Nmap

Summary & Cheat Sheet

We have learned how ARP, ICMP, TCP, and UDP can detect live hosts. Any response from a host is an indication that it is online. Below is a quick summary of the command-line options for Nmap that we have covered.

Scan Type	Example Command
ARP Scan	`sudo nmap -PR -sn MACHINE_IP/24`
ICMP Echo Scan	`sudo nmap -PE -sn MACHINE_IP/24`
ICMP Timestamp Scan	`sudo nmap -PP -sn MACHINE_IP/24`
ICMP Address Mask Scan	`sudo nmap -PM -sn MACHINE_IP/24`
TCP SYN Ping Scan	`sudo nmap -PS22,80,443 -sn MACHINE_IP/30`
TCP ACK Ping Scan	`sudo nmap -PA22,80,443 -sn MACHINE_IP/30`
UDP Ping Scan	`sudo nmap -PU53,161,162 -sn MACHINE_IP/30`

Remember to add `-sn` if you are only interested in host discovery without port-scanning. Omitting `-sn` will let Nmap default to port-scanning the live hosts.

Option	Purpose
`-n`	no DNS lookup
`-R`	reverse-DNS lookup for all hosts
`-sn`	host discovery only

Nmap Basic Port Scan room covered three types of scans.

Port Scan Type	Example Command
TCP Connect Scan	`nmap -sT 10.10.203.74`
TCP SYN Scan	`sudo nmap -sS 10.10.203.74`
UDP Scan	`sudo nmap -sU 10.10.203.74`

These scan types should get you started discovering running TCP and UDP services on a target host.

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

Nmap Advanced Port Scan room covered the following types of scans.

Port Scan Type	Example Command
TCP Null Scan	`sudo nmap -sN 10.10.98.74`
TCP FIN Scan	`sudo nmap -sF 10.10.98.74`
TCP Xmas Scan	`sudo nmap -sX 10.10.98.74`
TCP Maimon Scan	`sudo nmap -sM 10.10.98.74`
TCP ACK Scan	`sudo nmap -sA 10.10.98.74`
TCP Window Scan	`sudo nmap -sW 10.10.98.74`
Custom TCP Scan	`sudo nmap --scanflags URGACKPSHRSTSYNFIN 10.10.98.74`
Spoofed Source IP	`sudo nmap -S SPOOFED_IP 10.10.98.74`
Spoofed MAC Address	`--spoof-mac SPOOFED_MAC`
Decoy Scan	`nmap -D DECOY_IP,ME 10.10.98.74`
Idle (Zombie) Scan	`sudo nmap -sI ZOMBIE_IP 10.10.98.74`
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
`-v`	verbose
`-vv`	very verbose
`-d`	debugging
`-dd`	more details for debugging

In Post Port Scan room, we learned how to detect the running services and their versions along with the host operating system. We learned how to enable traceroute and we covered selecting one or more scripts to aid in penetration testing. Finally, we covered the different formats to save the scan results for future reference. The table below summarizes the most important options we covered in this room.

Option	Meaning
`-sV`	determine service/version info on open ports
`-sV --version-light`	try the most likely probes (2)
`-sV --version-all`	try all available probes (9)
`-O`	detect OS
`--traceroute`	run traceroute to target
`--script=SCRIPTS`	Nmap scripts to run
`-sC` or `--script=default`	run default scripts
`-A`	equivalent to `-sV -O -sC --traceroute`
`-oN`	save output in normal format
`-oG`	save output in grepable format
`-oX`	save output in XML format
`-oA`	save output in normal, XML and Grepable formats

Nmap Live Host Discovery

Introduction:

Nmap is an industry-standard tool for mapping networks, identifying live hosts, and discovering running services.

When we want to target a network, we want to find an efficient tool to help us handle repetitive tasks and answer the following questions:

1. Which systems are up?
2. What services are running on these systems?

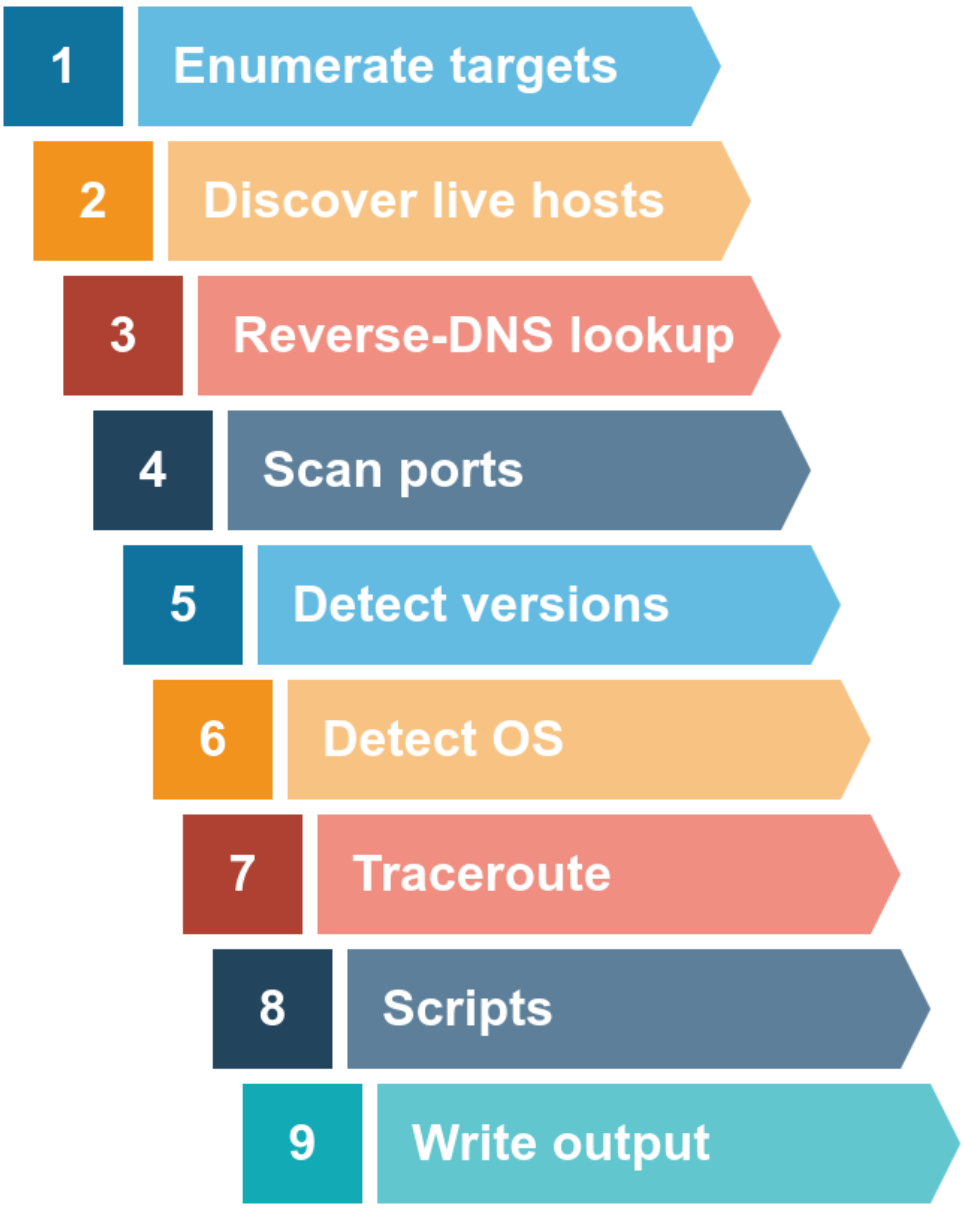
The tool that we will rely on is Nmap.

This part explains the steps that Nmap carries out to discover the systems that are online before port-scanning. This stage is crucial because trying to port-scan offline systems will only waste time and create unnecessary noise on the network.

We present the different approaches that Nmap uses to discover live hosts. In particular, we cover:

1. ARP scan: This scan uses ARP requests to discover live hosts
2. ICMP scan: This scan uses ICMP requests to identify live hosts
3. TCP/UDP ping scan: This scan sends packets to TCP ports and UDP ports to determine live hosts.

A Nmap scan usually goes through the steps shown in the figure below, although many are optional and depend on the command-line arguments you provide.

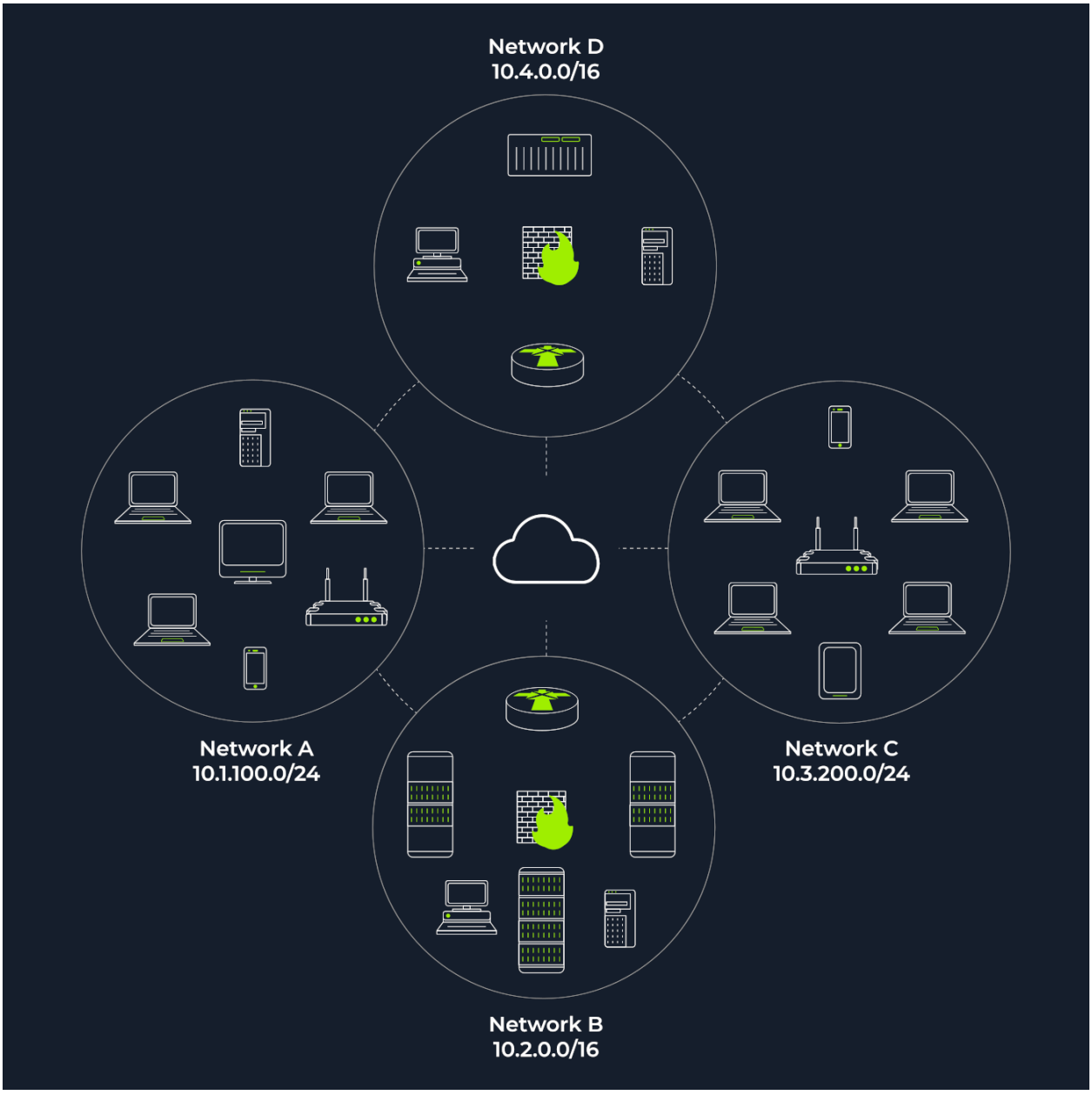


Subnetworks:

Let's review a couple of terms before we move on to the main tasks. A *network segment* is a group of computers connected using a shared medium. For instance, the medium can be the Ethernet switch or WiFi access point. In an IP network, a *subnetwork* is usually the equivalent of one or more network segments connected together and configured to use the same router. The network segment refers to a physical connection, while a subnetwork refers to a logical connection.

In the following network diagram, we have four network segments or subnetworks. Generally speaking, your system would be connected to one of these network

segments/subnetworks. A subnetwork, or simply a subnet, has its own IP address range and is connected to a more extensive network via a router. There might be a firewall enforcing security policies depending on each network.



The figure above shows two types of subnets:

- Subnets with `/16`, which means that the subnet mask can be written as `255.255.0.0`. This subnet can have around 65 thousand hosts.

- Subnets with `/24`, which indicates that the subnet mask can be expressed as `255.255.255.0`. This subnet can have around 250 hosts.

As part of active reconnaissance, we want to discover more information about a group of hosts or about a subnet. If you are connected to the same subnet, you would expect your scanner to rely on ARP (Address Resolution Protocol) queries to discover live hosts. An ARP query aims to get the hardware address (MAC address) so that communication over the link-layer becomes possible; however, we can use this to infer that the host is online. (We revisit link-layer in Task 4.)

If you are in Network A, you can use ARP only to discover the devices within that subnet (10.1.100.0/24). Suppose you are connected to a subnet different from the subnet of the target system(s). In that case, all packets generated by your scanner will be routed via the default gateway (router) to reach the systems on another subnet; however, the ARP queries won't be routed and hence cannot cross the subnet router. ARP is a link-layer protocol, and ARP packets are bound to their subnet.

Enumerating Targets:

Generally speaking, you can provide a list, a range, or a subnet. Examples of target specification are:

- list: `MACHINE_IP scanme.nmap.org example.com` will scan 3 IP addresses.
- range: `10.11.12.15-20` will scan 6 IP addresses: `10.11.12.15`, `10.11.12.16`,… and `10.11.12.20`.
- subnet: `MACHINE_IP/30` will scan 4 IP addresses.

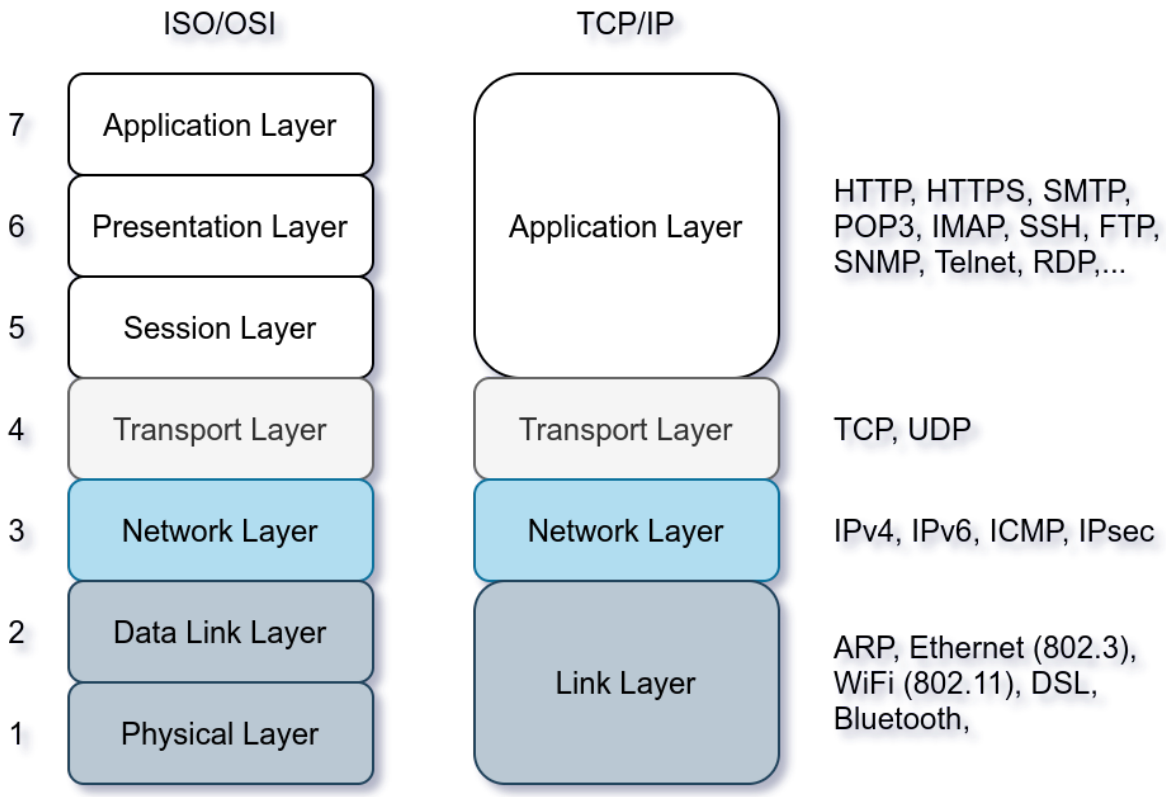
You can also provide a file as input for your list of targets, `nmap -iL list_of_hosts.txt`.

If you want to check the list of hosts that Nmap will scan, you can use `nmap -sL TARGETS`. This option will give you a detailed list of the hosts that Nmap will scan without scanning them; however, Nmap will attempt a reverse-DNS resolution on all the targets to obtain their names. Names might reveal various information to the pentester. (If you don't want Nmap to the DNS server, you can add `-n`.)

Discovering Live Hosts:

Let's revisit the TCP/IP layers shown in the figure next. We will leverage the protocols to discover the live hosts. Starting from bottom to top, we can use:

- ARP from Link Layer
- ICMP from Network Layer
- TCP from Transport Layer
- UDP from Transport Layer



ARP has one purpose: sending a frame to the broadcast address on the network segment and asking the computer with a specific IP address to respond by providing its MAC (hardware) address.

ICMP has many types. ICMP ping uses Type 8 (Echo) and Type 0 (Echo Reply).

If you want to ping a system on the same subnet, an ARP query should precede the ICMP Echo.

Although TCP and UDP are transport layers, for network scanning purposes, a scanner can send a specially-crafted packet to common TCP or UDP ports to check whether the target will respond. This method is efficient, especially when ICMP Echo is blocked.

Nmap Host Discovery Using ARP:

How would we know which hosts are up and running? It is essential to avoid wasting our time port-scanning an offline host or an IP address not in use. There are various ways to discover online hosts. When no host discovery options are provided, Nmap follows the following approaches to discover live hosts:

1. When a *privileged* user tries to scan targets on a local network (Ethernet), Nmap uses *ARP requests*. A privileged user is `root` or a user who belongs to `sudoers` and can run `sudo`.
2. When a *privileged* user tries to scan targets outside the local network, Nmap uses ICMP echo requests, TCP ACK (Acknowledge) to port 80, TCP SYN (Synchronize) to port 443, and ICMP timestamp request.
3. When an *unprivileged* user tries to scan targets outside the local network, Nmap resorts to a TCP 3-way handshake by sending SYN packets to ports 80 and 443.

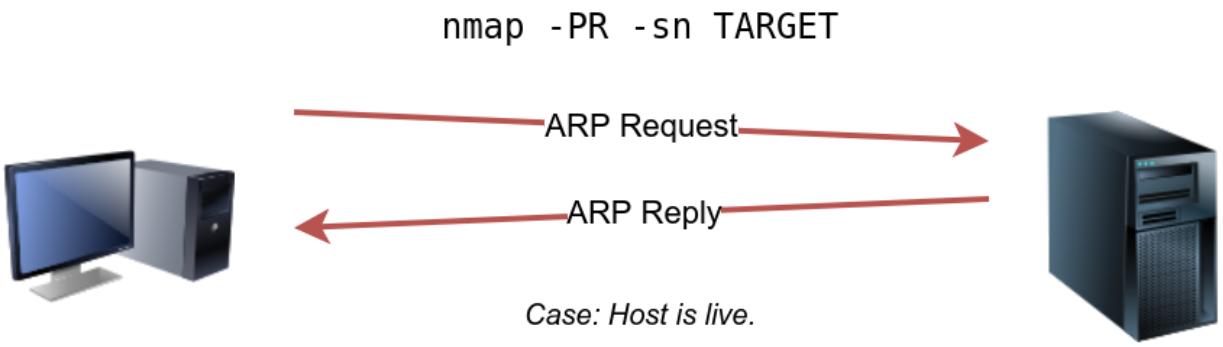
Nmap, by default, uses a ping scan to find live hosts, then proceeds to scan live hosts only. If you want to use Nmap to discover online hosts without port-scanning the live systems, you can issue `nmap -sn TARGETS`. Let's dig deeper to gain a solid understanding of the different techniques used.

ARP scan is possible only if you are on the same subnet as the target systems. On an Ethernet (802.3) and WiFi (802.11), you need to know the MAC address of any system before you can communicate with it. The MAC address is necessary for the link-layer header; the header contains the source MAC address and the destination MAC address among other fields. To get the MAC address, the OS sends an ARP query. A host that replies to ARP queries is up. The ARP query only works if the target is on the same subnet as yourself, i.e., on the same Ethernet/WiFi. You should expect to see many ARP queries generated during a Nmap scan of a local network. If you want Nmap only to perform an ARP scan without port-scanning, you can use `nmap -PR -sn TARGETS`, where `-PR` indicates that you only want an ARP scan. The following example shows Nmap using ARP for host discovery without any port scanning. We run `nmap -PR -sn MACHINE_IP/24` to discover all the live systems on the same subnet as our target machine.

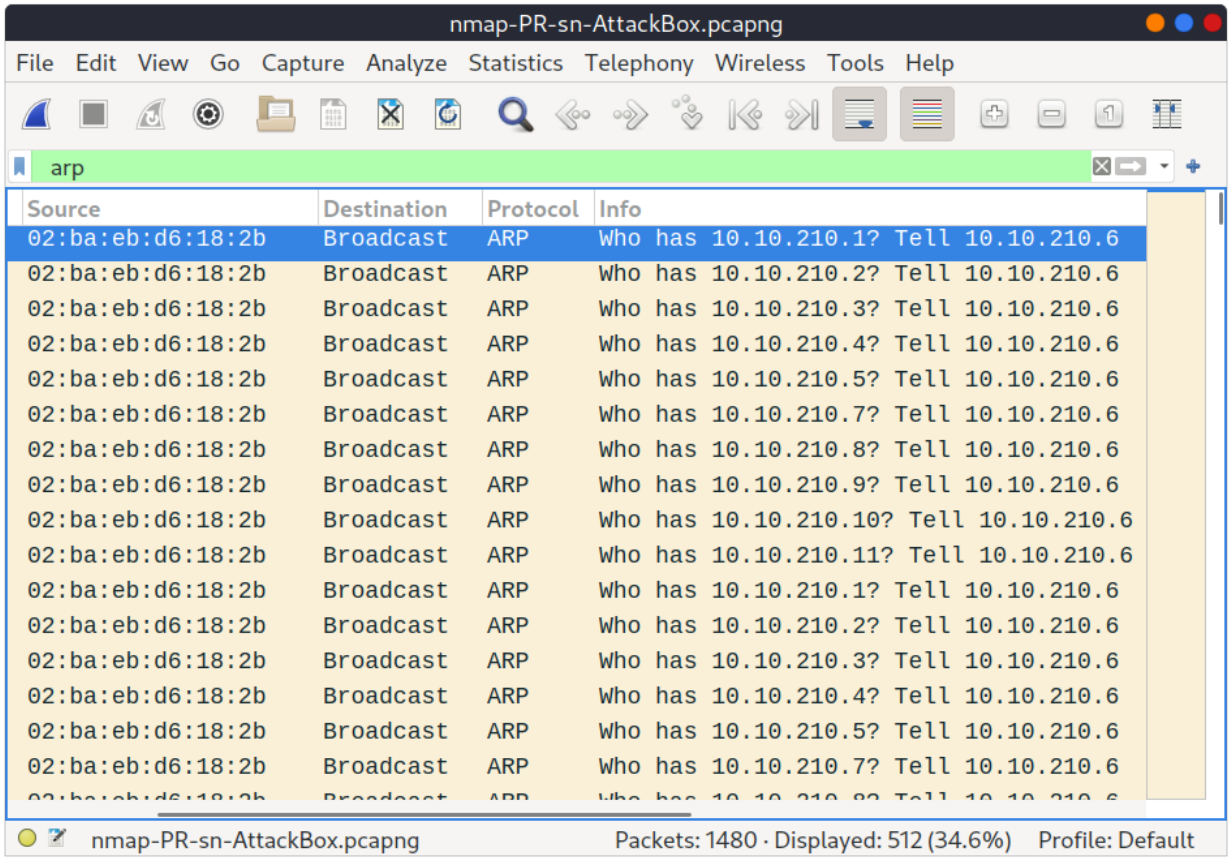
```
pentester@TryHackMe$ sudo nmap -PR -sn 10.10.210.6/24
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-02 07:12 BST
Nmap scan report for ip-10-10-210-75.eu-west-1.compute.internal (10.10.210.75)
Host is up (0.00013s latency).
```

```
MAC Address: 02:83:75:3A:F2:89 (Unknown)
Nmap scan report for ip-10-10-210-100.eu-west-1.compute.internal (10.10.210.100)
Host is up (-0.100s latency).
MAC Address: 02:63:D0:1B:2D:CD (Unknown)
Nmap scan report for ip-10-10-210-165.eu-west-1.compute.internal (10.10.210.165)
Host is up (0.00025s latency).
MAC Address: 02:59:79:4F:17:B7 (Unknown)
Nmap scan report for ip-10-10-210-6.eu-west-1.compute.internal (10.10.210.6)
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.12 seconds
```

In this case, the AttackBox had the IP address 10.10.210.6, and it used ARP requests to discover the live hosts on the same subnet. ARP scan works, as shown in the figure below. Nmap sends ARP requests to all the target computers, and those online should send an ARP reply back.



If we look at the packets generated using a tool such as tcpdump or Wireshark, we will see network traffic similar to the figure below. In the figure below, Wireshark displays the source MAC address, destination MAC address, protocol, and query related to each ARP request. The source address is the MAC address of our AttackBox, while the destination is the broadcast address as we don't know the MAC address of the target. However, we see the target's IP address, which appears in the Info column. In the figure, we can see that we are requesting the MAC addresses of all the IP addresses on the subnet, starting with `10.10.210.1`. The host with the IP address we are asking about will send an ARP reply with its MAC address, and that's how we will know that it is online.



Talking about ARP scans, we should mention a scanner built around ARP queries: `arp-scan`; it provides many options to customize your scan. Visit the arp-scan wiki for detailed information. One popular choice is `arp-scan --localnet` or simply `arp-scan -l`. This command will send ARP queries to all valid IP addresses on your local networks. Moreover, if your system has more than one interface and you are interested in discovering the live hosts on one of them, you can specify the interface using `-I`. For instance, `sudo arp-scan -I eth0 -l` will send ARP queries for all valid IP addresses on the `eth0` interface.

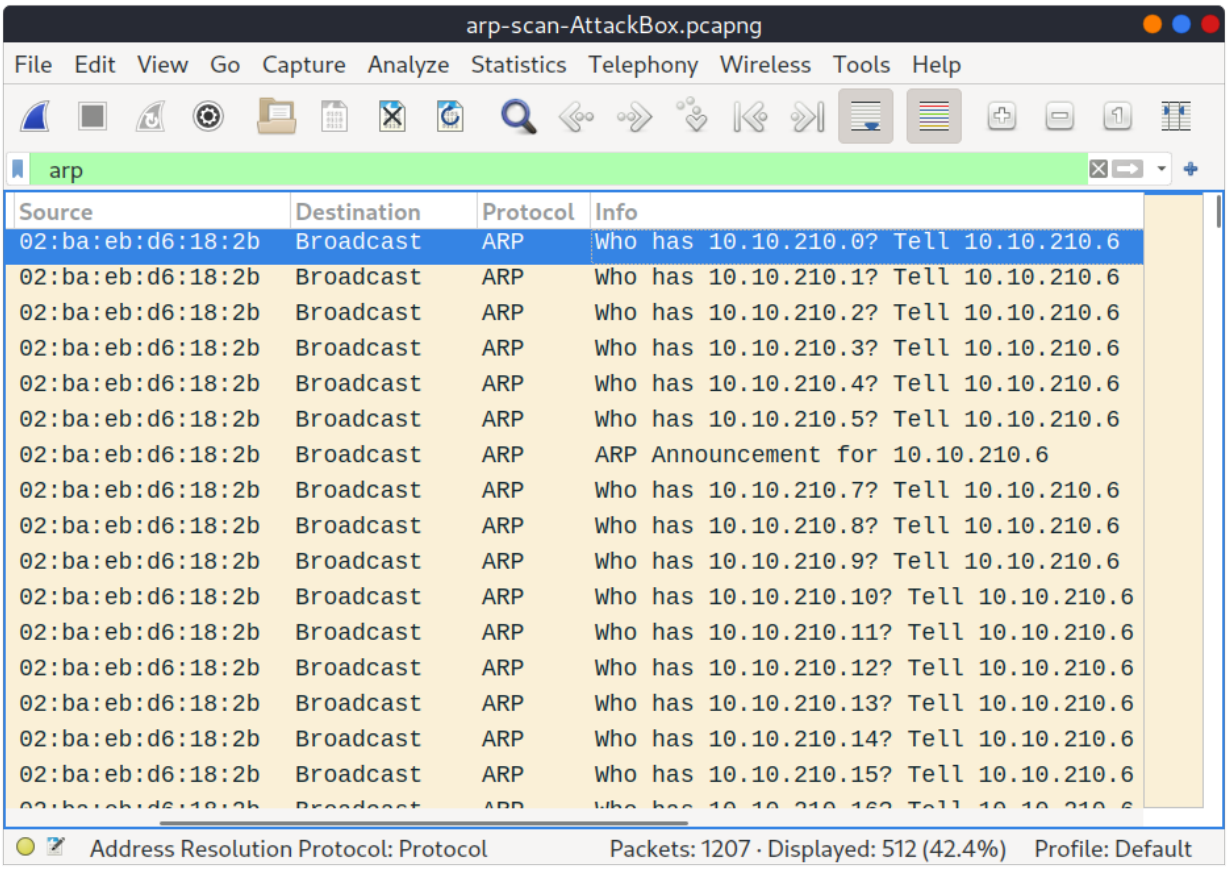
Note that `arp-scan` is not installed on the AttackBox; however, it can be installed using `apt install arp-scan`.

In the example below, we scanned the subnet of the AttackBox using `arp-scan ATTACKBOX_IP/24`. Since we ran this scan at a time frame close to the previous one `nmap -PR -sn ATTACKBOX_IP/24`, we obtained the same three live targets.

```
pentester@TryHackMe$ sudo arp-scan 10.10.210.6/24
Interface: eth0, datalink type: EN10MB (Ethernet)
```

```
WARNING: host part of 10.10.210.6/24 is non-zero
Starting arp-scan 1.9 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
10.10.210.75  02:83:75:3a:f2:89 (Unknown)
10.10.210.100 02:63:d0:1b:2d:cd (Unknown)
10.10.210.165 02:59:79:4f:17:b7 (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9: 256 hosts scanned in 2.726 seconds (93.91 hosts/sec). 3 responded
```

Similarly, the command `arp-scan` will generate many ARP queries that we can see using tcpdump, Wireshark, or a similar tool. We can notice that the packet capture for `arp-scan` and `nmap -PR -sn` yield similar traffic patterns. Below is the Wireshark output.

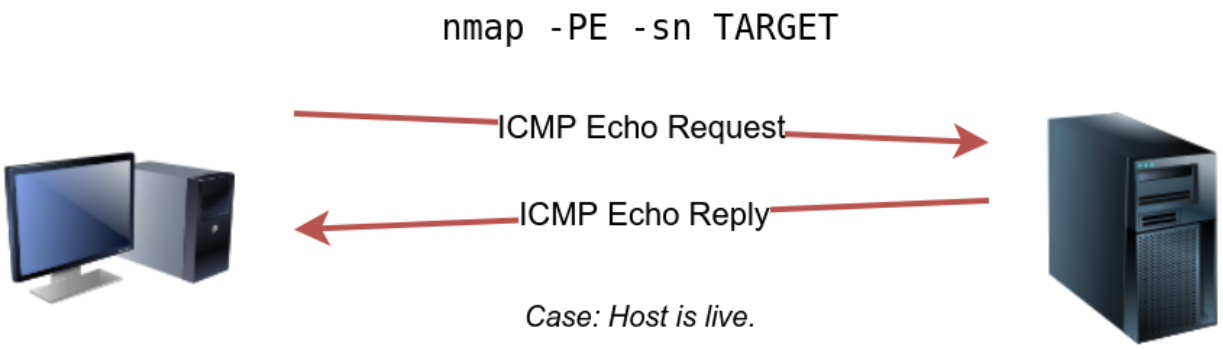


Nmap Host Discovery using ICMP:

We can ping every IP address on a target network and see who would respond to our `ping` (ICMP Type 8/Echo) requests with a ping reply (ICMP Type 0). Simple, isn't it? Although this would be the most straightforward approach, it is not always reliable. Many firewalls block ICMP echo; new versions of MS Windows are configured with a

host firewall that blocks ICMP echo requests by default. Remember that an ARP query will precede the ICMP request if your target is on the same subnet.

To use ICMP echo request to discover live hosts, add the option `-PE`. (Remember to add `-sn` if you don't want to follow that with a port scan.) As shown in the following figure, an ICMP echo scan works by sending an ICMP echo request and expects the target to reply with an ICMP echo reply if it is online.



In the example below, we scanned the target's subnet using `nmap -PE -sn MACHINE_IP/24`. This scan will send ICMP echo packets to every IP address on the subnet. Again, we expect live hosts to reply; however, it is wise to remember that many firewalls block ICMP. The output below shows the result of scanning the virtual machine's class C subnet using `sudo nmap -PE -sn MACHINE_IP/24`.

```
pentester@TryHackMe$ sudo nmap -PE -sn 10.10.68.220/24
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-02 10:16 BST
Nmap scan report for ip-10-10-68-50.eu-west-1.compute.internal (10.10.68.50)
Host is up (0.00017s latency).
MAC Address: 02:95:36:71:5B:87 (Unknown)
Nmap scan report for ip-10-10-68-52.eu-west-1.compute.internal (10.10.68.52)
Host is up (0.00017s latency).
MAC Address: 02:48:E8:BF:78:E7 (Unknown)
Nmap scan report for ip-10-10-68-77.eu-west-1.compute.internal (10.10.68.77)
Host is up (-0.100s latency).
MAC Address: 02:0F:0A:1D:76:35 (Unknown)
Nmap scan report for ip-10-10-68-110.eu-west-1.compute.internal (10.10.68.110)
Host is up (-0.10s latency).
MAC Address: 02:6B:50:E9:C2:91 (Unknown)
Nmap scan report for ip-10-10-68-140.eu-west-1.compute.internal (10.10.68.140)
Host is up (0.00021s latency).
MAC Address: 02:58:59:63:0B:6B (Unknown)
Nmap scan report for ip-10-10-68-142.eu-west-1.compute.internal (10.10.68.142)
Host is up (0.00016s latency).
MAC Address: 02:C6:41:51:0A:0F (Unknown)
```

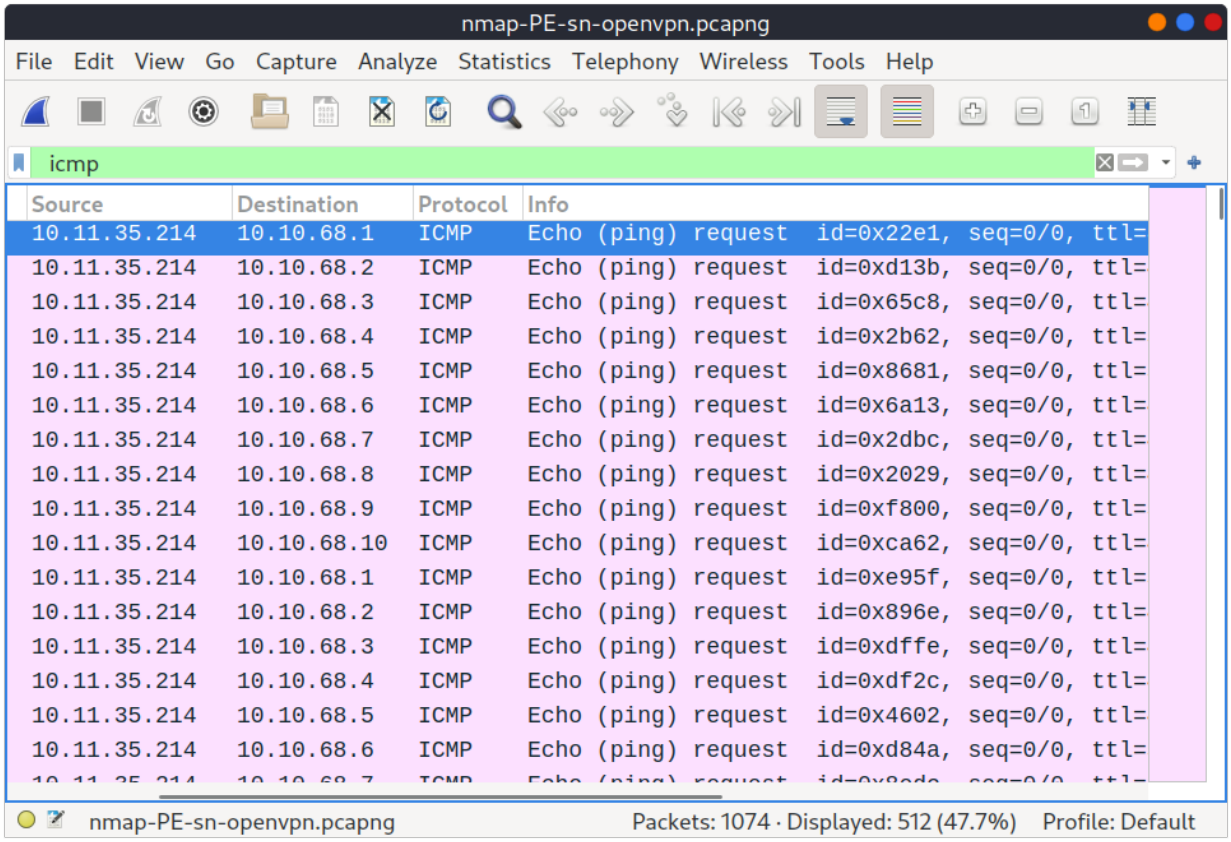
```
Nmap scan report for ip-10-10-68-220.eu-west-1.compute.internal (10.10.68.220)
Host is up (0.00026s latency).
MAC Address: 02:25:3F:DB:EE:0B (Unknown)
Nmap scan report for ip-10-10-68-222.eu-west-1.compute.internal (10.10.68.222)
Host is up (0.00025s latency).
MAC Address: 02:28:B1:2E:B0:1B (Unknown)
Nmap done: 256 IP addresses (8 hosts up) scanned in 2.11 seconds
```

The scan output shows that eight hosts are up; moreover, it shows their MAC addresses. Generally speaking, we don't expect to learn the MAC addresses of the targets unless they are on the same subnet as our system. The output above indicates that Nmap didn't need to send ICMP packets as it confirmed that these hosts are up based on the ARP responses it received.

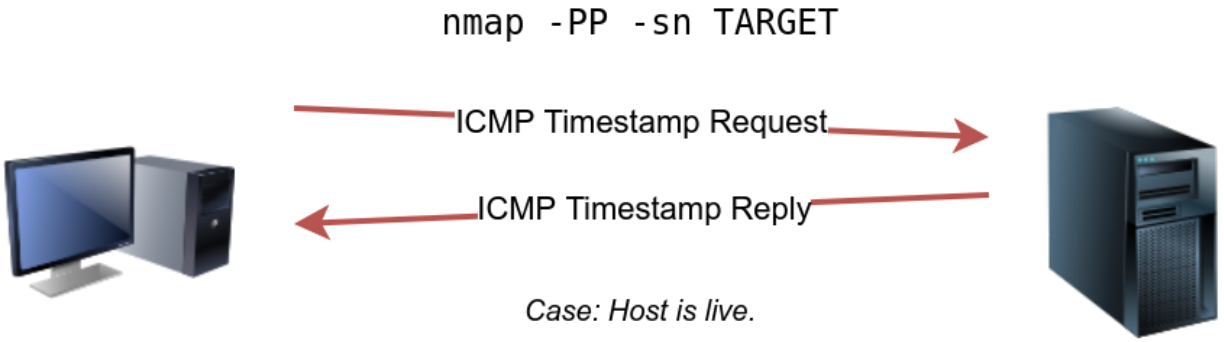
We will repeat the scan above; however, this time, we will scan from a system that belongs to a different subnet. The results are similar but without the MAC addresses.

```
pentester@TryHackMe$ sudo nmap -PE -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 12:16 EEST
Nmap scan report for 10.10.68.50
Host is up (0.12s latency).
Nmap scan report for 10.10.68.52
Host is up (0.12s latency).
Nmap scan report for 10.10.68.77
Host is up (0.11s latency).
Nmap scan report for 10.10.68.110
Host is up (0.11s latency).
Nmap scan report for 10.10.68.140
Host is up (0.11s latency).
Nmap scan report for 10.10.68.142
Host is up (0.11s latency).
Nmap scan report for 10.10.68.220
Host is up (0.11s latency).
Nmap scan report for 10.10.68.222
Host is up (0.11s latency).
Nmap done: 256 IP addresses (8 hosts up) scanned in 8.26 seconds
```

If you look at the network packets using a tool like Wireshark, you will see something similar to the image below. You can see that we have one source IP address on a different subnet than that of the destination subnet, sending ICMP echo requests to all the IP addresses in the target subnet to see which one will reply.



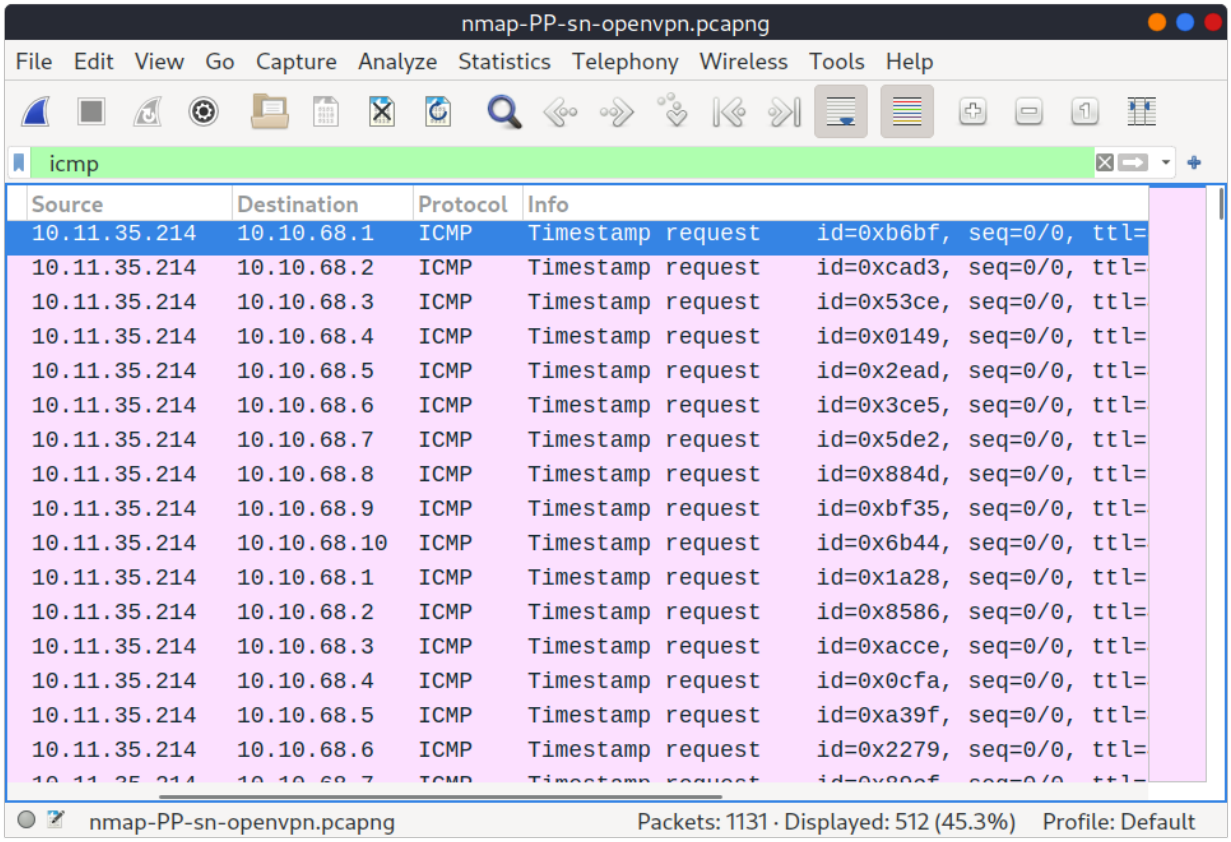
Because ICMP echo requests tend to be blocked, you might also consider ICMP Timestamp or ICMP Address Mask requests to tell if a system is online. Nmap uses timestamp request (ICMP Type 13) and checks whether it will get a Timestamp reply (ICMP Type 14). Adding the `-PP` option tells Nmap to use ICMP timestamp requests. As shown in the figure below, you expect live hosts to reply.



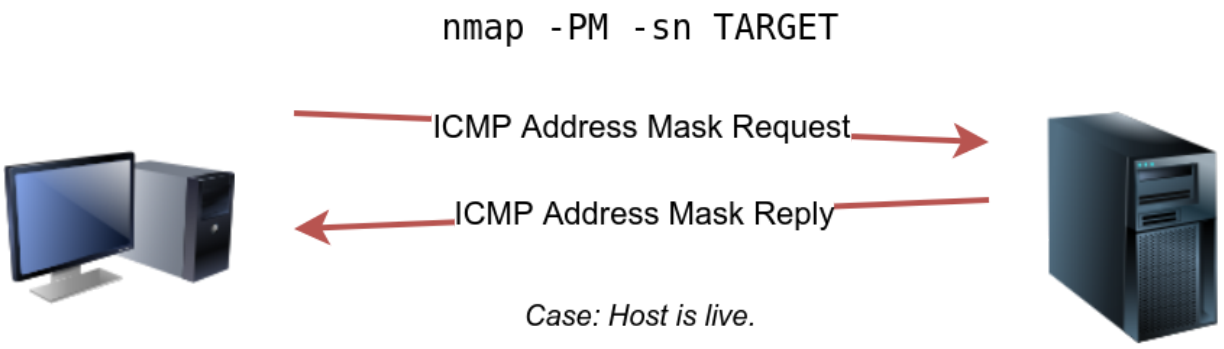
In the following example, we run `nmap -PP -sn MACHINE_IP/24` to discover the online computers on the target machine subnet.

```
pentester@TryHackMe$ sudo nmap -PP -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 12:06 EEST
Nmap scan report for 10.10.68.50
Host is up (0.13s latency).
Nmap scan report for 10.10.68.52
Host is up (0.25s latency).
Nmap scan report for 10.10.68.77
Host is up (0.14s latency).
Nmap scan report for 10.10.68.110
Host is up (0.14s latency).
Nmap scan report for 10.10.68.140
Host is up (0.15s latency).
Nmap scan report for 10.10.68.209
Host is up (0.14s latency).
Nmap scan report for 10.10.68.220
Host is up (0.14s latency).
Nmap scan report for 10.10.68.222
Host is up (0.14s latency).
Nmap done: 256 IP addresses (8 hosts up) scanned in 10.93 seconds
```

Similar to the previous ICMP scan, this scan will send many ICMP timestamp requests to every valid IP address in the target subnet. In the Wireshark screenshot below, you can see one source IP address sending ICMP packets to every possible IP address to discover online hosts.



Similarly, Nmap uses address mask queries (ICMP Type 17) and checks whether it gets an address mask reply (ICMP Type 18). This scan can be enabled with the option `-PM`. As shown in the figure below, live hosts are expected to reply to ICMP address mask requests.

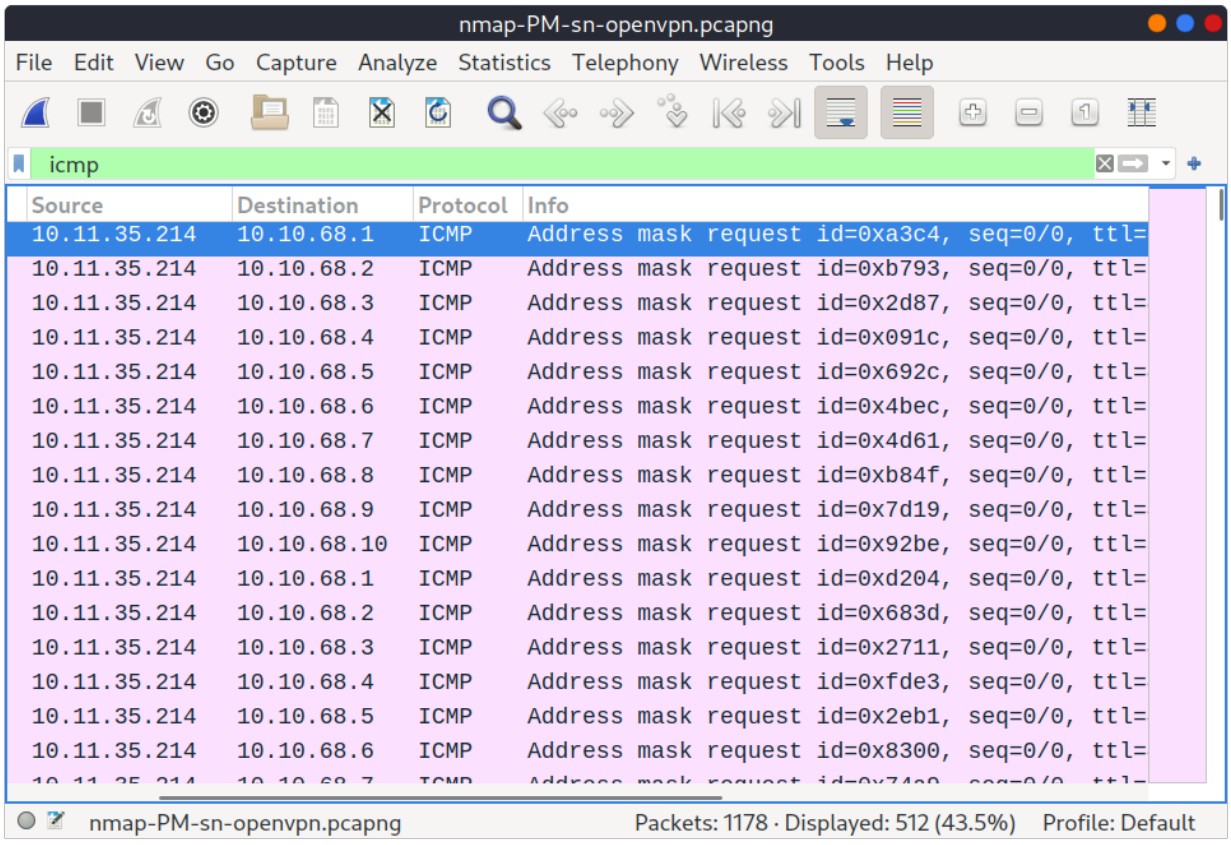


In an attempt to discover live hosts using ICMP address mask queries, we run the command `nmap -PM -sn MACHINE_IP/24`. Although, based on earlier scans, we know that at least eight hosts are up, this scan returned none. The reason is that the target system

or a firewall on the route is blocking this type of ICMP packet. Therefore, it is essential to learn multiple approaches to achieve the same result. If one type of packet is being blocked, we can always choose another to discover the target network and services.

```
pentester@TryHackMe$ sudo nmap -PM -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 12:13 EEST
Nmap done: 256 IP addresses (0 hosts up) scanned in 52.17 seconds
```

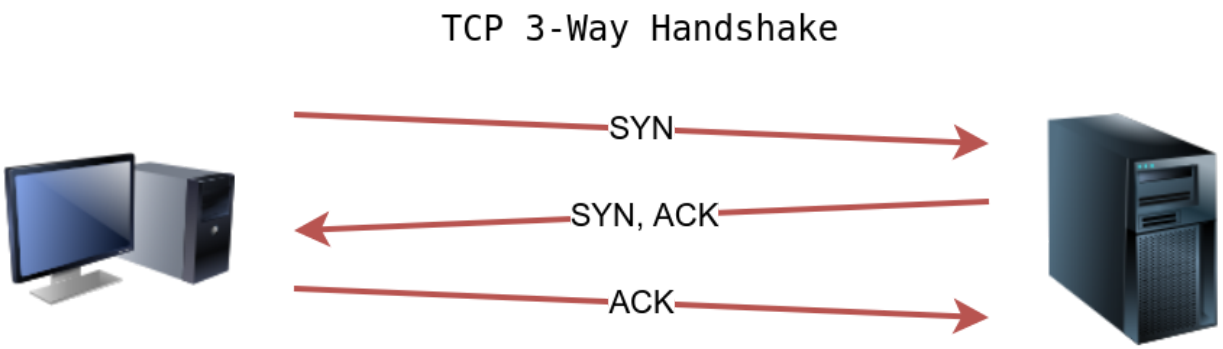
Although we didn't get any reply and could not figure out which hosts are online, it is essential to note that this scan sent ICMP address mask requests to every valid IP address and waited for a reply. Each ICMP request was sent twice, as we can see in the screenshot below.



Nmap Host Discovery Using TCP and UDP:

TCP SYN Ping

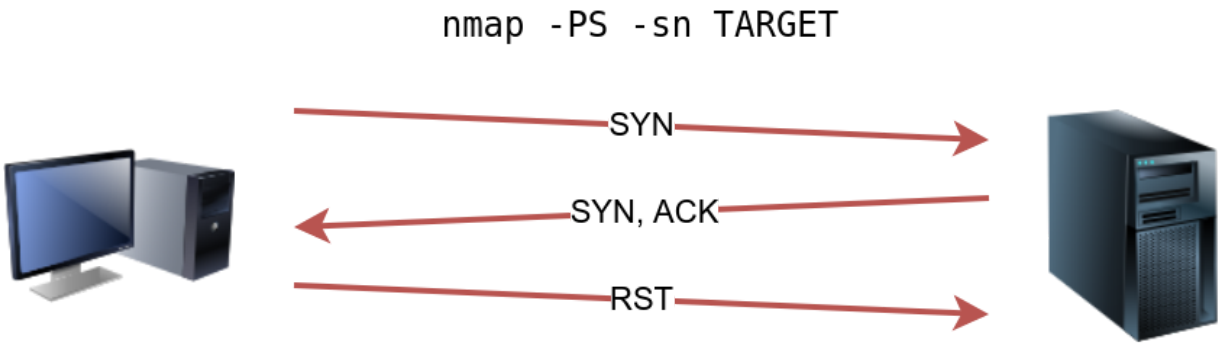
We can send a packet with the SYN (Synchronize) flag set to a TCP port, 80 by default, and wait for a response. An open port should reply with a SYN/ACK (Acknowledge); a closed port would result in an RST (Reset). In this case, we only check whether we will get any response to infer whether the host is up. The specific state of the port is not significant here. The figure below is a reminder of how a TCP 3-way handshake usually works.



Case: TCP port is open.

If you want Nmap to use TCP SYN ping, you can do so via the option `-PS` followed by the port number, range, list, or a combination of them. For example, `-PS21` will target port 21, while `-PS21-25` will target ports 21, 22, 23, 24, and 25. Finally `-PS80,443,8080` will target the three ports 80, 443, and 8080.

Privileged users (root and sudoers) can send TCP SYN packets and don't need to complete the TCP 3-way handshake even if the port is open, as shown in the figure below. Unprivileged users have no choice but to complete the 3-way handshake if the port is open.

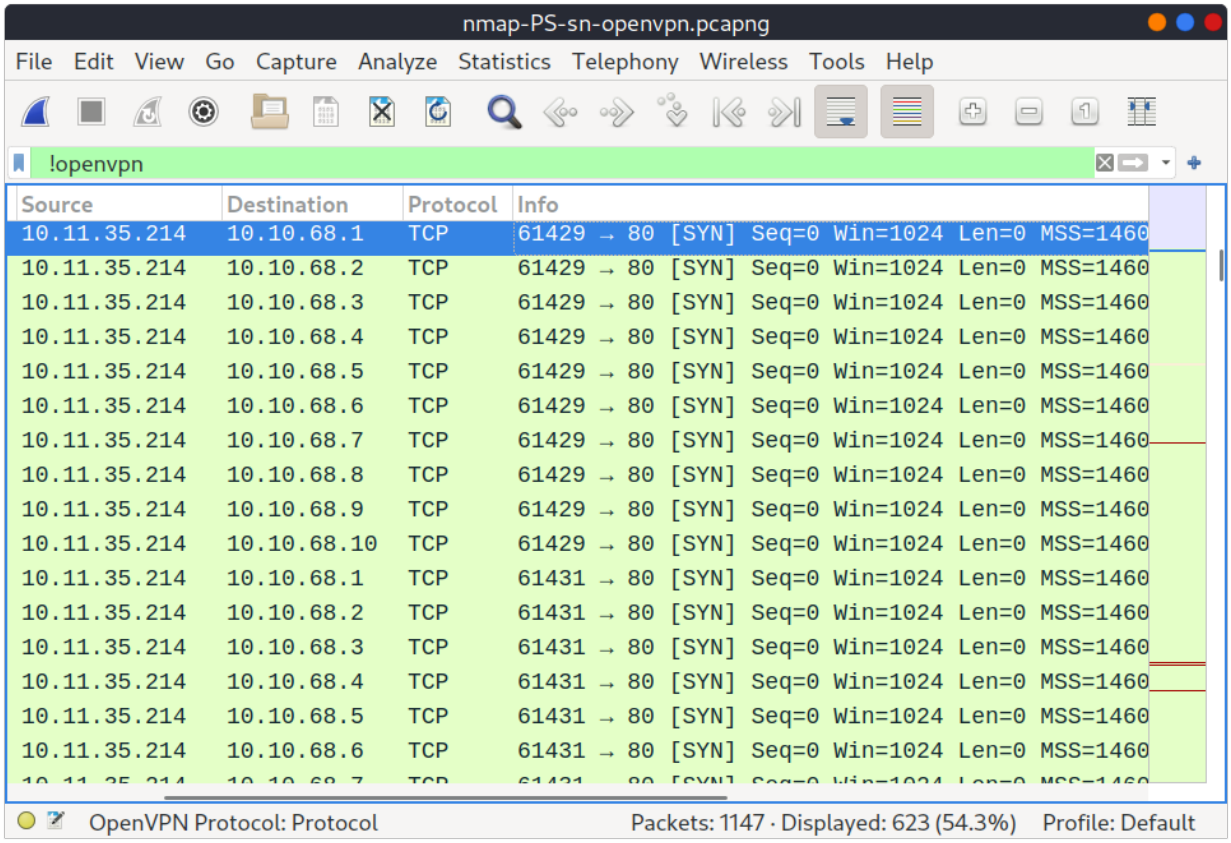


Case: TCP port is open.

We will run `nmap -PS -sn MACHINE_IP/24` to scan the target VM subnet. As we can see in the output below, we were able to discover five hosts.

```
pentester@TryHackMe$ sudo nmap -PS -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 13:45 EEST
Nmap scan report for 10.10.68.52
Host is up (0.10s latency).
Nmap scan report for 10.10.68.121
Host is up (0.16s latency).
Nmap scan report for 10.10.68.125
Host is up (0.089s latency).
Nmap scan report for 10.10.68.134
Host is up (0.13s latency).
Nmap scan report for 10.10.68.220
Host is up (0.11s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 17.38 seconds
```

Let’s take a closer look at what happened behind the scenes by looking at the network traffic on Wireshark in the figure below. Technically speaking, since we didn’t specify any TCP ports to use in the TCP ping scan, Nmap used common ports; in this case, it is TCP port 80. Any service listening on port 80 is expected to reply, indirectly indicating that the host is online.

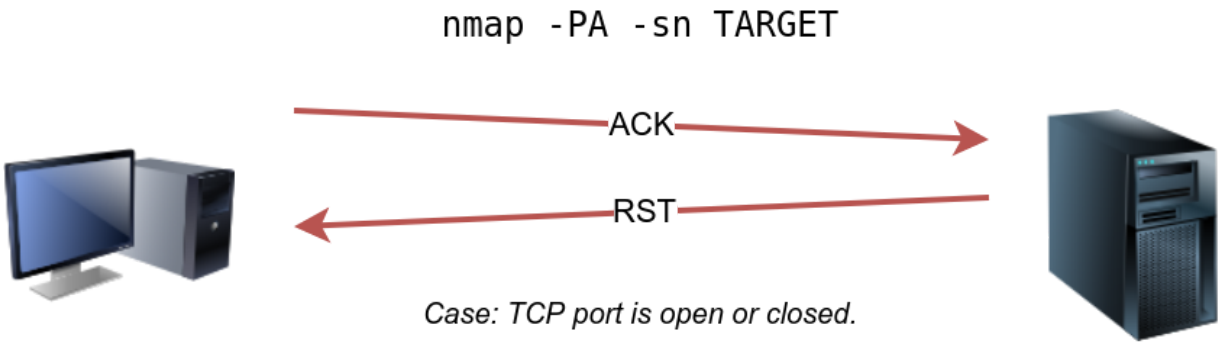


TCP ACK Ping

As you have guessed, this sends a packet with an ACK flag set. You must be running Nmap as a privileged user to be able to accomplish this. If you try it as an unprivileged user, Nmap will attempt a 3-way handshake.

By default, port 80 is used. The syntax is similar to TCP SYN ping. `-PA` should be followed by a port number, range, list, or a combination of them. For example, consider `-PA21`, `-PA21-25` and `-PA80,443,8080`. If no port is specified, port 80 will be used.

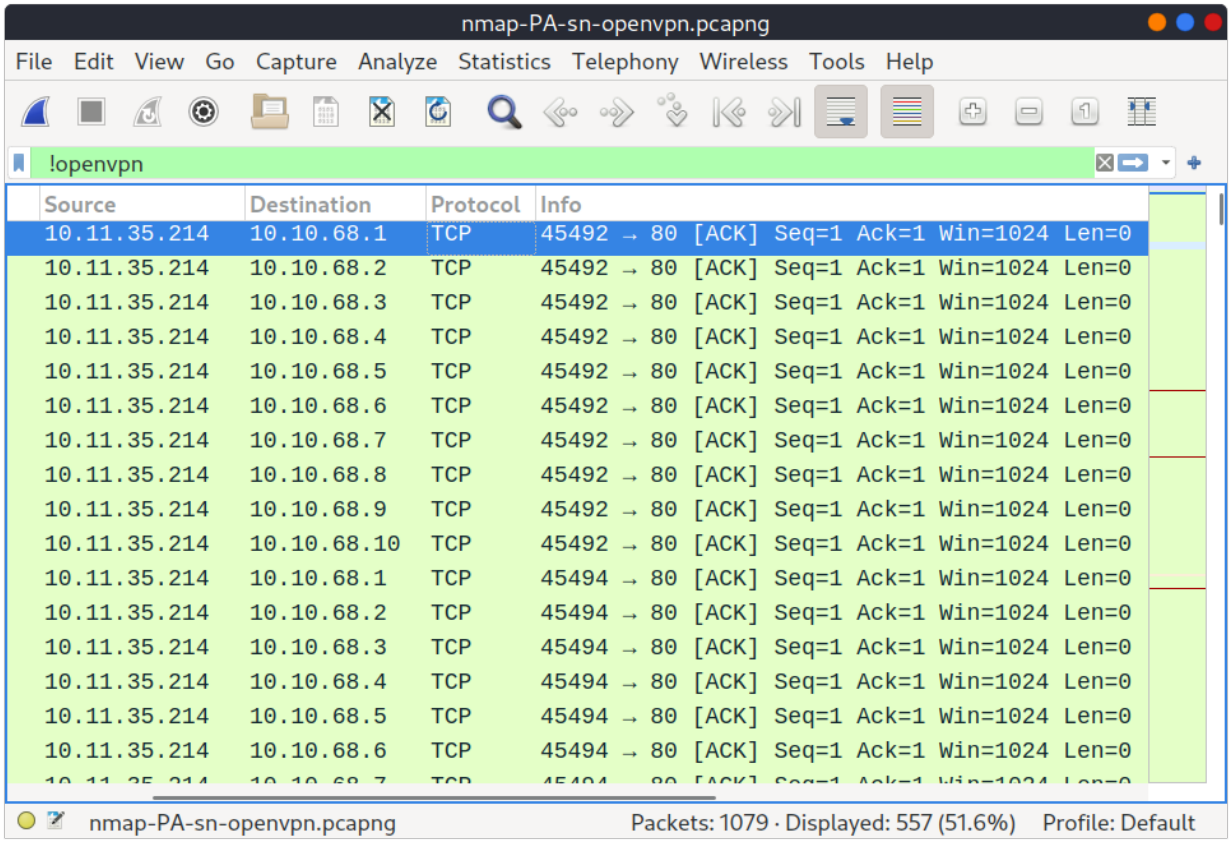
The following figure shows that any TCP packet with an ACK flag should get a TCP packet back with an RST flag set. The target responds with the RST flag set because the TCP packet with the ACK flag is not part of any ongoing connection. The expected response is used to detect if the target host is up.



In this example, we run `sudo nmap -PA -sn MACHINE_IP/24` to discover the online hosts on the target's subnet. We can see that the TCP ACK ping scan detected five hosts as up.

```
pentester@TryHackMe$ sudo nmap -PA -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 13:46 EEST
Nmap scan report for 10.10.68.52
Host is up (0.11s latency).
Nmap scan report for 10.10.68.121
Host is up (0.12s latency).
Nmap scan report for 10.10.68.125
Host is up (0.10s latency).
Nmap scan report for 10.10.68.134
Host is up (0.10s latency).
Nmap scan report for 10.10.68.220
Host is up (0.10s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 29.89 seconds
```

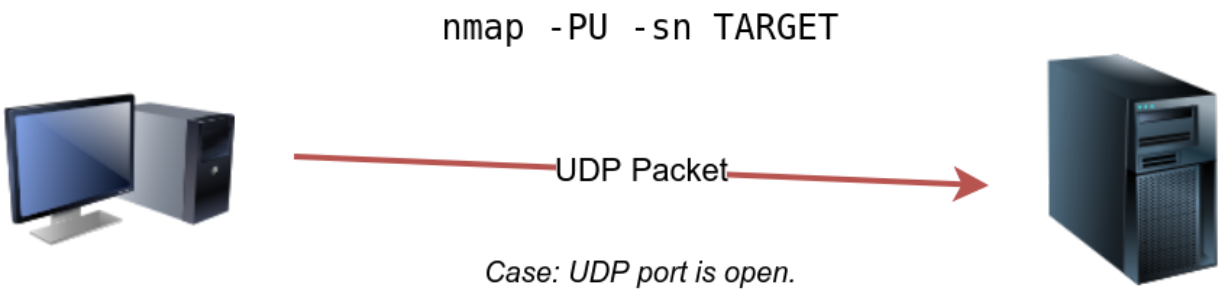
If we peek at the network traffic as shown in the figure below, we will discover many packets with the ACK flag set and sent to port 80 of the target systems. Nmap sends each packet twice. The systems that don't respond are offline or inaccessible.

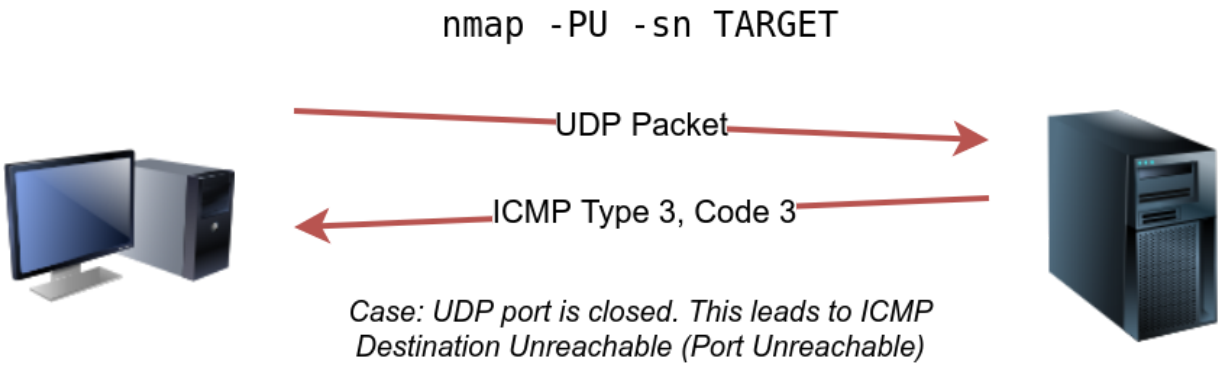


UDP Ping

Finally, we can use UDP to discover if the host is online. Contrary to TCP SYN ping, sending a UDP packet to an open port is not expected to lead to any reply. However, if we send a UDP packet to a closed UDP port, we expect to get an ICMP port unreachable packet; this indicates that the target system is up and available.

In the following figure, we see a UDP packet sent to an open UDP port and not triggering any response. However, sending a UDP packet to any closed UDP port can trigger a response indirectly indicating that the target is online.

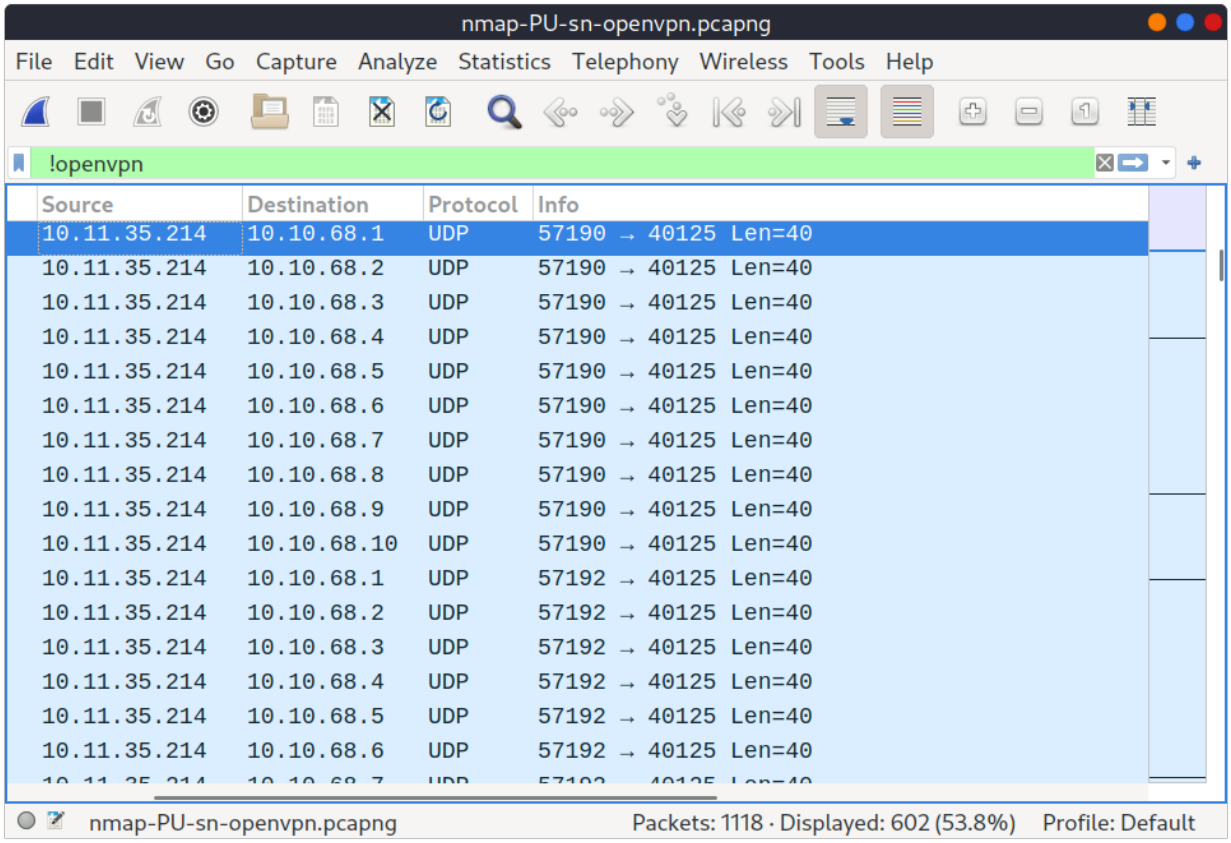




The syntax to specify the ports is similar to that of TCP SYN ping and TCP ACK ping; Nmap uses `-PU` for UDP ping. In the following example, we use a UDP scan, and we discover five live hosts.

```
pentester@TryHackMe$ sudo nmap -PU -sn 10.10.68.220/24
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-02 13:45 EEST
Nmap scan report for 10.10.68.52
Host is up (0.10s latency).
Nmap scan report for 10.10.68.121
Host is up (0.10s latency).
Nmap scan report for 10.10.68.125
Host is up (0.14s latency).
Nmap scan report for 10.10.68.134
Host is up (0.096s latency).
Nmap scan report for 10.10.68.220
Host is up (0.11s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 9.20 seconds
```

Let’s inspect the UDP packets generated. In the following Wireshark screenshot, we notice Nmap sending UDP packets to UDP ports that are most likely closed. The image below shows that Nmap uses an uncommon UDP port to trigger an ICMP destination unreachable (port unreachable) error.



Masscan

On a side note, Masscan uses a similar approach to discover the available systems. However, to finish its network scan quickly, Masscan is quite aggressive with the rate of packets it generates. The syntax is quite similar: `-p` can be followed by a port number, list, or range. Consider the following examples:

- `masscan MACHINE_IP/24 -p443`
- `masscan MACHINE_IP/24 -p80,443`
- `masscan MACHINE_IP/24 -p22-25`
- `masscan MACHINE_IP/24 --top-ports 100`

Masscan can be installed using `apt install masscan`

Reverse DNS Lookup:

Nmap's default behaviour is to use reverse-DNS online hosts. Because the hostnames can reveal a lot, this can be a helpful step. However, if you don't want to send such DNS

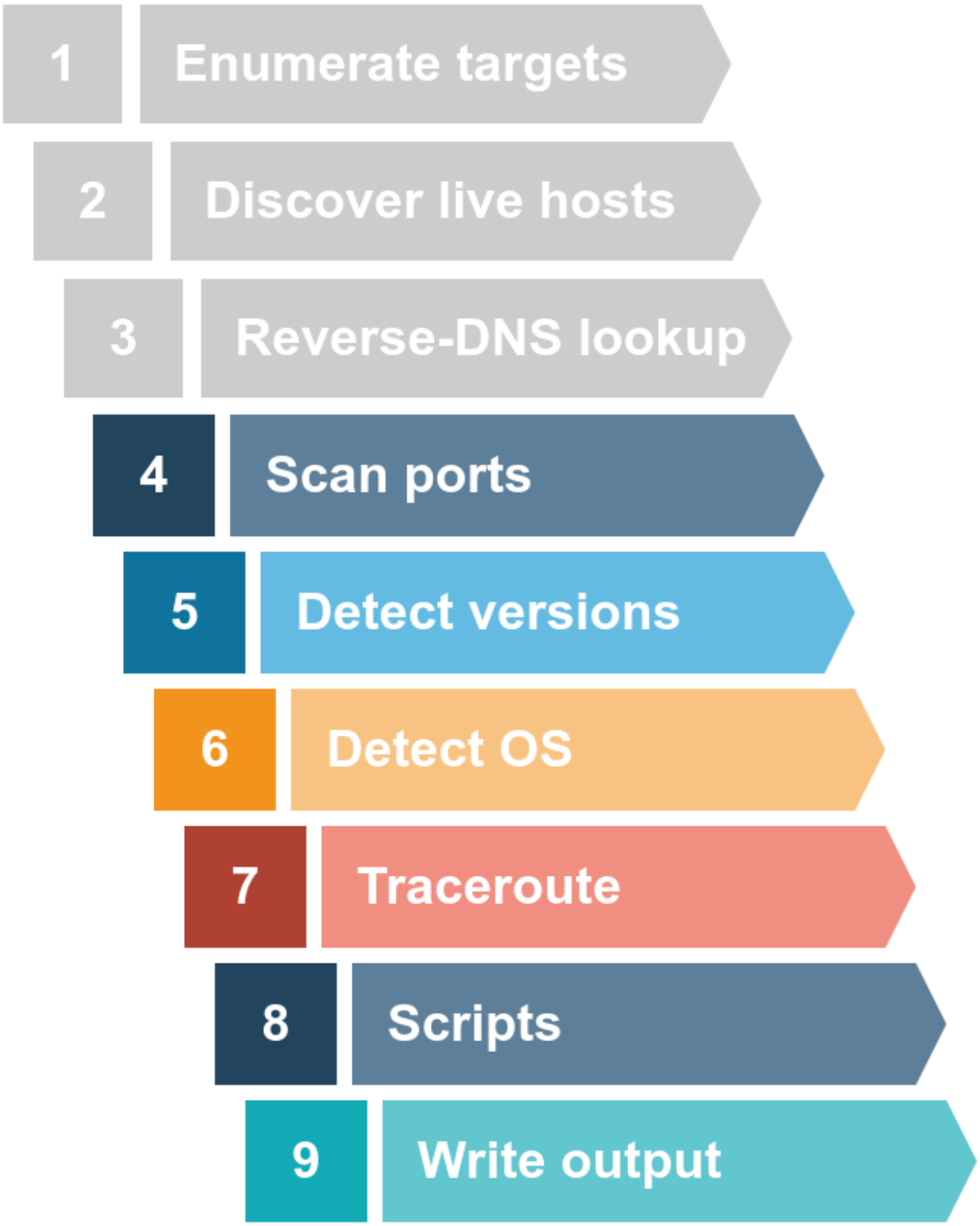
queries, you use `-n` to skip this step.

By default, Nmap will look up online hosts; however, you can use the option `-R` to query the DNS server even for offline hosts. If you want to use a specific DNS server, you can add the `--dns-servers DNS_SERVER` option.

Nmap Basic Port Scan

So far, we have covered three steps of a Nmap scan:

1. Enumerate targets
2. Discover live hosts
3. Reverse-DNS lookup



The next step would be checking which ports are open and listening and which ports are closed. Therefore, we focus on port scanning and the different types of port scans

used by `nmap`. This explains:

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

TCP and UDP Ports:

In the same sense that an IP address specifies a host on a network among many others, a TCP port 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. Examples include 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).

At the risk of oversimplification, 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.

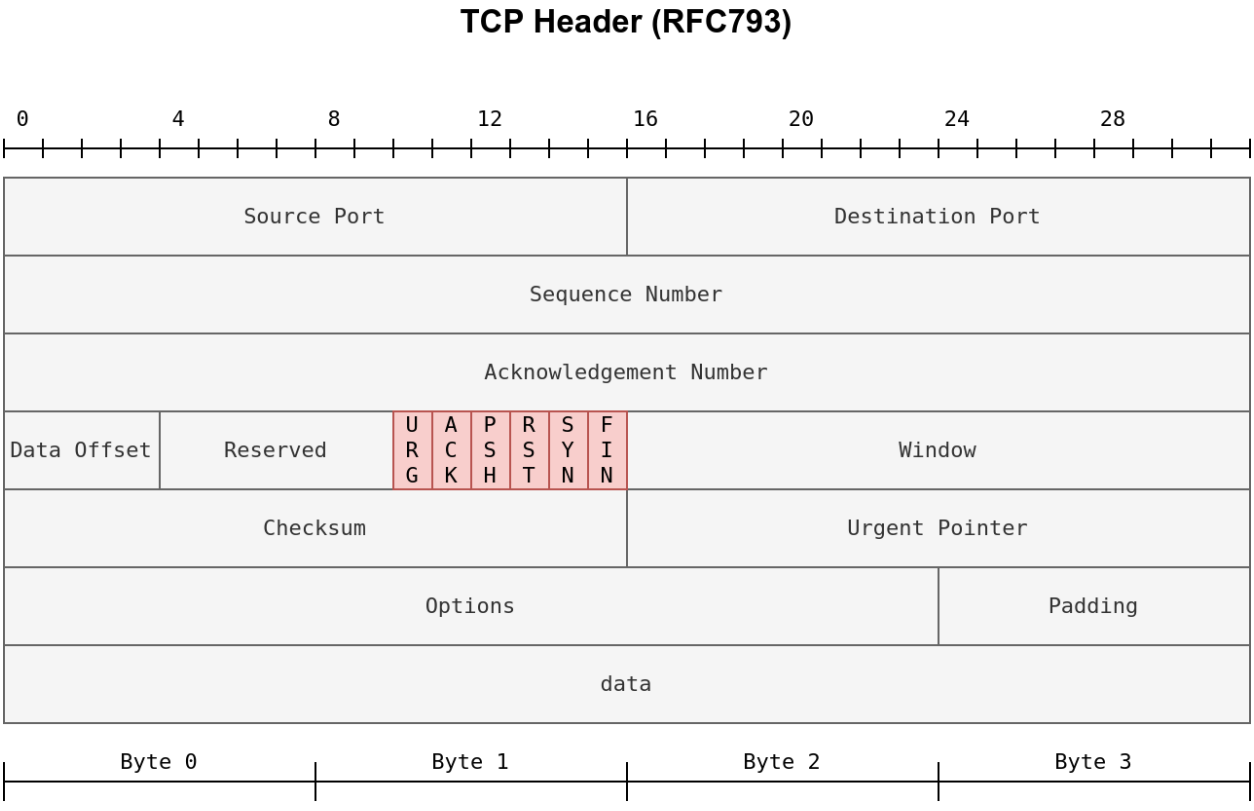
However, in practical situations, we need to consider the impact of firewalls. For instance, a port might be open, but a firewall might be blocking the packets. Therefore, Nmap considers the following six states:

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. **Open|Filtered**: This means that Nmap cannot determine whether the port is open or filtered.
6. **Closed|Filtered**: This means that Nmap cannot decide whether a port is closed or filtered.

TCP Flags:

Nmap 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 24 bytes of a TCP segment. The following figure shows the TCP header. In the first row, we have the source TCP port number and the destination port number. We can see that the port number is allocated 16 bits (2 bytes). In the second and third rows, we have the sequence number and the acknowledgement number. Each row has 32 bits (4 bytes) allocated, with six rows total, making up 24 bytes.

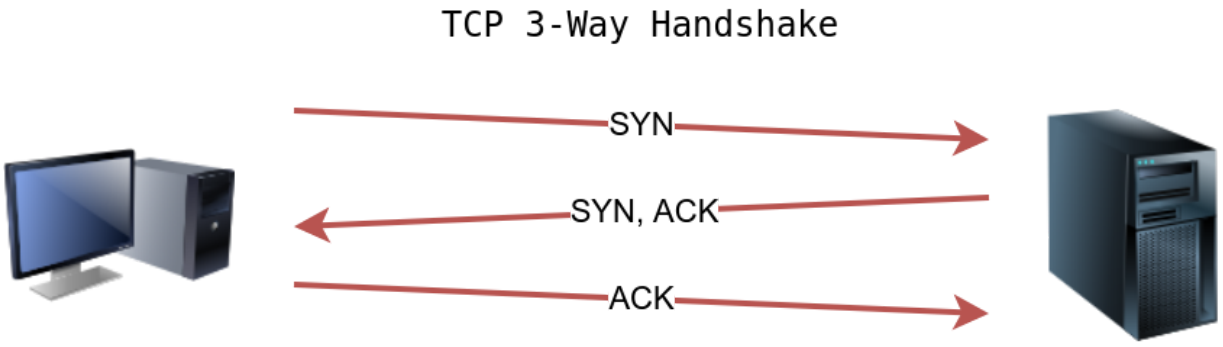


In particular, we need to focus on the flags that Nmap can set or unset. We have highlighted the TCP flags in red. Setting a flag bit means setting its value to 1. From left to right, the TCP header flags are:

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.

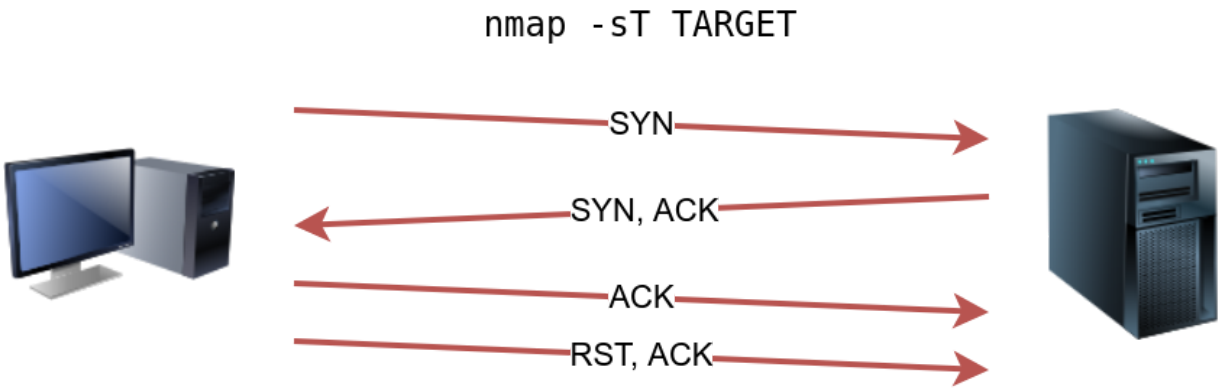
TCP Connect Scan:

TCP 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.



Case: TCP port is open.

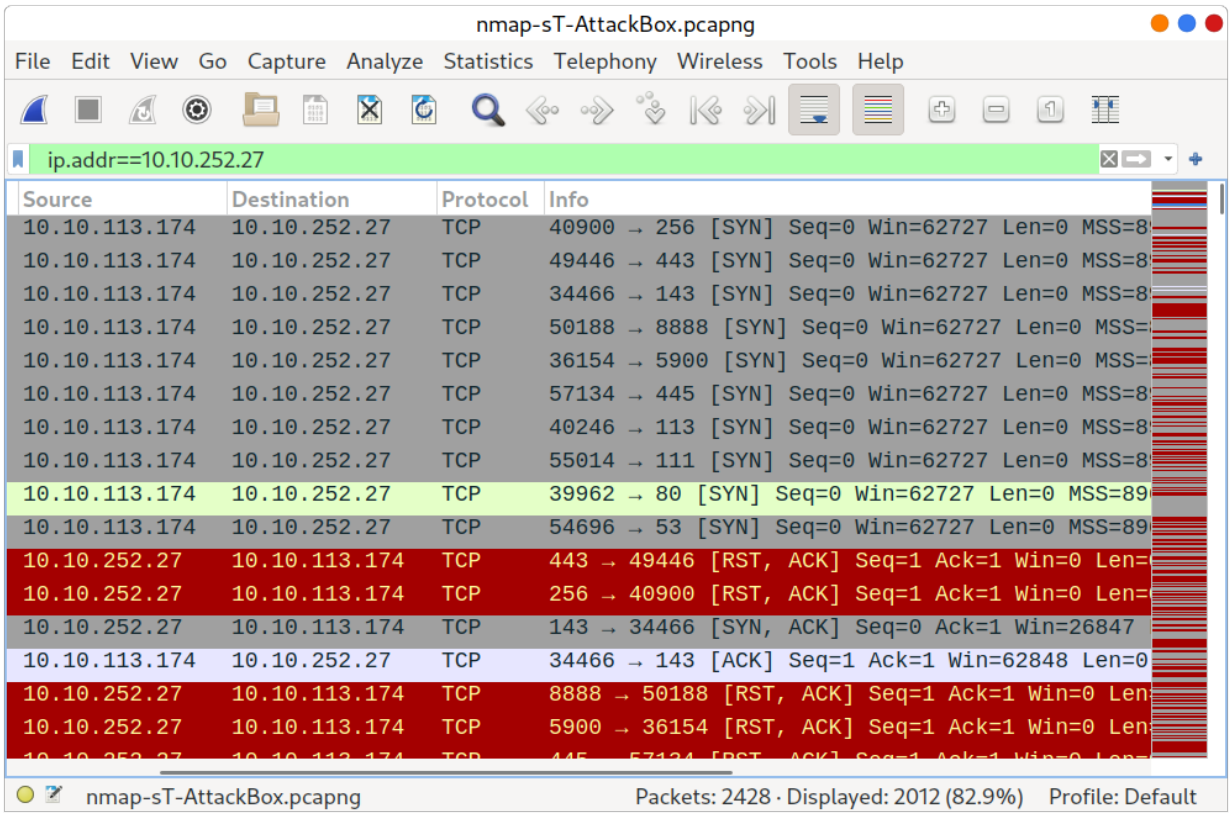
We are interested in learning whether the TCP port is open, not establishing a TCP connection. Hence the connection is torn as soon as its state is confirmed by sending a RST/ACK. You can choose to run TCP connect scan using `-sT`.



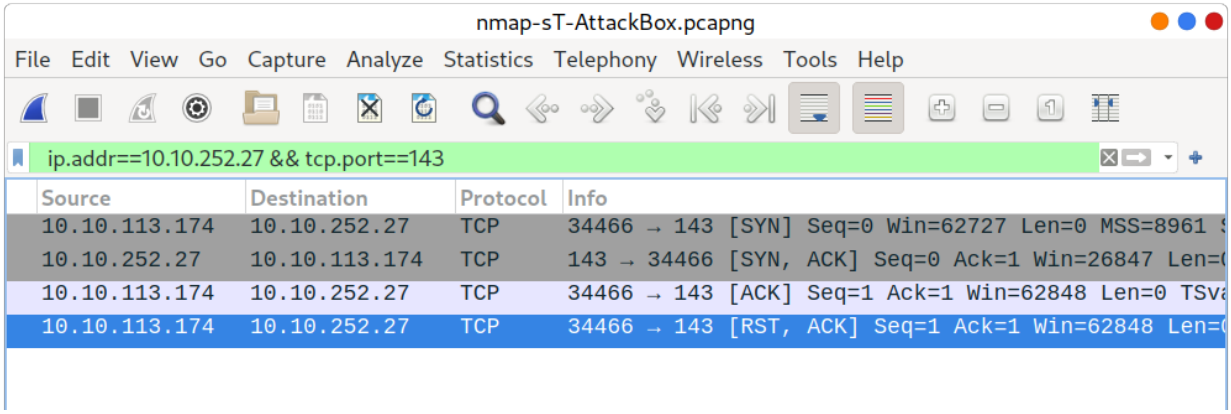
Case: TCP port is open.

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.

In the following 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.



We notice 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 below 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 following command example returned a detailed list of the open ports.

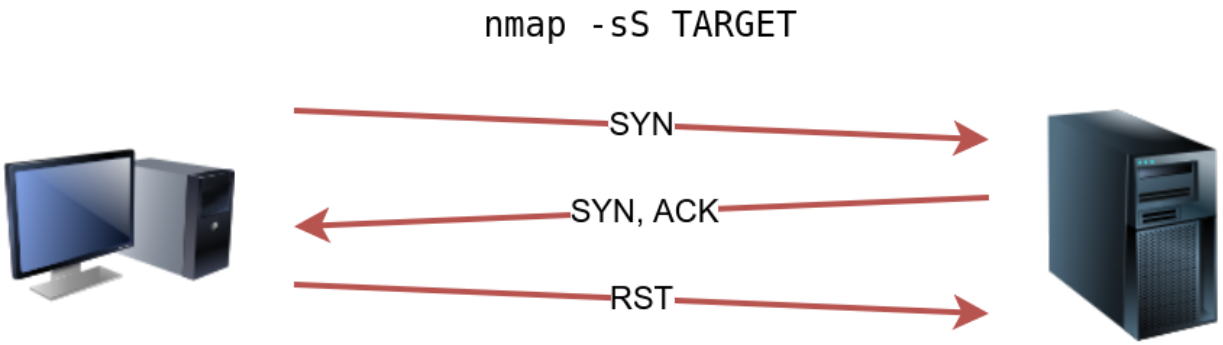
```
pentester@TryHackMe$ nmap -sT MACHINE_IP
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for MACHINE_IP
Host is up (0.0024s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
111/tcp open  rpcbind
143/tcp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

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

Note that we can use `-F` to enable fast mode and decrease the number of scanned ports from 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.

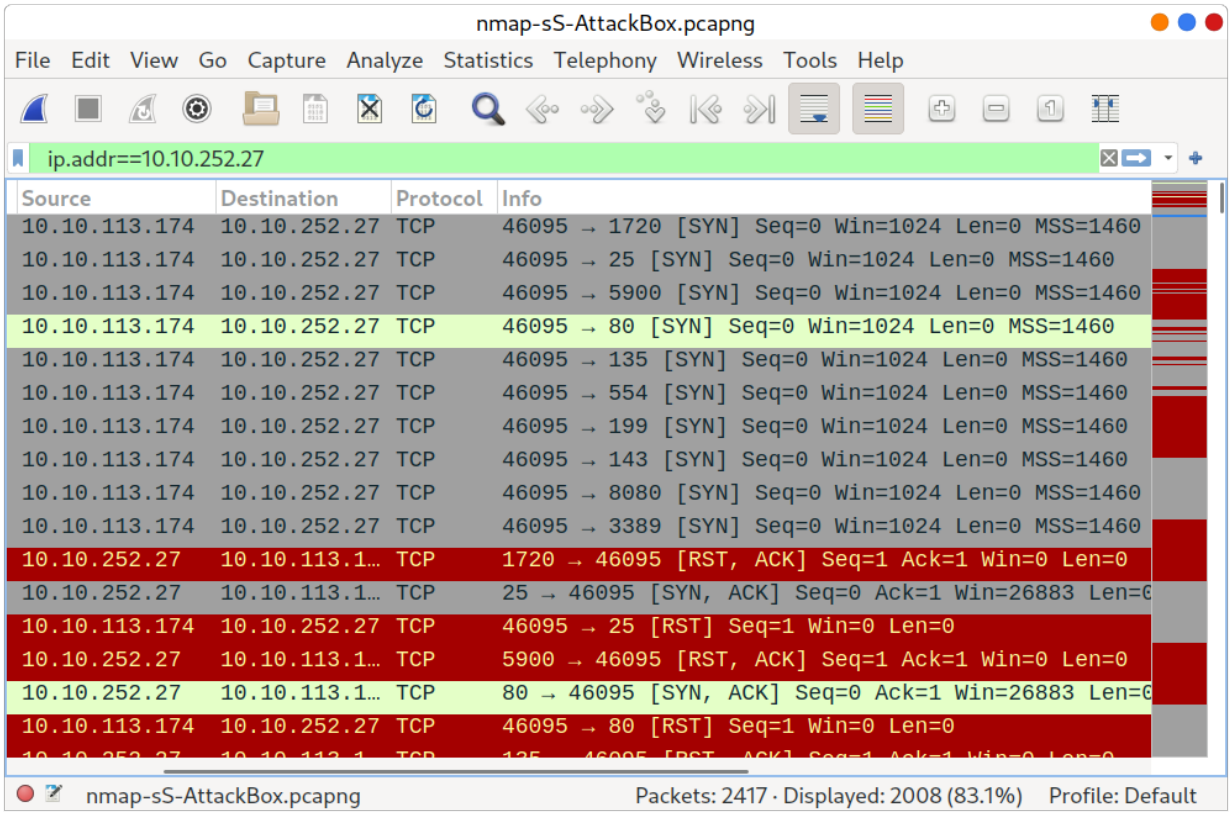
TCP SYN Scan

Unprivileged 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 below shows how the TCP SYN scan works without completing the TCP 3-way handshake.



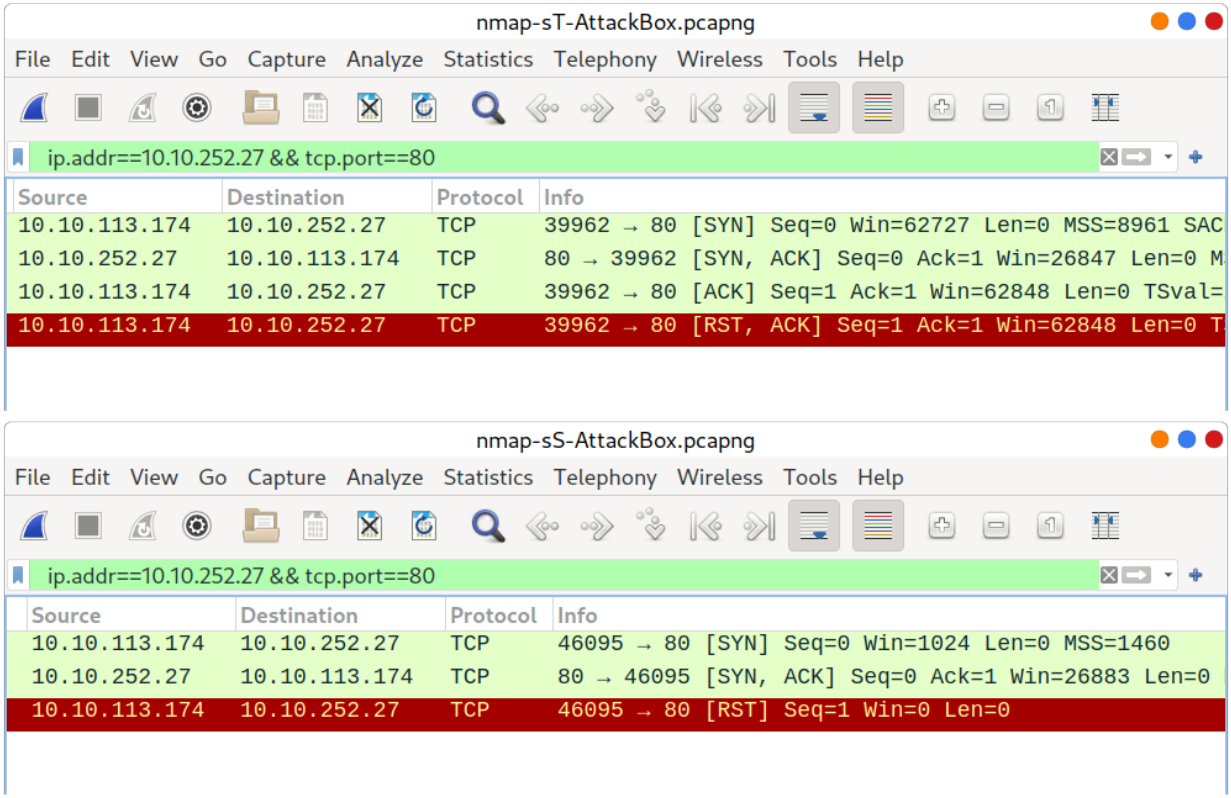
Case: TCP port is open.

The following screenshot from Wireshark shows a TCP SYN scan. 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 following screenshot. 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.

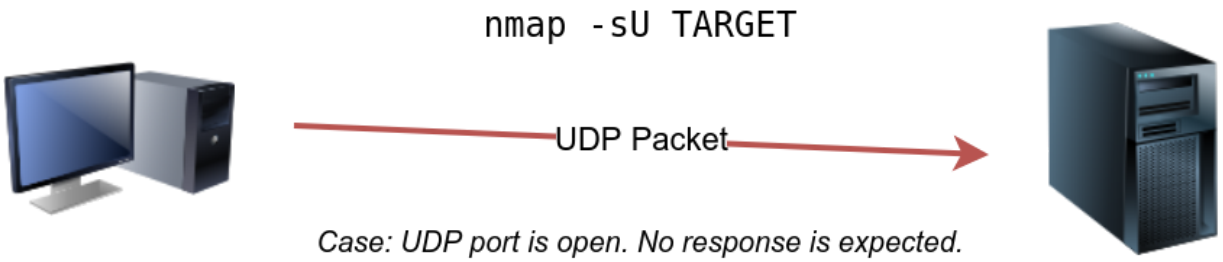
```
pentester@TryHackMe$ sudo nmap -sS MACHINE_IP
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for MACHINE_IP
Host is up (0.0073s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
110/tcp open  pop3
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
```

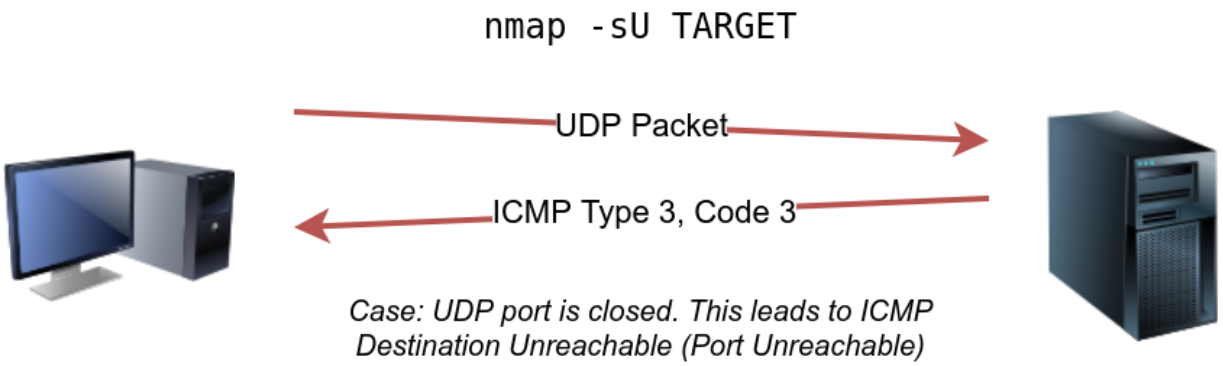
UDP Scan:

UDP 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.

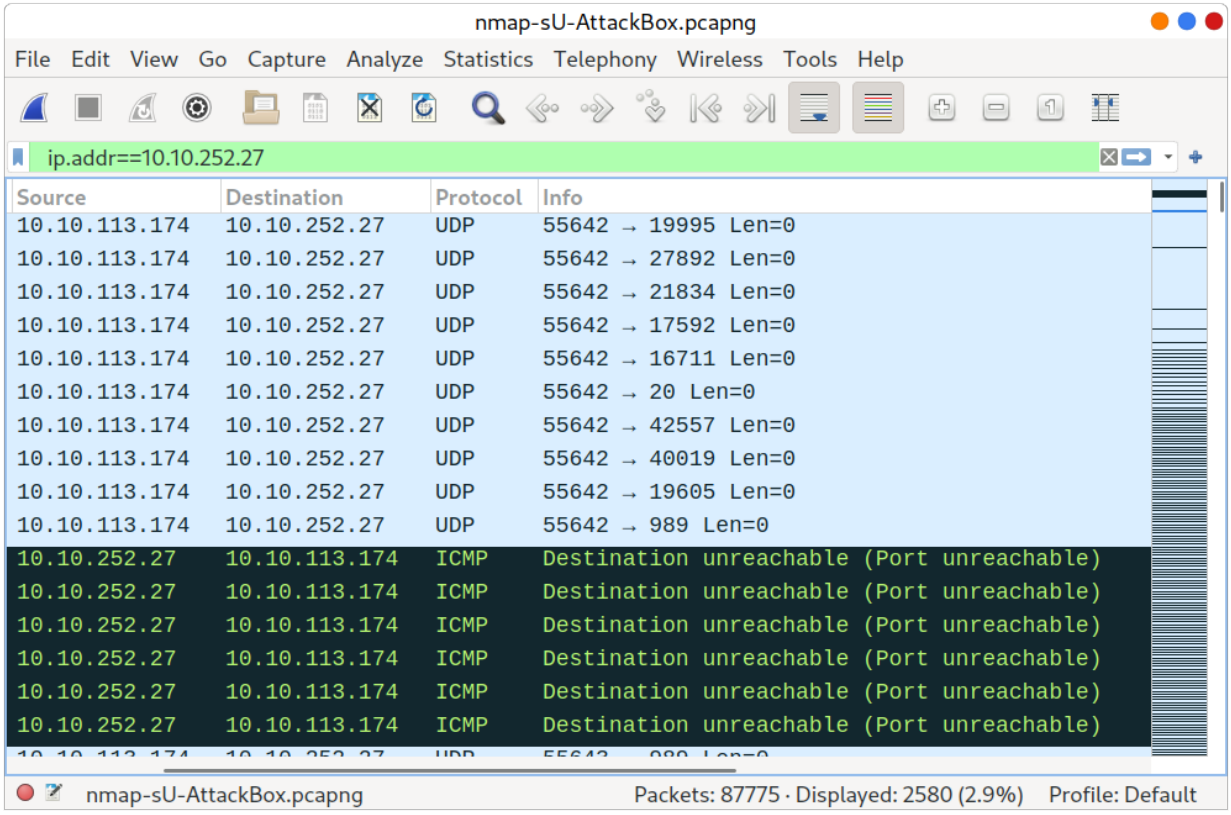
The following figure shows that if we send a UDP packet to an open UDP port, we cannot expect any reply in return. Therefore, sending a UDP packet to an open port won't tell us anything.



However, as shown in the figure below, 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.



In the Wireshark capture below, 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.

```
pentester@TryHackMe$ sudo nmap -sU MACHINE_IP
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:54 BST
Nmap scan report for MACHINE_IP
Host is up (0.00061s latency).
Not shown: 998 closed ports
PORT    STATE         SERVICE
68/udp  open|filtered dhcpc
111/udp open          rpcbind
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1085.05 seconds
```

Fine-Tuning Scope and Performance

You 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:

- port list: `p22,80,443` will scan ports 22, 80 and 443.
- 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 `--top-ports 10` 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:

- paranoid (0)
- sneaky (1)
- polite (2)
- normal (3)
- aggressive (4)
- 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.

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.

Nmap Advanced Port Scan

This room 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. We will cover the following types of port scans:

- Null Scan
- FIN Scan
- Xmas Scan
- Maimon Scan
- ACK Scan
- Window Scan
- Custom Scan

Moreover, we will cover the following:

- Spoofing IP
- Spoofing MAC
- Decoy Scan
- Fragmented Packets
- Idle/Zombie Scan

We will discuss options and techniques to evade firewalls and IDS systems. We also cover options to get more verbose details from Nmap.

TCP Null Scan, FIN Scan, and XMAS Scan:

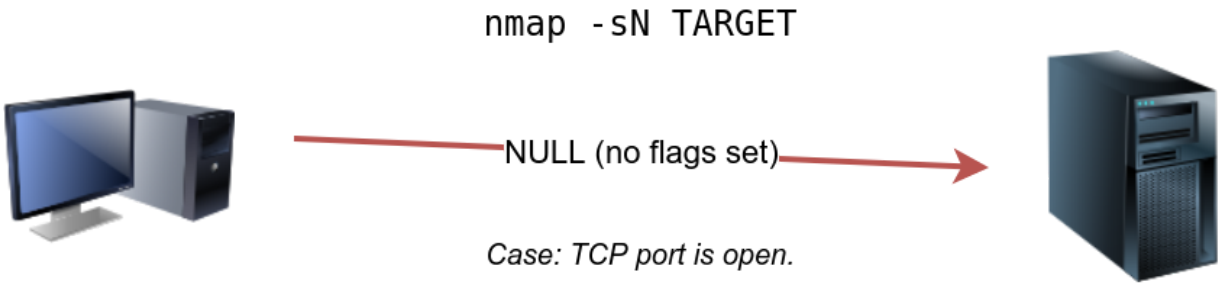
Let’s start with the following three types of scans:

- Null Scan

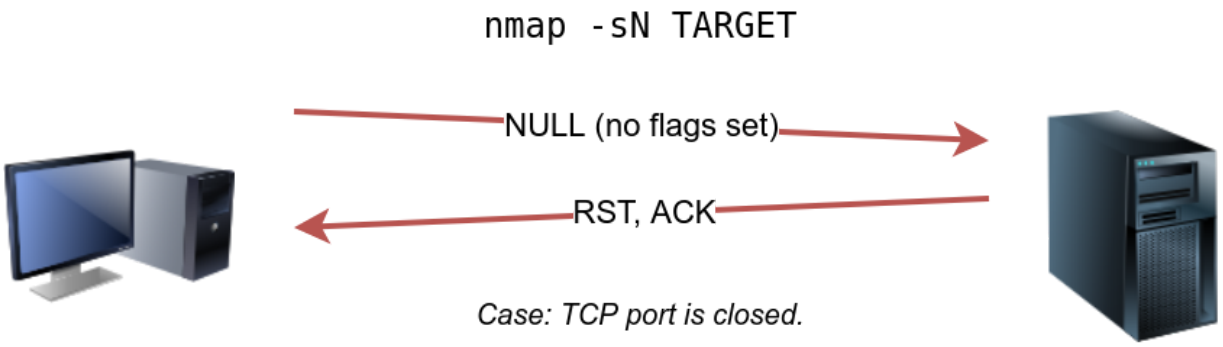
- FIN Scan
- Xmas Scan

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

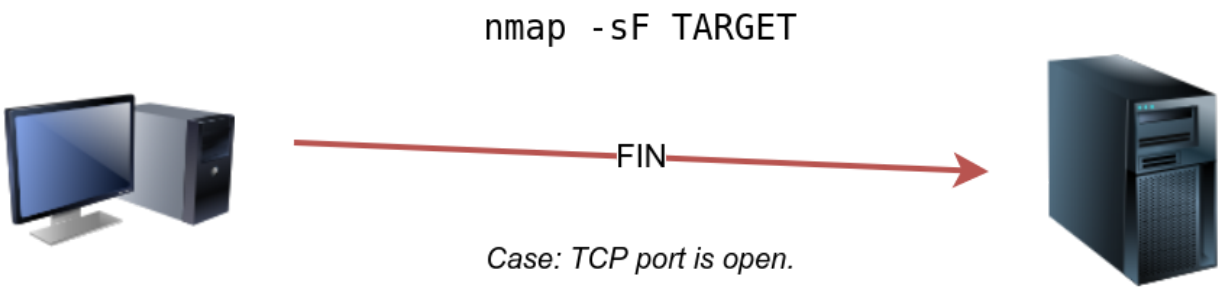
indicate with certainty that these ports are open; there is a possibility that the ports are not responding due to a firewall rule.

```
pentester@TryHackMe$ sudo nmap -sN 10.10.155.141
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:30 BST
Nmap scan report for 10.10.155.141
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
```

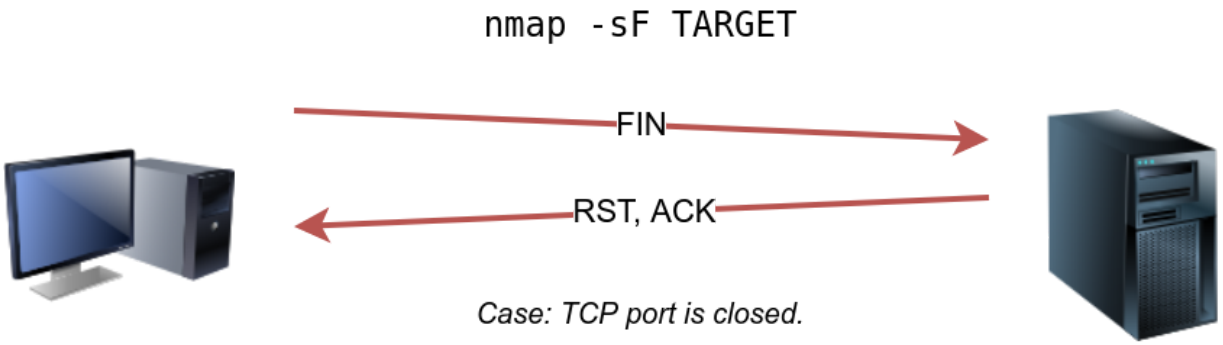
Note that many Nmap options require root privileges. Unless you are running Nmap as root, you need to use `sudo` as in the example above using the `-sN` option.

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.

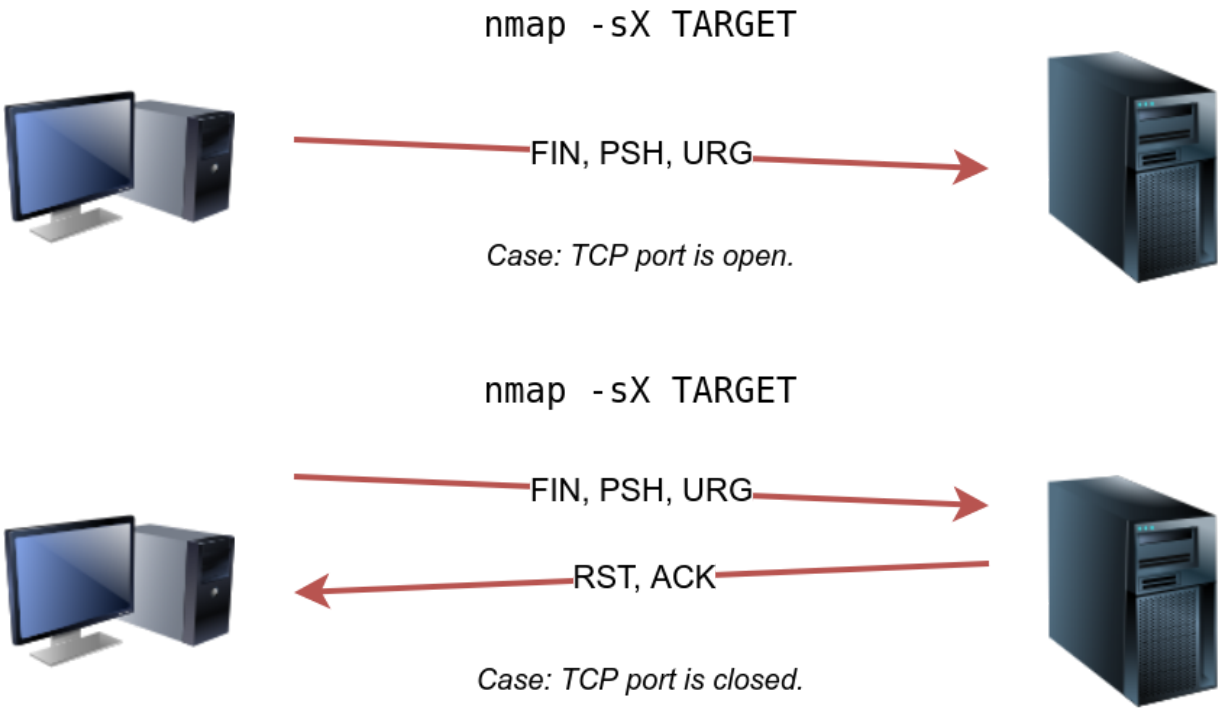
```
pentester@TryHackMe$ sudo nmap -sF 10.10.155.141
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:32 BST
Nmap scan report for 10.10.155.141
Host is up (0.0018s latency).
Not shown: 994 closed ports
PORT    STATE         SERVICE
22/tcp  open|filtered ssh
25/tcp  open|filtered smtp
80/tcp  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@TryHackMe$ sudo nmap -sX 10.10.155.141
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:34 BST
Nmap scan report for 10.10.155.141
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
```

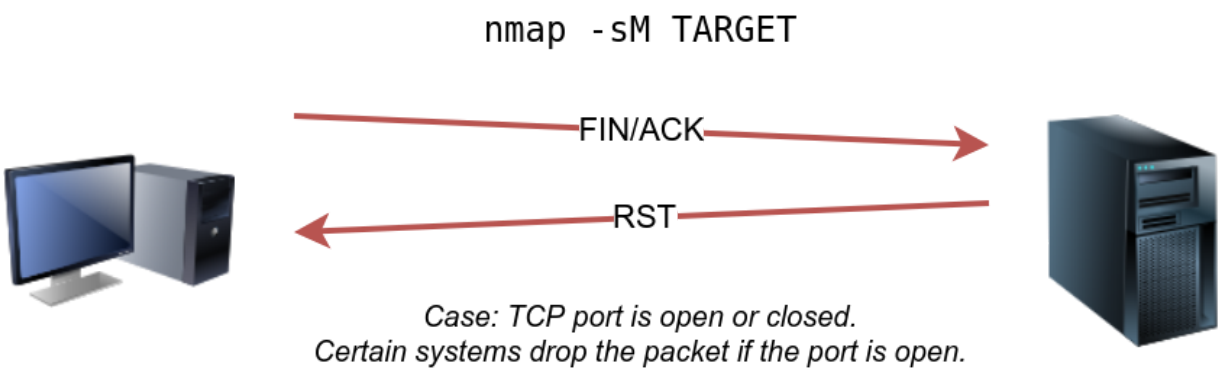
One scenario where these three scan types can be efficient is when scanning a target behind a stateless (non-stateful) firewall. A stateless firewall will check if the incoming packet has the SYN flag set to detect a connection attempt. Using a flag combination that does not match the SYN packet makes 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.

TCP Maimon Scan:

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.

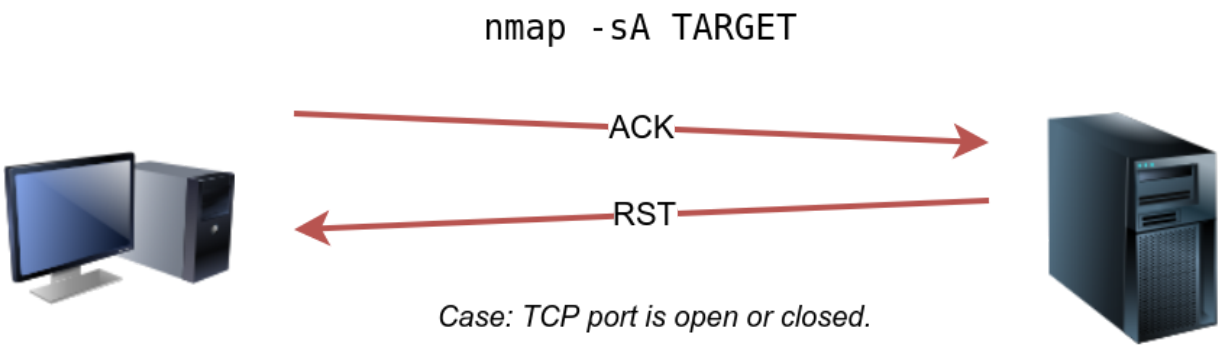
```
pentester@TryHackMe$ sudo nmap -sM 10.10.252.27
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:36 BST
Nmap scan report for ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)
Host is up (0.00095s latency).
All 1000 scanned ports on ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)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

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

This type of scan is not the first scan one would pick to discover a system; however, it is important to know about it as you don't know when it could come in handy.

TCP ACK, Window, and Custom 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@TryHackMe$ sudo nmap -sA 10.10.25.142
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:37 BST
Nmap scan report for 10.10.25.142
Host is up (0.0013s latency).
All 1000 scanned ports on 10.10.25.142 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

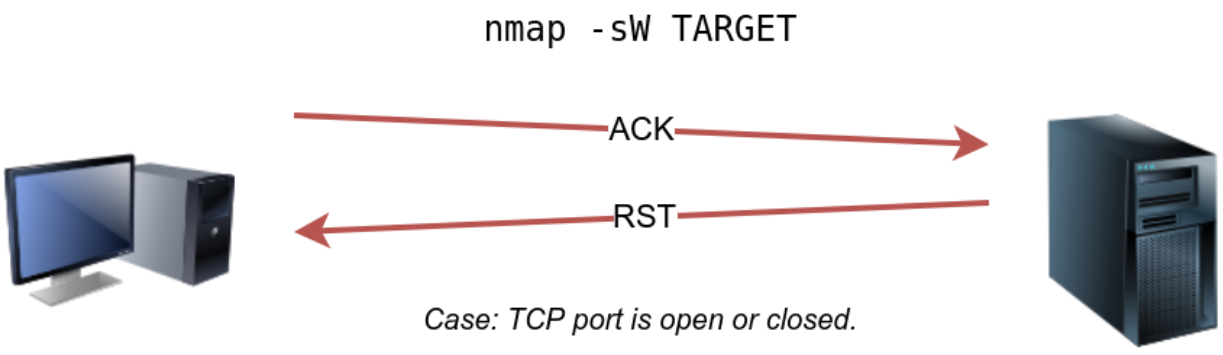
ports were not blocked by the firewall. In other words, this type of scan is more suitable to discover firewall rule sets and configuration.

After setting up the target VM `10.10.25.142` 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@TryHackMe$ sudo nmap -sA 10.10.25.142
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-07 11:34 BST
Nmap scan report for 10.10.25.142
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@TryHackMe$ sudo nmap -sW 10.10.25.142
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:38 BST
Nmap scan report for 10.10.25.142
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 cl
osed
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.

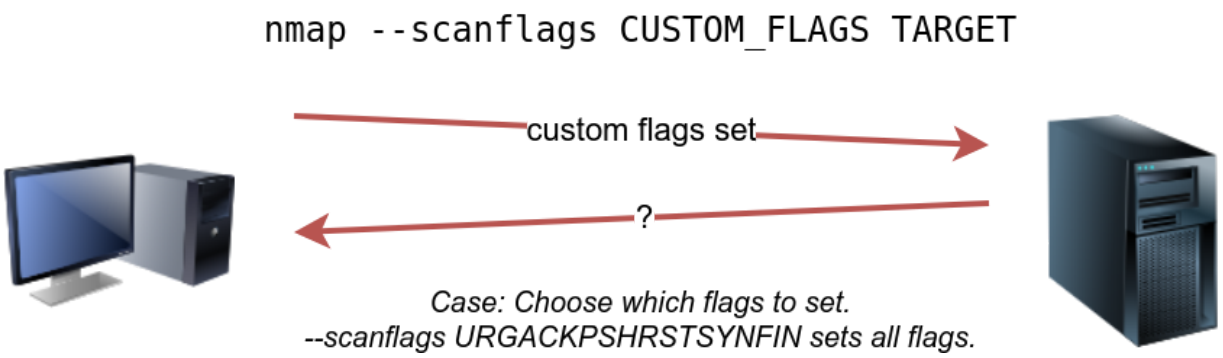
```
pentester@TryHackMe$ sudo nmap -sW 10.10.25.142
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-07 11:39 BST
Nmap scan report for 10.10.25.142
Host is up (0.00040s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE
22/tcp  closed ssh
25/tcp  closed smtp
80/tcp  closed http
MAC Address: 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

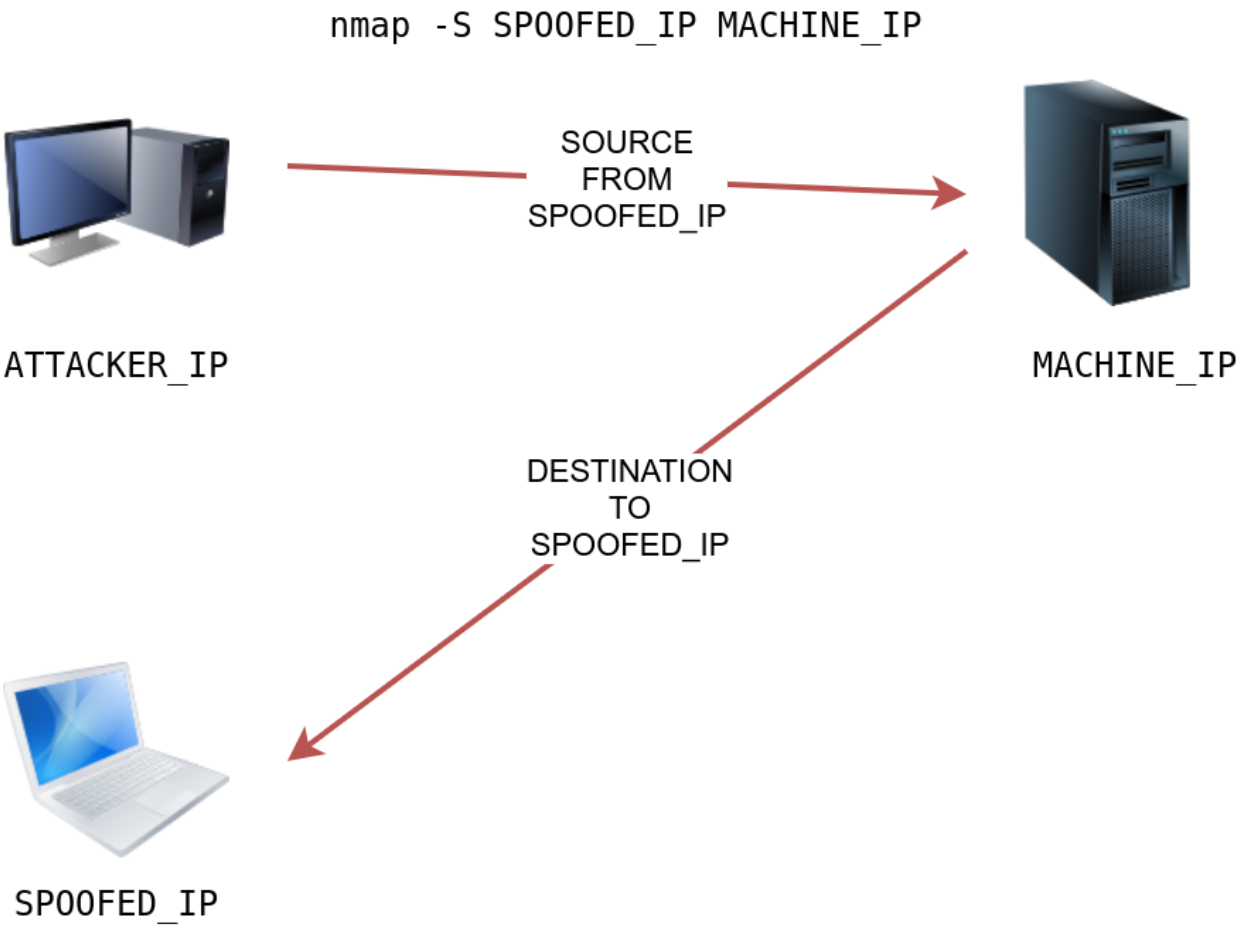


Finally, it is essential to note that the ACK scan and the window scan were very efficient at helping us map out the firewall rules. However, it is vital to remember that just because a firewall is not blocking a specific port, it does not necessarily mean that a service is listening on that port. For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

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 10.10.109.43`. 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.

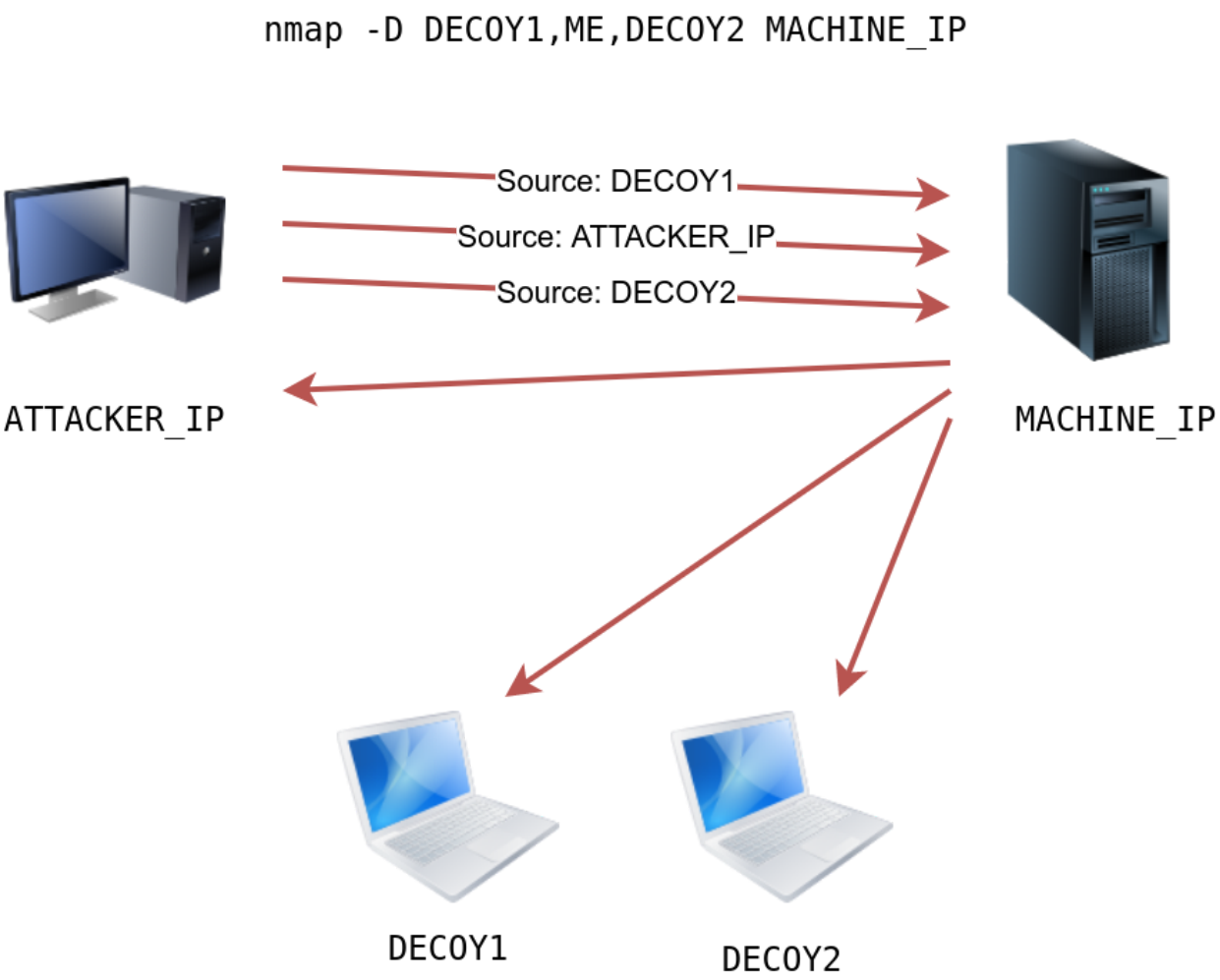
In brief, scanning with a spoofed IP address is three steps:

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 10.10.109.43`, you will need to issue `nmap -e NET_INTERFACE -Pn -S SPOOFED_IP 10.10.109.43` 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 10.10.109.43` will make the scan of 10.10.109.43 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 10.10.109.43`, 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.

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.

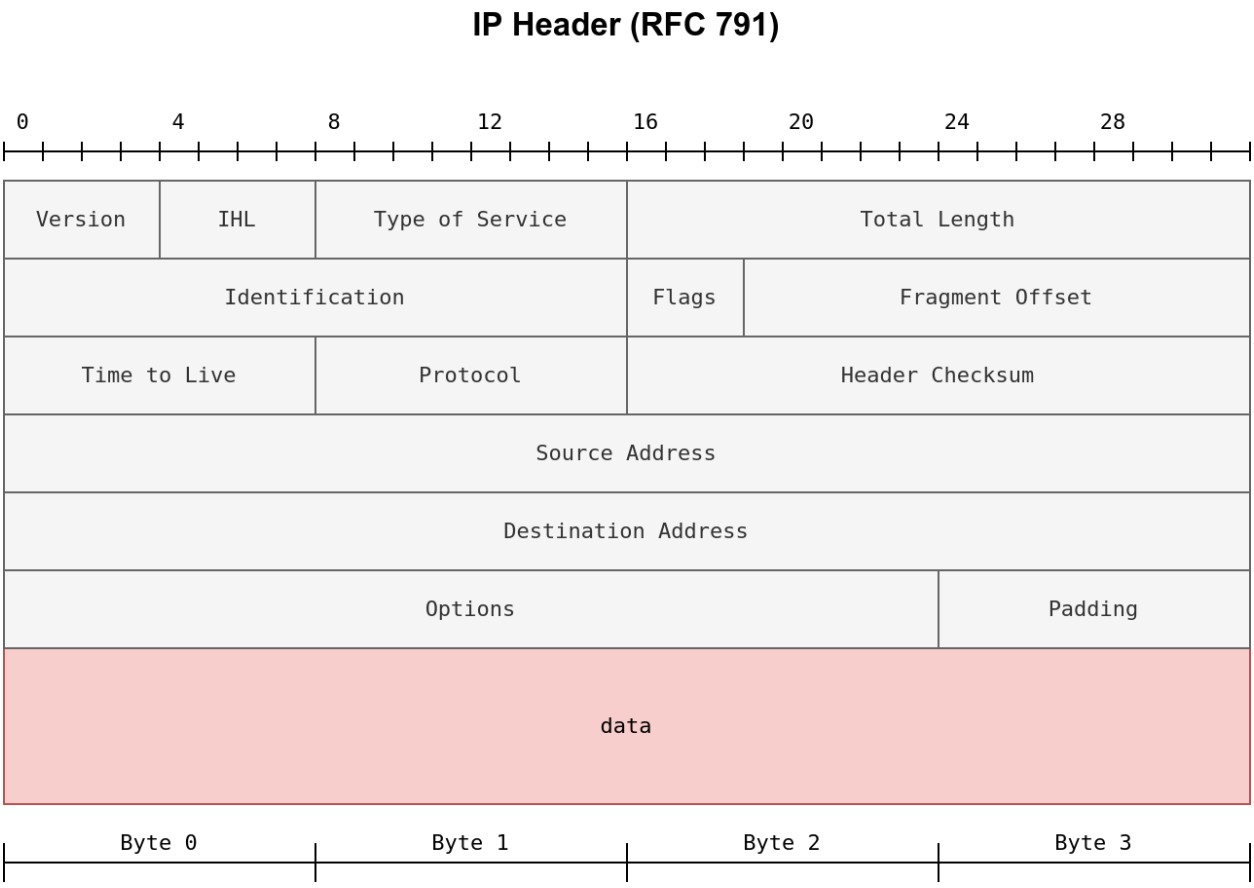
IDS

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.

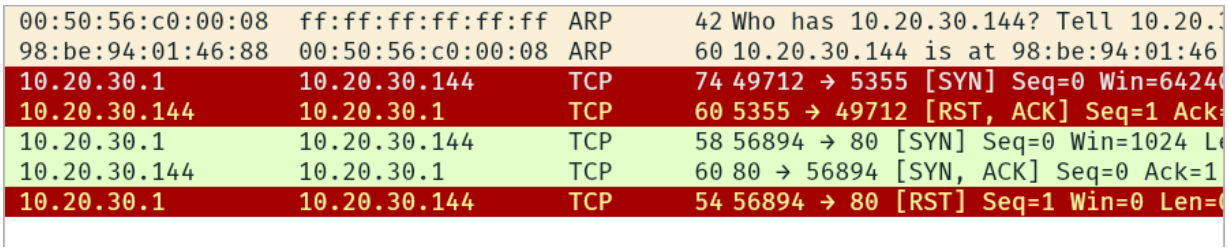
To properly understand fragmentation, we need to look at the IP header in the figure below. It might look complicated at first, but we notice that we know most of its fields. In particular, notice the source address taking 32 bits (4 bytes) on the fourth row, while the destination address is taking another 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.



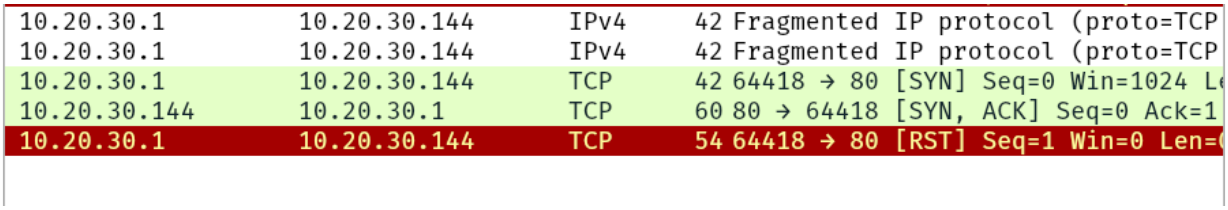
Let's compare running `sudo nmap -sS -p80 10.20.30.144` and `sudo nmap -sS -p80 -f 10.20.30.144`. As you know by now, this will use stealth TCP SYN scan on port 80; however, in the second command, we are requesting Nmap to fragment the IP packets.

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.



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.

Idle/Zombie Scan:

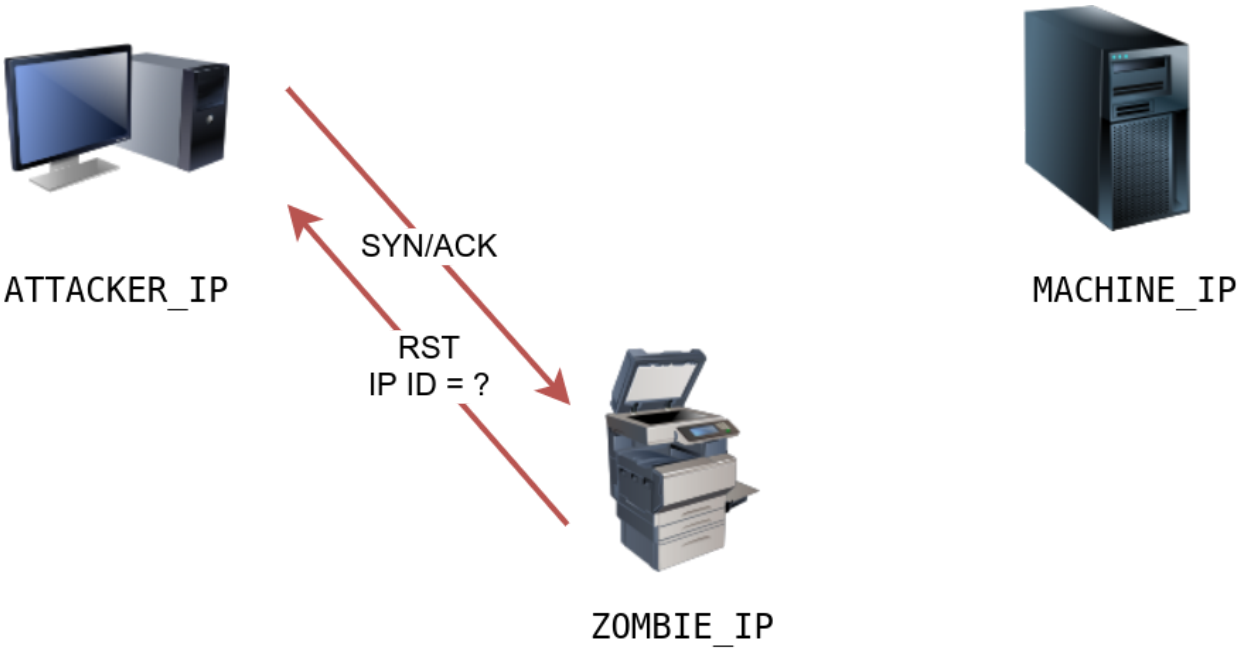
Spoofing the source IP address can be a great approach to scanning stealthily. However, spoofing will only work in specific network setups. It requires you to be in a position where you can monitor the traffic. Considering these limitations, spoofing your IP address can have little use; however, we can give it an upgrade with the idle scan.

The idle scan, or zombie scan, requires an idle system connected to the network that you can communicate with. Practically, Nmap will make each probe appear as if coming from the idle (zombie) host, then it will check for indicators whether the idle (zombie) host received any response to the spoofed probe. This is accomplished by checking the IP identification (IP ID) value in the IP header. You can run an idle scan using `nmap -sI ZOMBIE_IP 10.10.109.43`, where `ZOMBIE_IP` is the IP address of the idle host (zombie).

The idle (zombie) scan requires the following three steps to discover whether a port is open:

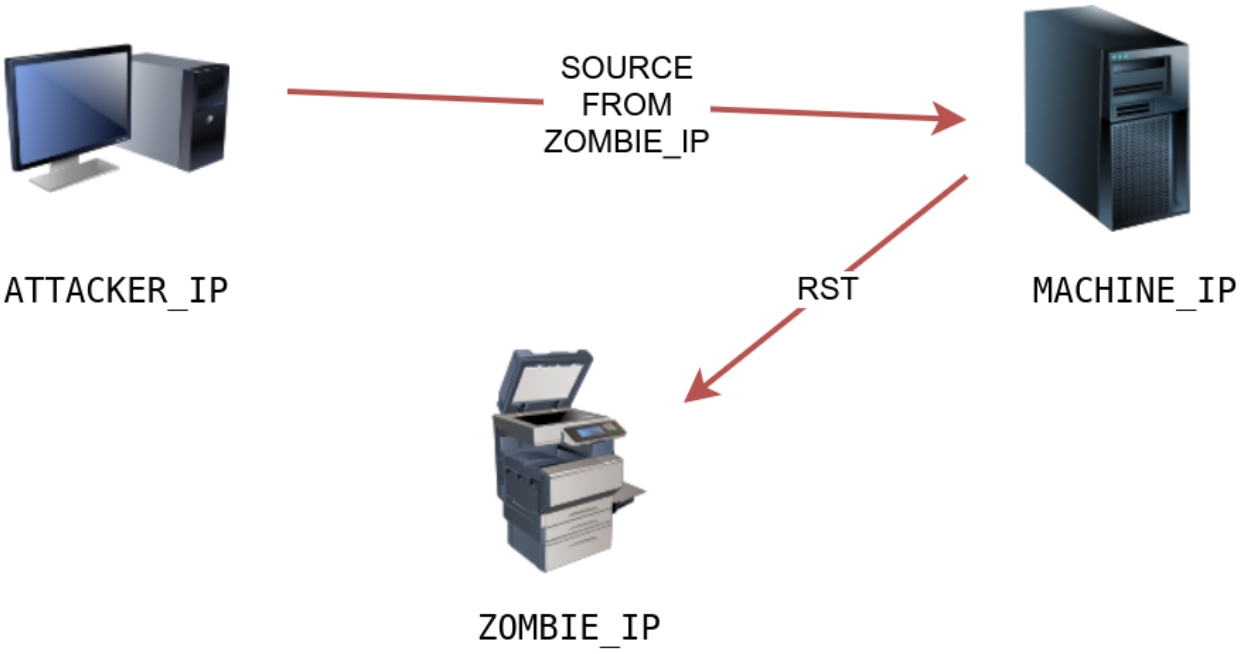
1. Trigger the idle host to respond so that you can record the current IP ID on the idle host.
2. Send a SYN packet to a TCP port on the target. The packet should be spoofed to appear as if it was coming from the idle host (zombie) IP address.
3. Trigger the idle machine again to respond so that you can compare the new IP ID with the one received earlier.

Let’s explain with figures. In the figure below, we have the attacker system probing an idle machine, a multi-function printer. By sending a SYN/ACK, it responds with an RST packet containing its newly incremented IP ID.



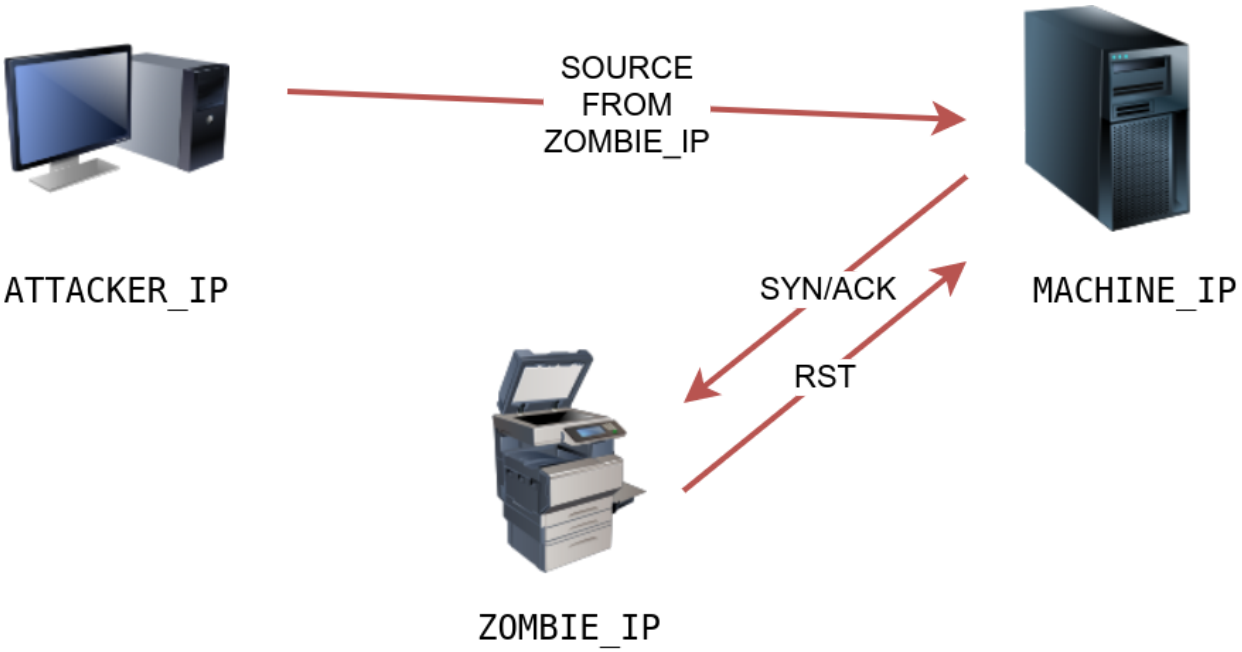
Attacker system communicates with an idle system to find its current IP ID

The attacker will send a SYN packet to the TCP port they want to check on the target machine in the next step. However, this packet will use the idle host (zombie) IP address as the source. Three scenarios would arise. In the first scenario, shown in the figure below, the TCP port is closed; therefore, the target machine responds to the idle host with an RST packet. The idle host does not respond; hence its IP ID is not incremented.



Attacker system sends to target machine
a SYN packet spoofed as sent by the idle system.
Case: Port is closed

In the second scenario, as shown below, the TCP port is open, so the target machine responds with a SYN/ACK to the idle host (zombie). The idle host responds to this unexpected packet with an RST packet, thus incrementing its IP ID.



Attacker system sends to target machine
a SYN packet spoofed as sent by the idle system.
Case: Port is open

In the third scenario, the target machine does not respond at all due to firewall rules. This lack of response will lead to the same result as with the closed port; the idle host won't increase the IP ID.

For the final step, the attacker sends another SYN/ACK to the idle host. The idle host responds with an RST packet, incrementing the IP ID by one again. The attacker needs to compare the IP ID of the RST packet received in the first step with the IP ID of the RST packet received in this third step. If the difference is 1, it means the port on the target machine was closed or filtered. However, if the difference is 2, it means that the port on the target was open.

It is worth repeating that this scan is called an idle scan because choosing an idle host is indispensable for the accuracy of the scan. If the "idle host" is busy, all the returned IP IDs would be useless.

Getting More Details:

You might consider adding `--reason` if you want Nmap to provide more details regarding its reasoning and conclusions. Consider the two scans below to the system; however, the latter adds `--reason`.

```
pentester@TryHackMe$ sudo nmap -sS 10.10.252.27
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:39 BST
Nmap scan report for ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)
Host is up (0.0020s 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
```

```
pentester@TryHackMe$ sudo nmap -sS --reason 10.10.252.27
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:40 BST
Nmap scan report for ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)
Host is up, received arp-response (0.0020s latency).
Not shown: 994 closed ports
Reason: 994 resets
PORT    STATE SERVICE REASON
22/tcp  open  ssh     syn-ack ttl 64
25/tcp  open  smtp    syn-ack ttl 64
80/tcp  open  http    syn-ack ttl 64
110/tcp open  pop3    syn-ack ttl 64
111/tcp open  rpcbind syn-ack ttl 64
143/tcp open  imap    syn-ack ttl 64
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
```

Providing the `--reason` flag gives us the explicit reason why Nmap concluded that the system is up or a particular port is open. In this console output above, we can see that this system is considered online because Nmap “received arp-response.” On the other hand, we know that the SSH port is deemed to be open because Nmap received a “syn-ack” packet back.

For more detailed output, you can consider using `-v` for verbose output or `-vv` for even more verbosity.

```
pentester@TryHackMe$ sudo nmap -sS -vv 10.10.252.27
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 10:41 BST
Initiating ARP Ping Scan at 10:41
```

```
Scanning 10.10.252.27 [1 port]
Completed ARP Ping Scan at 10:41, 0.22s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:41
Completed Parallel DNS resolution of 1 host. at 10:41, 0.00s elapsed
Initiating SYN Stealth Scan at 10:41
Scanning ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27) [1000 ports]
Discovered open port 22/tcp on 10.10.252.27
Discovered open port 25/tcp on 10.10.252.27
Discovered open port 80/tcp on 10.10.252.27
Discovered open port 110/tcp on 10.10.252.27
Discovered open port 111/tcp on 10.10.252.27
Discovered open port 143/tcp on 10.10.252.27
Completed SYN Stealth Scan at 10:41, 1.25s elapsed (1000 total ports)
Nmap scan report for ip-10-10-252-27.eu-west-1.compute.internal (10.10.252.27)
Host is up, received arp-response (0.0019s latency).
Scanned at 2021-08-30 10:41:02 BST for 1s
Not shown: 994 closed ports
Reason: 994 resets
PORT    STATE SERVICE REASON
22/tcp  open  ssh     syn-ack ttl 64
25/tcp  open  smtp    syn-ack ttl 64
80/tcp  open  http    syn-ack ttl 64
110/tcp open  pop3    syn-ack ttl 64
111/tcp open  rpcbind syn-ack ttl 64
143/tcp open  imap    syn-ack ttl 64
MAC Address: 02:45:BF:8A:2D:6B (Unknown)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
           Raw packets sent: 1002 (44.072KB) | Rcvd: 1002 (40.092KB)
```

If `-vv` does not satisfy your curiosity, you can use `-d` for debugging details or `-dd` for even more details. You can guarantee that using `-d` will create an output that extends beyond a single screen.

Nmap Post Port Scans

In this room, we focus on the steps that follow port-scanning: in particular, service detection, OS detection, Nmap scripting engine, and saving the scan results. We focus on how Nmap can be used to:

- Detect versions of the running services (on all open ports)
- Detect the OS based on any signs revealed by the target
- Run Nmap’s traceroute

- Run select Nmap scripts
- Save the scan results in various formats

Service Detection:

Once Nmap discovers open ports, you can probe the available port to detect the running service. Further investigation of open ports is an essential piece of information as the pentester can use it to learn if there are any known vulnerabilities of the service.

Adding `-sV` to your Nmap command will collect and determine service and version information for the open ports. You can control the intensity with `--version-intensity LEVEL` where the level ranges between 0, the lightest, and 9, the most complete. `-sV --version-light` has an intensity of 2, while `-sV --version-all` has an intensity of 9.

It is important to note that using `-sV` will force Nmap to proceed with the TCP 3-way handshake and establish the connection. The connection establishment is necessary because Nmap cannot discover the version without establishing a connection fully and communicating with the listening service. In other words, stealth SYN scan `-sS` is not possible when `-sV` option is chosen.

The console output below shows a simple Nmap stealth SYN scan with the `-sV` option. Adding the `-sV` option leads to a new column in the output showing the version for each detected service. For instance, in the case of TCP port 22 being open, instead of `22/tcp open ssh`, we obtain `22/tcp open ssh OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)`. Notice that the SSH protocol is guessed as the service because TCP port 22 is open; Nmap didn't need to connect to port 22 to confirm. However, `-sV` required connecting to this open port to grab the service banner and any version information it can get, such as `nginx 1.6.2`. Hence, unlike the *service* column, the *version* column is not a guess.

```
pentester@TryHackMe$ sudo nmap -sV 10.10.103.217
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:03 BST
Nmap scan report for 10.10.103.217
Host is up (0.0040s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
25/tcp  open  smtp    Postfix smtpd
80/tcp  open  http    nginx 1.6.2
110/tcp open  pop3    Dovecot pop3d
111/tcp open  rpcbind 2-4 (RPC#100000)MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)
Service Info: Host:  debra2.thm.local; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
Nmap done: 1 IP address (1 host up) scanned in 8.40 seconds
```

OS Detection and Traceroute:

OS Detection

Nmap can detect the Operating System (OS) based on its behaviour and any telltale signs in its responses. OS detection can be enabled using `-O`; this is an *uppercase O* as in OS. In this example, we ran `nmap -sS -O 10.10.103.217` on the AttackBox. Nmap detected the OS to be Linux 3.X, and then it guessed further that it was running kernel 3.13.

```
pentester@TryHackMe$ sudo nmap -sS -O 10.10.103.217
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:04 BST
Nmap scan report for 10.10.103.217
Host is up (0.00099s 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:A0:E7:B5:B6:C5 (Unknown)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3.13
OS details: Linux 3.13
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.91 seconds
```

The system that we scanned and attempted to detect its OS version is running kernel version 3.16. Nmap was able to make a close guess in this case. In another case, we scanned a Fedora Linux system with kernel 5.13.14; however, Nmap detected it as Linux 2.6.X. The good news is that Nmap detected the OS correctly; the not-so-good news is that the kernel version was wrong.

The OS detection is very convenient, but many factors might affect its accuracy. First and foremost, Nmap needs to find at least one open and one closed port on the target to make a reliable guess. Furthermore, the guest OS fingerprints might get distorted due to the rising use of virtualization and similar technologies. Therefore, always take the OS version with a grain of salt.

Traceroute

If you want Nmap to find the routers between you and the target, just add `--traceroute`. In the following example, Nmap appended a traceroute to its scan results. Note that Nmap's traceroute works slightly different than the `traceroute` command found on Linux and macOS or `tracert` found on MS Windows. Standard `traceroute` starts with a packet of low TTL (Time to Live) and keeps increasing until it reaches the target. Nmap's traceroute starts with a packet of high TTL and keeps decreasing it.

In the following example, we executed `nmap -sS --traceroute 10.10.103.217` on the AttackBox. We can see that there are no routers/hops between the two as they are connected directly.

```
pentester@TryHackMe$ sudo nmap -sS --traceroute 10.10.103.217
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:05 BST
Nmap scan report for 10.10.103.217
Host is up (0.0015s 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:A0:E7:B5:B6:C5 (Unknown)

TRACEROUTE
HOP RTT     ADDRESS
1   1.48 ms MACHINE_IP

Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
```

It is worth mentioning that many routers are configured not to send ICMP Time-to-Live exceeded, which would prevent us from discovering their IP addresses.

Nmap Scripting Engine (NSE):

A script is a piece of code that does not need to be compiled. In other words, it remains in its original human-readable form and does not need to be converted to machine language. Many programs provide additional functionality via scripts; moreover, scripts make it possible to add custom functionality that did not exist via the built-in commands. Similarly, Nmap provides support for scripts using the Lua language. A part of Nmap, Nmap Scripting Engine (NSE) is a Lua interpreter that allows Nmap to execute Nmap scripts written in Lua language. However, we don't need to learn Lua to make use of Nmap scripts.

Your Nmap default installation can easily contain close to 600 scripts. Take a look at your Nmap installation folder. Check the files at `/usr/share/nmap/scripts`, and you will notice that there are hundreds of scripts conveniently named starting with the protocol they target. We listed all the scripts starting with the HTTP on the AttackBox in the console output below; we found around 130 scripts starting with http. With future updates, you can only expect the number of installed scripts to increase.

```
pentester@AttackBox /usr/share/nmap/scripts# ls http*
http-adobe-coldfusion-apsa1301.nse      http-passwd.nse
http-affiliate-id.nse                   http-php-version.nse
http-apache-negotiation.nse             http-phpmyadmin-dir-traversal.nse
http-apache-server-status.nse           http-phpself-xss.nse
http-aspnet-debug.nse                   http-proxy-brute.nse
http-auth-finder.nse                    http-put.nse
http-auth.nse                           http-qnap-nas-info.nse
http-avaya-ipoffice-users.nse           http-referer-checker.nse
http-awstatstotals-exec.nse             http-rfi-spider.nse
http-axis2-dir-traversal.nse            http-robots.txt.nse
http-backup-finder.nse                  http-robtex-reverse-ip.nse
http-barracuda-dir-traversal.nse        http-robtex-shared-ns.nse
http-brute.nse                          http-security-headers.nse
http-cakephp-version.nse                http-server-header.nse
http-chrono.nse                         http-shellshock.nse
http-cisco-anyconnect.nse               http-sitemap-generator.nse
http-coldfusion-subzero.nse             http-slowloris-check.nse
http-comments-displayer.nse             http-slowloris.nse
http-config-backup.nse                  http-sql-injection.nse
http-cookie-flags.nse                   http-stored-xss.nse
http-cors.nse                           http-svn-enum.nse
http-cross-domain-policy.nse            http-svn-info.nse
http-csrf.nse                           http-title.nse
http-date.nse                           http-tplink-dir-traversal.nse
http-default-accounts.nse               http-trace.nse
http-devframework.nse                   http-traceroute.nse
http-dlink-backdoor.nse                 http-unsafe-output-escaping.nse
```

```
http-dombased-xss.nse                    http-useragent-tester.nse
http-domino-enum-passwords.nse           http-userdir-enum.nse
http-drupal-enum-users.nse               http-vhosts.nse
http-drupal-enum.nse                     http-virustotal.nse
http-enum.nse                            http-vlcstreamer-ls.nse
http-errors.nse                          http-vmware-path-vuln.nse
http-exif-spider.nse                     http-vuln-cve2006-3392.nse
http-favicon.nse                         http-vuln-cve2009-3960.nse
http-feed.nse                            http-vuln-cve2010-0738.nse
http-fetch.nse                           http-vuln-cve2010-2861.nse
http-fileupload-exploiter.nse            http-vuln-cve2011-3192.nse
http-form-brute.nse                      http-vuln-cve2011-3368.nse
http-form-fuzzer.nse                     http-vuln-cve2012-1823.nse
http-frontpage-login.nse                 http-vuln-cve2013-0156.nse
http-generator.nse                       http-vuln-cve2013-6786.nse
http-git.nse                             http-vuln-cve2013-7091.nse
http-gitweb-projects-enum.nse            http-vuln-cve2014-2126.nse
http-google-malware.nse                  http-vuln-cve2014-2127.nse
http-grep.nse                            http-vuln-cve2014-2128.nse
http-headers.nse                         http-vuln-cve2014-2129.nse
http-huawei-hg5xx-vuln.nse               http-vuln-cve2014-3704.nse
http-icloud-findmyiphone.nse             http-vuln-cve2014-8877.nse
http-icloud-sendmsg.nse                  http-vuln-cve2015-1427.nse
http-iis-short-name-brute.nse            http-vuln-cve2015-1635.nse
http-iis-webdav-vuln.nse                 http-vuln-cve2017-1001000.nse
http-internal-ip-disclosure.nse          http-vuln-cve2017-5638.nse
http-joomla-brute.nse                    http-vuln-cve2017-5689.nse
http-litespeed-sourcecode-download.nse   http-vuln-cve2017-8917.nse
http-ls.nse                              http-vuln-misfortune-cookie.nse
http-majordomo2-dir-traversal.nse        http-vuln-wnr1000-creds.nse
http-malware-host.nse                    http-waf-detect.nse
http-mcmp.nse                            http-waf-fingerprint.nse
http-method-tamper.nse                   http-webdav-scan.nse
http-methods.nse                         http-wordpress-brute.nse
http-mobileversion-checker.nse           http-wordpress-enum.nse
http-ntlm-info.nse                       http-wordpress-users.nse
http-open-proxy.nse                      http-xssed.nse
http-open-redirect.nse
```

You can specify to use any or a group of these installed scripts; moreover, you can install other user's scripts and use them for your scans. Let's begin with the default scripts. You can choose to run the scripts in the default category using `--script=default` or simply adding `-sC`. In addition to default, categories include auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, and vuln. A brief description is shown in the following table.

Script Category	Description
`auth`	Authentication related scripts

`broadcast`	Discover hosts by sending broadcast messages
`brute`	Performs brute-force password auditing against logins
`default`	Default scripts, same as `-sC`
`discovery`	Retrieve accessible information, such as database tables and DNS names
`dos`	Detects servers vulnerable to Denial of Service (DoS)
`exploit`	Attempts to exploit various vulnerable services
`external`	Checks using a third-party service, such as Geoplugin and Virustotal
`fuzzer`	Launch fuzzing attacks
`intrusive`	Intrusive scripts such as brute-force attacks and exploitation
`malware`	Scans for backdoors
`safe`	Safe scripts that won't crash the target
`version`	Retrieve service versions
`vuln`	Checks for vulnerabilities or exploit vulnerable services

Some scripts belong to more than one category. Moreover, some scripts launch brute-force attacks against services, while others launch DoS attacks and exploit systems. Hence, it is crucial to be careful when selecting scripts to run if you don't want to crash services or exploit them.

We use Nmap to run a SYN scan against `10.10.198.96` and execute the default scripts in the console shown below. The command is `sudo nmap -sS -sC 10.10.198.96`, where `-sC` will ensure that Nmap will execute the default scripts following the SYN scan. There are new details that appear below. Take a look at the SSH service at port 22; Nmap recovered all four public keys related to the running server. Consider another example, the HTTP service at port 80; Nmap retrieved the default page title. We can see that the page has been left as default.

```
pentester@TryHackMe$ sudo nmap -sS -sC 10.10.198.96
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 05:08 BST
Nmap scan report for ip-10-10-161-170.eu-west-1.compute.internal (10.10.161.170)
Host is up (0.0011s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
| ssh-hostkey:
|   1024 d5:80:97:a3:a8:3b:57:78:2f:0a:78:ae:ad:34:24:f4 (DSA)
|   2048 aa:66:7a:45:eb:d1:8c:00:e3:12:31:d8:76:8e:ed:3a (RSA)
|   256 3d:82:72:a3:07:49:2e:cb:d9:87:db:08:c6:90:56:65 (ECDSA)
```

```
|_  256 dc:f0:0c:89:70:87:65:ba:52:b1:e9:59:f7:5d:d2:6a (EdDSA)
25/tcp  open  smtp
|_smtp-commands: debra2.thm.local, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANC
EDSTATUSCODES, 8BITMIME, DSN,
| ssl-cert: Subject: commonName=debra2.thm.local
| Not valid before: 2021-08-10T12:10:58
|_Not valid after:  2031-08-08T12:10:58
|_ssl-date: TLS randomness does not represent time
80/tcp  open  http
|_http-title: Welcome to nginx on Debian!
110/tcp open  pop3
|_pop3-capabilities: RESP-CODES CAPA TOP SASL UIDL PIPELINING AUTH-RESP-CODE
111/tcp open  rpcbind
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          38099/tcp  status
|_  100024  1          54067/udp  status
143/tcp open  imap
|_imap-capabilities: LITERAL+ capabilities IMAP4rev1 OK Pre-login ENABLE have LOGINDISABLE
DA0001 listed SASL-IR ID more post-login LOGIN-REFERRALS IDLE
MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.21 seconds
```

You can also specify the script by name using `--script "SCRIPT-NAME"` or a pattern such as `--script "ftp*"`, which would include `ftp-brute`. If you are unsure what a script does, you can open the script file with a text reader, such as `less`, or a text editor. In the case of `ftp-brute`, it states: “Performs brute force password auditing against FTP servers.” You have to be careful as some scripts are pretty intrusive. Moreover, some scripts might be for a specific server and, if chosen at random, will waste your time with no benefit. As usual, make sure that you are authorized to launch such tests on the target server.

Let’s consider a benign script, `http-date`, which we guess would retrieve the http server date and time, and this is indeed confirmed in its description: “Gets the date from HTTP-like services. Also, it prints how much the date differs from local time…” On the AttackBox, we execute `sudo nmap -sS -n --script "http-date" 10.10.198.96` as shown in the console below.

```
pentester@TryHackMe$ sudo nmap -sS -n --script "http-date" 10.10.198.96
Starting Nmap 7.60 ( https://nmap.org ) at 2021-09-10 08:04 BST
Nmap scan report for 10.10.198.96
Host is up (0.0011s latency).
```

```
Not shown: 994 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
|_http-date: Fri, 10 Sep 2021 07:04:26 GMT; 0s from local time.
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
MAC Address: 02:44:87:82:AC:83 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.78 seconds
```

Finally, you might expand the functionality of Nmap beyond the official Nmap scripts; you can write your script or download Nmap scripts from the Internet. Downloading and using a Nmap script from the Internet holds a certain level of risk. So it is a good idea not to run a script from an author you don't trust.

Saving the Output:

Whenever you run a Nmap scan, it is only reasonable to save the results in a file. Selecting and adopting a good naming convention for your filenames is also crucial. The number of files can quickly grow and hinder your ability to find a previous scan result. The three main formats are:

1. Normal
2. Grepable (`grep`able)
3. XML

There is a fourth one that we cannot recommend:

- Script Kiddie

Normal

As the name implies, the normal format is similar to the output you get on the screen when scanning a target. You can save your scan in normal format by using `-oN FILENAME`; N stands for normal. Here is an example of the result.

```
pentester@TryHackMe$ cat 10.10.188.188_scan.nmap
# Nmap 7.60 scan initiated Fri Sep 10 05:14:19 2021 as: nmap -sS -sV -O -oN 10.10.188.188_
scan MACHINE_IP
```

```
Nmap scan report for 10.10.188.188
Host is up (0.00086s latency).
Not shown: 994 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
25/tcp  open  smtp    Postfix smtpd
80/tcp  open  http    nginx 1.6.2
110/tcp open  pop3    Dovecot pop3d
111/tcp open  rpcbind 2-4 (RPC#100000)143/tcp open  imap    Dovecot imapd
MAC Address: 02:A0:E7:B5:B6:C5 (Unknown)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3.13
OS details: Linux 3.13
Network Distance: 1 hop
Service Info: Host:  debra2.thm.local; OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
# Nmap done at Fri Sep 10 05:14:28 2021 -- 1 IP address (1 host up) scanned in 9.99 second
s
```

Grepable

The grepable format has its name from the command `grep`; grep stands for Global Regular Expression Printer. In simple terms, it makes filtering the scan output for specific keywords or terms efficient. You can save the scan result in grepable format using `-oG FILENAME`. The scan output, displayed above in normal format, is shown in the console below using grepable format. The normal output is 21 lines; however, the grepable output is only 4 lines. The main reason is that Nmap wants to make each line meaningful and complete when the user applies `grep`. As a result, in grepable output, the lines are so long and are not convenient to read compared to normal output.

```
pentester@TryHackMe$ cat 10.10.188.188_scan.gnmap
# Nmap 7.60 scan initiated Fri Sep 10 05:14:19 2021 as: nmap -sS -sV -O -oG 10.10.188.188_
scan MACHINE_IP
Host: 10.10.188.188 Status: Up
Host: MACHINE_IP  Ports: 22/open/tcp//ssh//OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)/,
 25/open/tcp//smtp//Postfix smtpd/, 80/open/tcp//http//nginx 1.6.2/, 110/open/tcp//pop3//D
ovecot pop3d/, 111/open/tcp//rpcbind//2-4 (RPC #100000)/, 143/open/tcp//imap//Dovecot imap
d/  Ignored State: closed (994) OS: Linux 3.13  Seq Index: 257  IP ID Seq: All zeros
# Nmap done at Fri Sep 10 05:14:28 2021 -- 1 IP address (1 host up) scanned in 9.99 second
s
```

An example use of `grep` is `grep KEYWORD TEXT_FILE`; this command will display all the lines containing the provided keyword. Let's compare the output of using `grep` on normal output and grepable output. You will notice that the former does not provide the IP address of the host. Instead, it returned `80/tcp open http nginx 1.6.2`, making it very inconvenient if you are sifting through the scan results of multiple systems. However, the latter provides enough information, such as the host's IP address, in each line to make it complete.

```
pentester@TryHackMe$ grep http 10.10.188.188_scan.nmap
80/tcp  open  http    nginx 1.6.2
OS and Service detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
```

```
pentester@TryHackMe$ grep http 10.10.188.188_scan.gnmap
Host: 10.10.188.188 Ports: 22/open/tcp//ssh//OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.
0)/, 25/open/tcp//smtp//Postfix smtpd/, 80/open/tcp//http//nginx 1.6.2/, 110/open/tcp//pop
3//Dovecot pop3d/, 111/open/tcp//rpcbind//2-4 (RPC #100000)/, 143/open/tcp//imap//Dovecot
 imapd/
Ignored State: closed (994) OS:Linux 3.13 Seq Index: 257  IP ID Seq: All zeros
```

XML

The third format is XML. You can save the scan results in XML format using `-oX FILENAME`. The XML format would be most convenient to process the output in other programs. Conveniently enough, you can save the scan output in all three formats using `-oA FILENAME` to combine `-oN`, `-oG`, and `-oX` for normal, grepable, and XML.

Script Kiddie

A fourth format is script kiddie. You can see that this format is useless if you want to search the output for any interesting keywords or keep the results for future reference. However, you can use it to save the output of the scan `nmap -sS 127.0.0.1 -oS FILENAME`, display the output filename, and look 31337 in front of friends who are not tech-savvy.

```
pentester@TryHackMe$ cat 10.10.188.188_scan.kiddie
$tart!ng nMaP 7.60 ( httpz://nMap.0rG ) at 2021-09-10 05:17 B$T
Nmap scan rEp0rt f0r |p-10-10-161-170.EU-w3$t-1.C0mputE.intErnaL (10.10.161.170)
HOSt !s uP (0.00095s LatEncy).
N0T$H0wn: 994 closed pOrtSPoRT    st4Te SeRViC3 VERS1on
22/tcp  Open  ssH     Op3n$$H 6.7p1 Deb|an 5+dEb8u8 (pr0t0COl 2.0)25/tCp  Op3n  SmTp    P0
```

```
$Tf!x Smtpd80/tcp  0p3n  http    Ng1nx 1.6.2
110/tCP 0pen  pOP3    d0v3coT P0p3D
111/TcP op3n  RpcbInd 2-4 (RPC#100000)143/Tcp opEn  Imap    Dovecot 1mApd
mAC 4Ddr3sz: 02:40:e7:B5:B6:c5 (Unknown)
Netw0rk d!stanc3: 1 h0p
$3rv1c3 InFO: Ho$t:  dEBra2.thM.lOcal; 0s:Linux; cPe: cP3:/0:linux:l|nux_k3rnel0S and serv
Ic3 D3tEcti0n pErf0rm3d. Plea$e r3p0rt any !nc0RrecT rE$ultz at hTtpz://nmap.0rg/$ubmit/ .
Nmap d0nE: 1 |P addr3SS (1 hoSt up) $CaNnEd !n 21.80 s3c0Ndz
```