected as closed. (This is in contrast with the ACK scan that labelled the same three ports as unfiltered.) Although we know that these three ports are not closed, we realize they responded differently, indicating that the firewall does not block them.

Pentester Terminal

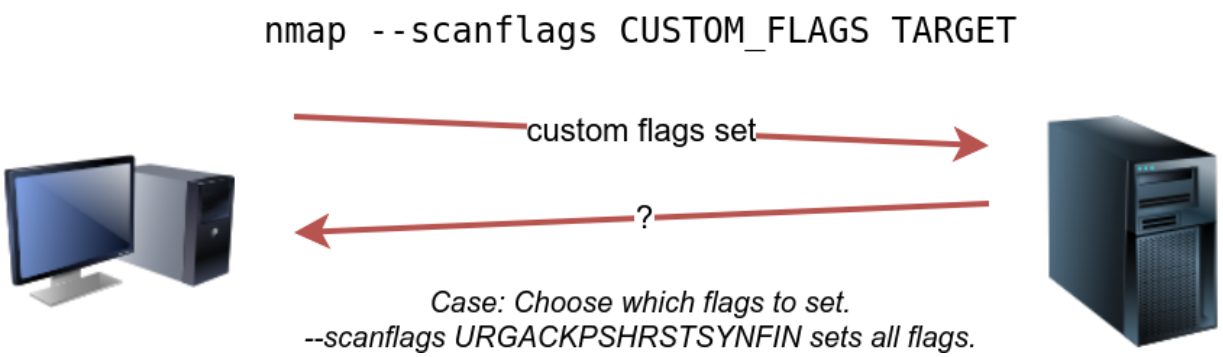
```
pentester@TryHackMe$ sudo nmap -sW MACHINE_IP

Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-07 11:39 BST
Nmap scan report for MACHINE_IP
Host is up (0.00040s latency).
```

```
Not shown: 997 filtered ports
PORT    STATE  SERVICE
22/tcp  closed ssh
25/tcp  closed smtp
80/tcp  closed http
MAC Address: 02:78:C0:D0:4E:E9 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 14.84 seconds
```

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. As shown in the figure below, if you develop your custom scan, you need to know how the different ports will behave to interpret the results in different scenarios correctly.

x



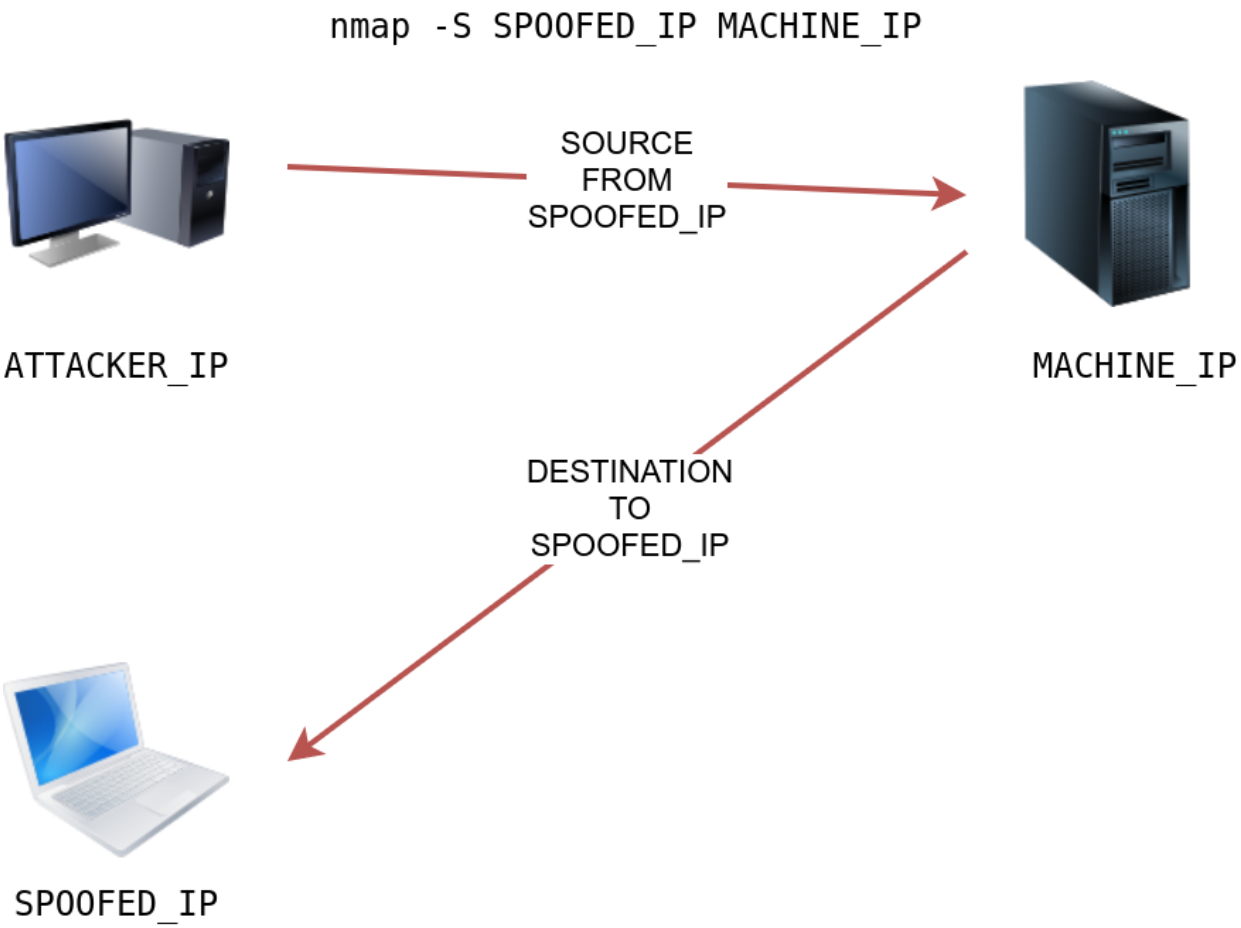
Finally, it is essential to note that the ACK scan and the window scan were very efficient at helping us map out the firewall rules. However, it is vital to remember that just because a firewall is not blocking a specific port, it does not necessarily mean that a service is listening on that port. For 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.

Task5: 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 following figure shows 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.



Using SPOOFED_IP as the source IP address for the scan.

In brief, scanning with a spoofed IP address is three steps:

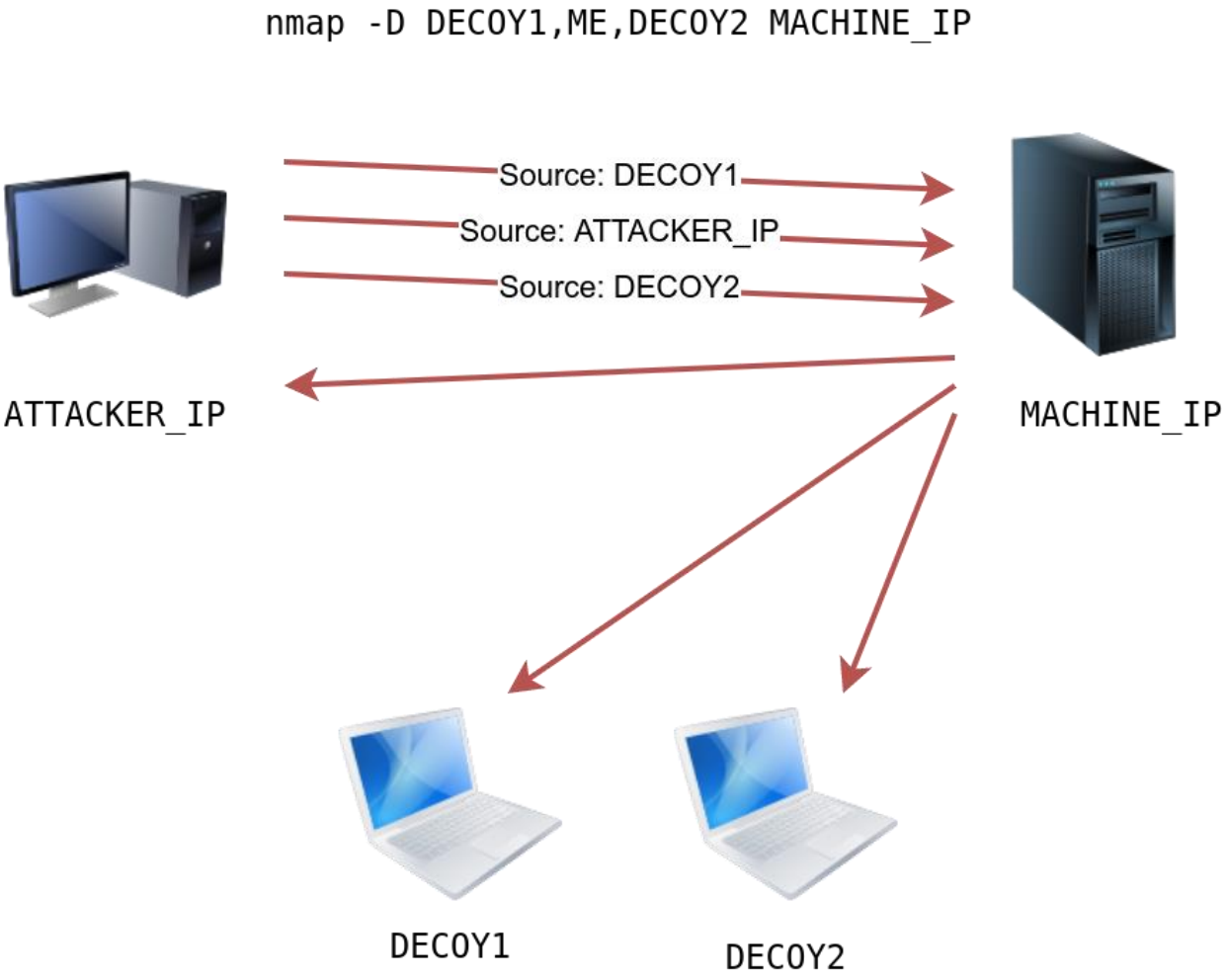
1. Attacker sends a packet with a spoofed source IP address to the target machine.
2. Target machine replies to the spoofed IP address as the destination.
3. Attacker captures the replies to figure out open ports.

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 we see in the figure below, 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.



Using DECOY1, ATTACKER_IP, DECOY2 as the source IP addresses for the scan.

You can launch a decoy scan by specifying a specific or random IP address after -D. For example, 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.

Task6: Fragmented Packets

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.

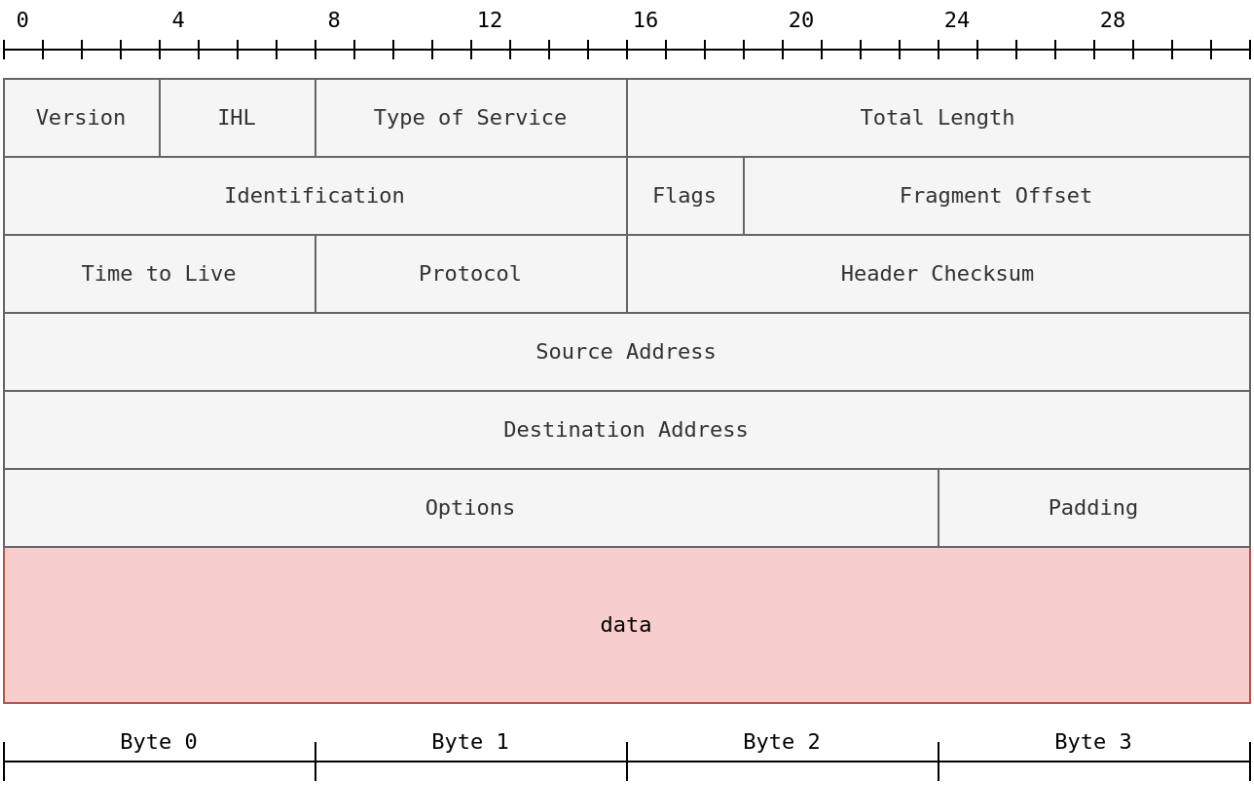
IDS

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.

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`; however, you should always choose a multiple of 8.

To properly understand fragmentation, we need to look at the IP header in the figure below. 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 figure below.



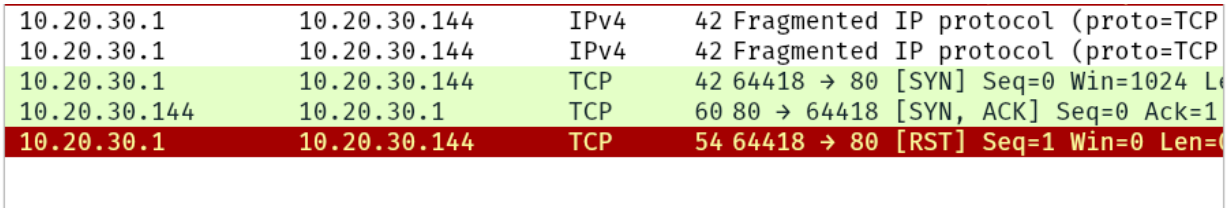
Let's compare running `sudo nmap -sS -p80 10.20.30.144` and `sudo nmap -sS -p80 -f 10.20.30.144`. As you know by now, this will use stealth TCP SYN scan on port 80; however, in the second command, we are requesting Nmap to fragment the IP packets.

In the first two lines, 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.

```
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=
```

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 between the fifth and the seventh lines.



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 append to your packets.

Summary

This room covered the following types of scans.

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 Maimon Scan	`sudo nmap -sM 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`

Option	Purpose
`--source-port PORT_NUM`	specify source port number
`--data-length NUM`	append random data to reach given length

These scan types rely on setting TCP flags in unexpected ways to prompt ports for a reply. Null, FIN, and Xmas scan provoke a response from closed ports, while Maimon, ACK, and Window scans provoke a response from open and closed ports.

Option	Purpose
`--reason`	explains how Nmap made its conclusion

Option	Purpose
-v	verbose
-vv	very verbose
-d	debugging
-dd	more details for debugging