

## **EXPERIMENT 8**

# **Nmap Live Host Discovery**

### **Aim:**

To Discover Live Hosts Using Nmap Scans (ARP, ICMP, TCP/UDP) on the TryHackMe Platform

### **Introduction:**

When targeting a network, we need an efficient tool to handle repetitive tasks. This tool should help us find out which systems are active and what services are running on those systems. The tool that we will rely on is Nmap. The first question about finding live computers is answered in this room. This room is the first in a series of four rooms dedicated to Nmap. This room is the first of four in this Nmap series. These four rooms are also part of the Network Security module.

- Nmap Live Host Discovery
- Nmap Basic Port Scans
- Nmap Advanced Port Scans
- Nmap Post Port Scans

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

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

## Task 1 Introduction

This room outlines the processes that Nmap takes before port-scanning to find which systems are online. This stage is critical since attempting to port-scan offline systems will merely waste time and create unneeded network noise (because it is active recon).

The following is the information that will be covered in an attempt to discover live hosts:

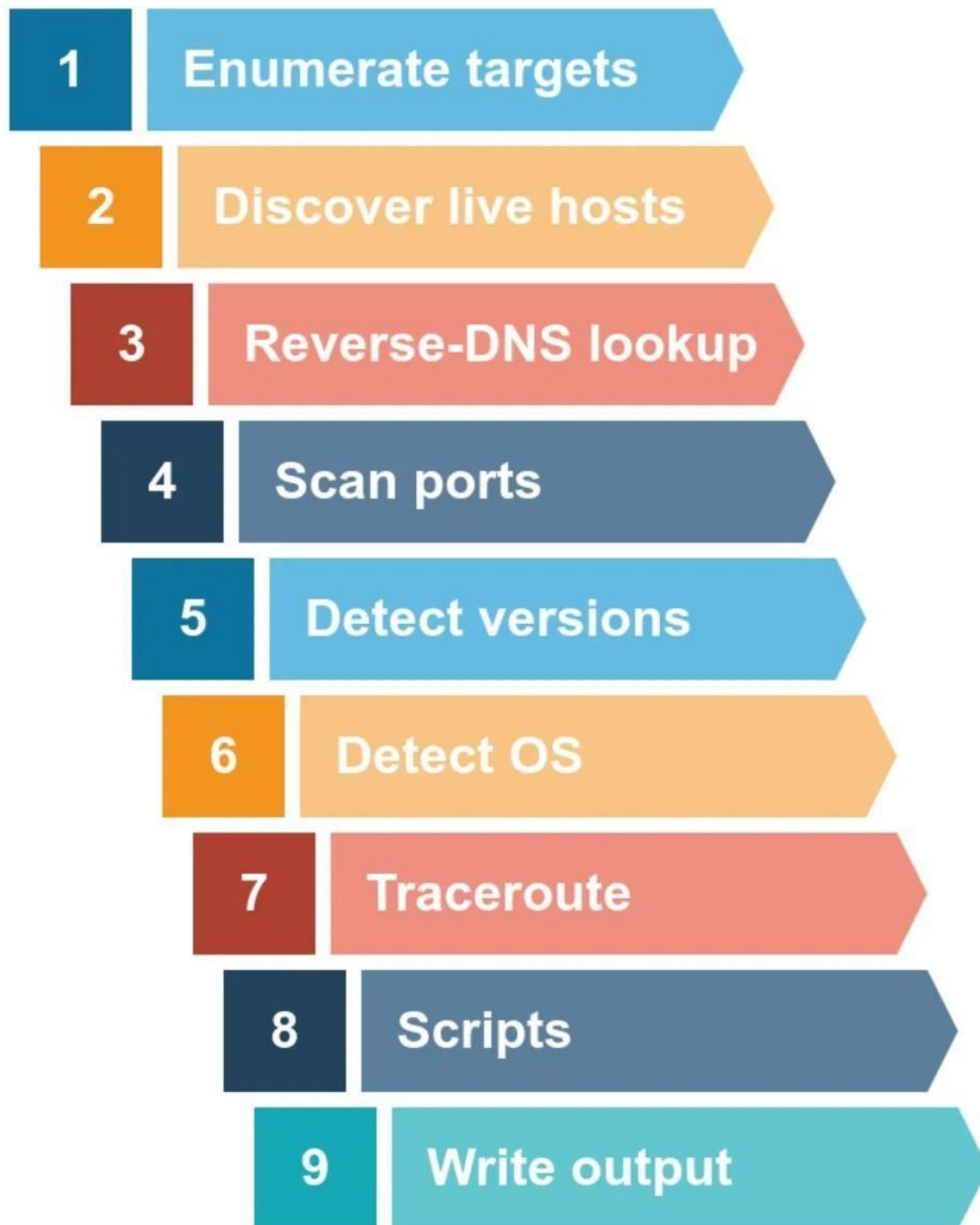
- ARP scan: This scan uses ARP requests to discover live hosts
- ICMP scan: This scan uses ICMP requests to identify live hosts
- TCP/UDP ping scan: This scan sends packets to TCP ports and UDP ports to determine live hosts.

There will be two scanners introduced:

- arp-scan
- masscan

Nmap (Network Mapper) — It is a well-known tool for mapping networks, locating live hosts, and detecting running services. Nmap's scripting engine can be used to extend its capabilities, such as fingerprinting services and exploiting flaws.

The scans typically follow the steps represented in the image below, but several are optional and are conditional on the "command-line" options provided prior to the scan:



[Question 1.1] Some of these questions will require the use of a static site to answer the task questions, while others require the use of the AttackBox and the target VM.

Answer: No answer is needed.

---

## Task 2 Subnetworks

Let's go through a few concepts before we get started on the primary tasks.

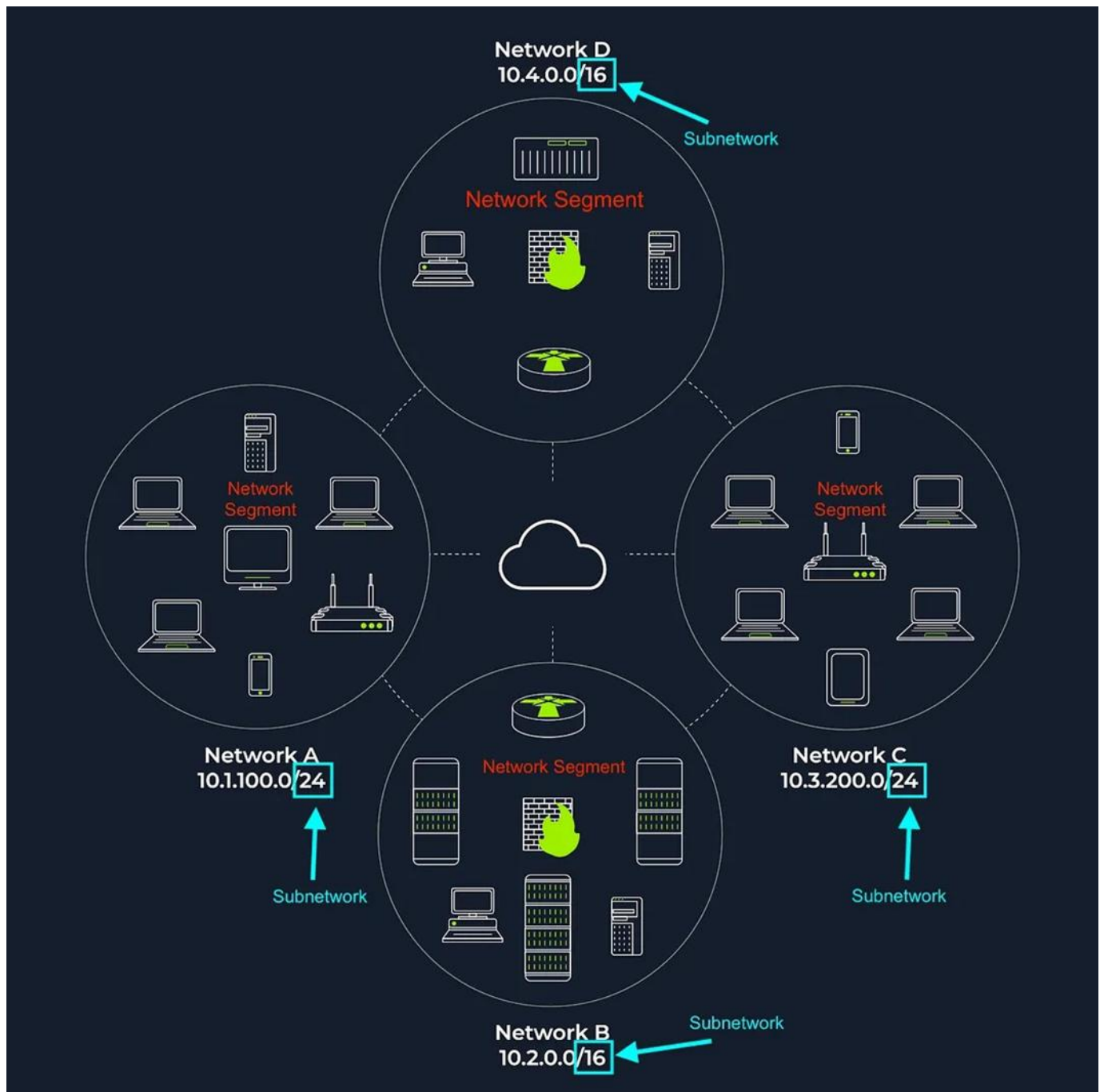
**Network Segment** — It is a collection of computers linked together through a common “medium.”

- In this case, the “medium” could be an Ethernet switch or a WiFi access point.
- Physical connection

**Subnetwork** — It is usually the equivalent of one or more network segments linked together and configured to utilize the same router in an IP network.

- It has its own IP address range and is joined to a larger network using a router.
- Logical connection

Depending on the network, a firewall may be used to enforce security policies.



The diagram above illustrates two types of subnets:

1. 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.
2. 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.

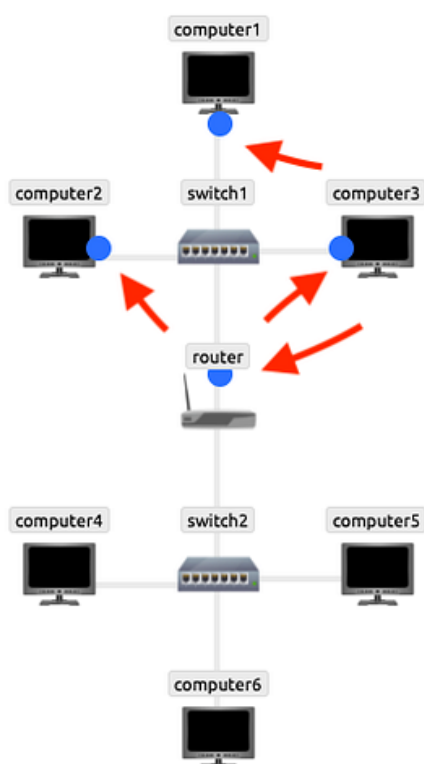
If you're on the same subnet, you'd expect your scanner to use ARP (Address Resolution Protocol) queries to find live hosts.

- An ARP query seeks the hardware address (MAC address) in order to enable communication over the link-layer.
- We can assume from this that the host is online.

If you're on Network A, you can only use ARP to discover devices on that subnet (10.1.100.0/24).

- Assume you are linked to a subnet that is unique from the target system's subnet(s).
- In that situation, all packets generated by your scanner will be sent through the default gateway (router) to reach systems on another subnet; however, ARP queries will not be routed and therefore will not be able to cross the subnet router.
- ARP is a link-layer protocol that binds ARP packets to their subnet.

[Question 2.1] How many devices can see the ARP Request?



Legend	
●	TCP Packet
●	TCP Handshake Packet
●	UDP Packet
●	ARP Packet
●	Ping Packet

Send Packet	
From:	<input type="text" value="computer1"/>
To:	<input type="text" value="computer1"/>
Packet Type:	<input type="text" value="arp_request"/>
Data:	<input type="text" value="computer6"/>
<input type="button" value="Send Packet"/>	

Network Log	
<div></div>	

Movement of the “Blue Dot”:

1st – Computer 1 > Switch 1

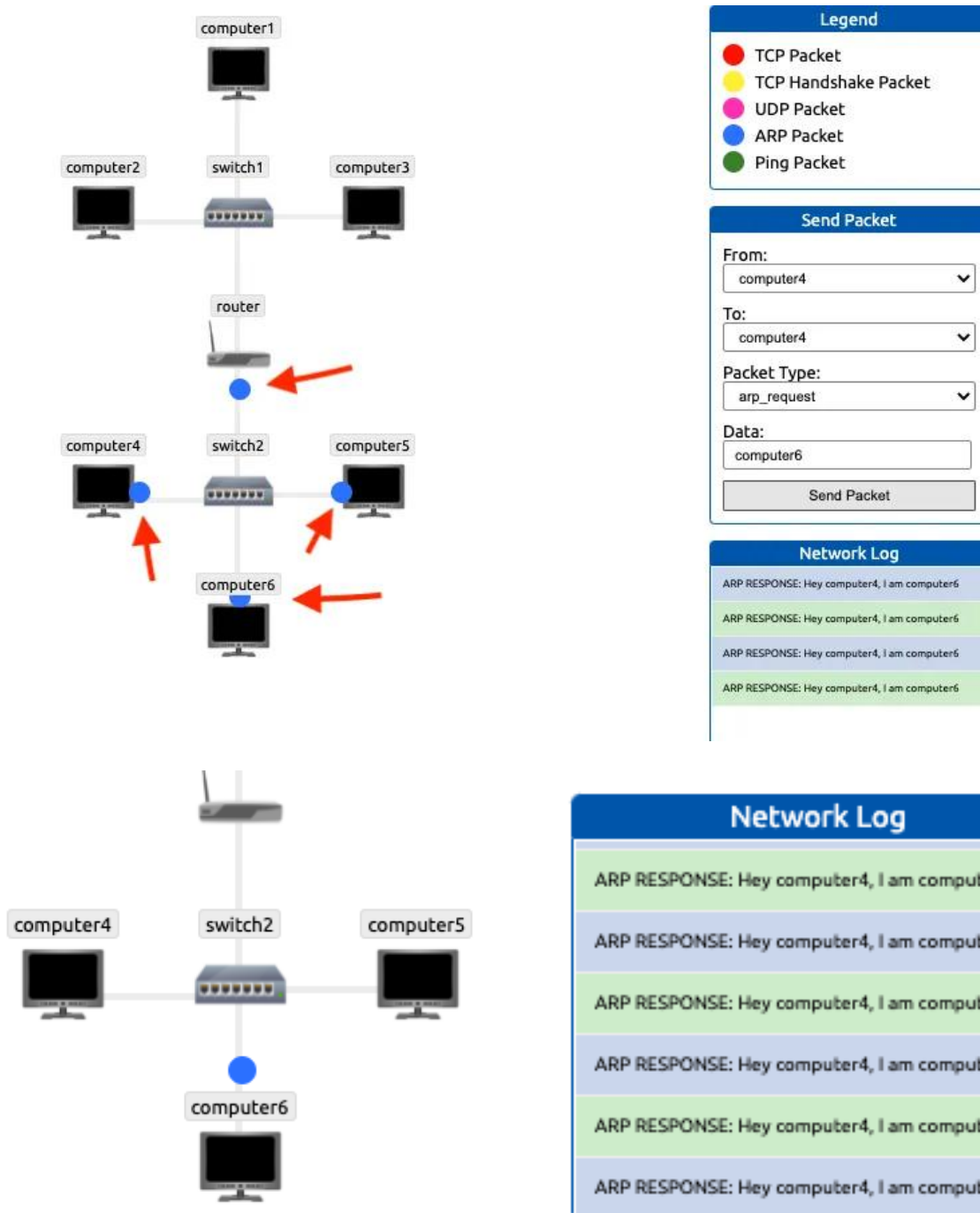
2nd – Switch 1 > Computers 1, 2, 3, and Router

Answer: 4

[Question 2.2] Did computer6 receive the ARP Request? (Y/N)

Answer: N

[Question 2.3] How many devices can see the ARP Request?



Movement of the “Blue Dot”:

1st – Computer 4 > Switch 2

2nd – Switch 2 > Computer 4, 6, 5, and Router

3rd – Computer 6 > Computer 4 – In response to “Hey computer4, I am computer 6”.

Answer: 4

[Question 2.4] Did computer6 reply to the ARP Request? (Y/N)

Answer: Y

---

### Task 3 Enumerating Targets

We need to identify the targets we want to scan before we conduct any “action.”

In general, you can provide a list, a range, or a subnet.

Here are some examples of target specifications (kindly ignore < >):

#### 1. List

- < MACHINE\_IP > < scanme.nmap.org > < example.com >
- will scan 3 IP addresses

#### 2. range

- 10.11.12.15-20
- will scan 6 IP addresses

#### 3. subnet

- < MACHINE\_IP/30 >
- will scan 4 IP addresses

You can also provide a file as input for your list of targets:

example: `nmap -iL list_of_hosts.txt`

- notice that it's a file name



If you prefer to see a list of hosts that Nmap will scan, go to:

- This option will provide you with a complete list of the hosts that Nmap will scan without actually scanning them; however, Nmap will perform reverse-DNS resolution on all targets to retrieve their names.

`nmap -sL <TARGETS>`

Names might give a variety of information to the pentester. (If you don't want Nmap to connect to the DNS server, use `-n`.)

[Question 3.1] What is the first IP address Nmap would scan if you provided 10.10.12.13/26 as your target?

`nmap -sL 10.10.12.13/29`

```
(root@Aircon)-[/home/kali]
# nmap -sL 10.10.12.13/29
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-20 13:29 EDT
Nmap scan report for 10.10.12.8
Nmap scan report for 10.10.12.9
Nmap scan report for 10.10.12.10
Nmap scan report for 10.10.12.11
Nmap scan report for 10.10.12.12
Nmap scan report for 10.10.12.13
Nmap scan report for 10.10.12.14
Nmap scan report for 10.10.12.15
Nmap done: 8 IP addresses (0 hosts up) scanned in 0.16 seconds
```

Answer: 10.10.12.8

[Question 3.2] How many IP addresses will Nmap scan if you provide the following range 10.10.0-255.101-125 ?

`nmap -sL -n 10.10.0-255.101-125`

```
(root@Aircon)-[/home/kali]
# nmap -sL -n 10.10.0-255.101-125
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-20 13:32 EDT
Nmap scan report for 10.10.0.101
Nmap scan report for 10.10.0.102
Nmap scan report for 10.10.0.103
Nmap scan report for 10.10.0.104
Nmap scan report for 10.10.0.105
Nmap scan report for 10.10.255.124
Nmap scan report for 10.10.255.125
Nmap done: 6400 IP addresses (0 hosts up) scanned in 0.07 seconds
```

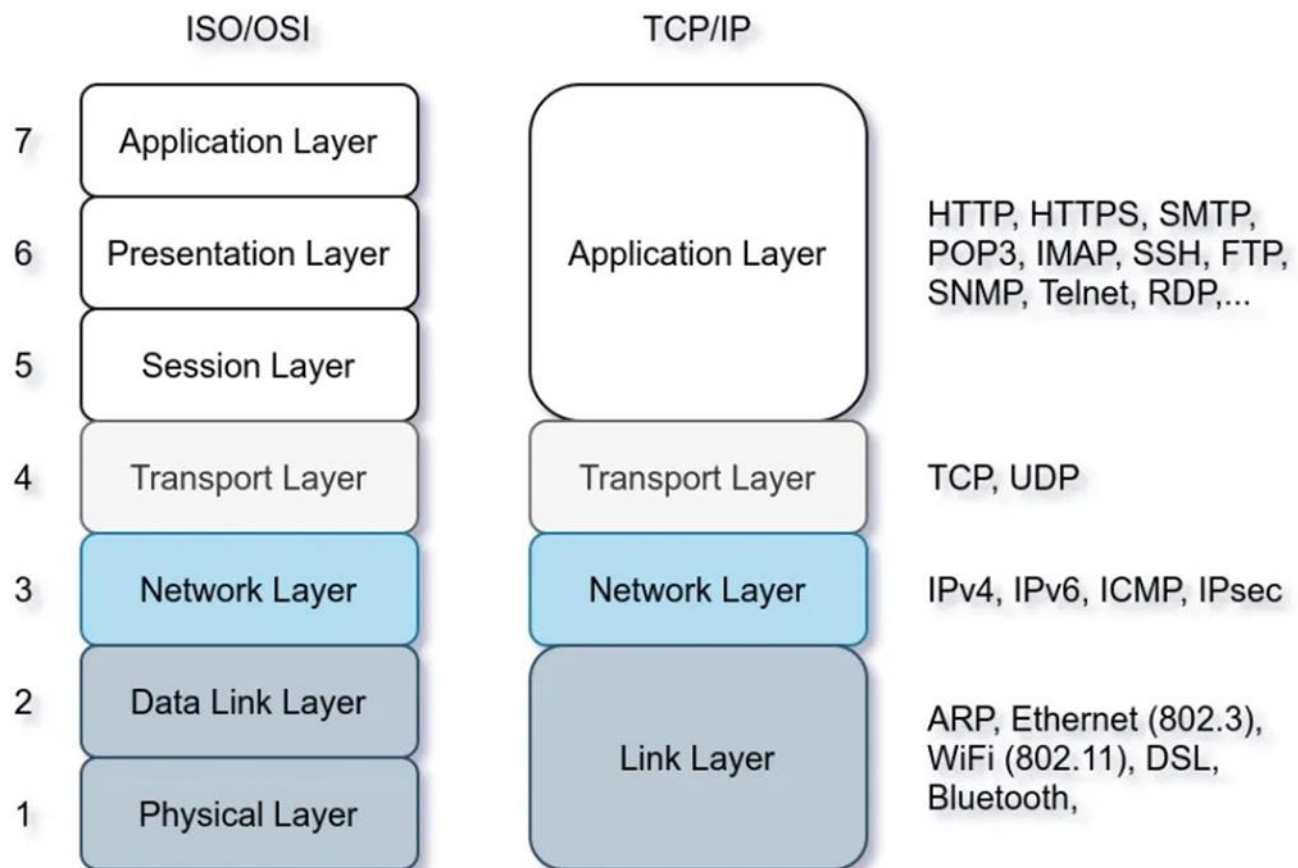
Answer: 6400

---

## Task 4 Discovering Live Hosts

Next, let's go over the TCP/IP layers illustrated in the image. We will use the protocols to identify the live hosts.

From the bottom to the top, we can use:



1. ARP from Link Layer - commonly used for Man-In-Middle-Attack (ARP Spoofing and ARP Poisoning)
2. ICMP from Network Layer - commonly used in DDoS attacks
3. TCP from Transport Layer - SYN flooding, TCP reset, TCP session hijacking attacks
4. UDP from Transport Layer - Also used in DDoS attacks, where the attacker targets and overwhelms random ports on the host with IP packets containing User Datagram Protocol (UDP) packets

ARP serves only one purpose:

- Sending a frame to the network segment's broadcast address and asking the computer with a certain IP address to respond with its MAC (hardware) address

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

Pinging a system on the same network requires an ARP query before the ICMP Echo.

- Despite the fact that TCP and UDP are transport layers, a scanner can send a specially constructed packet to common TCP or UDP ports to see if the target responds.
- This strategy is effective, particularly when ICMP Echo is disabled.

[Question 4.1] What is the type of packet that computer1 sent before the ping?

Network Log
ARP REQUEST: Who has computer3 tell computer1
ARP RESPONSE: Hey computer1, I am computer3
ARP REQUEST: Who has computer3 tell computer1
PING: Sending Ping Request packet from computer1 to computer3
ARP RESPONSE: Hey computer1, I am computer3
PING: computer3 received ping request from

Answer: ARP Request

[Question 4.2] What is the type of packet that computer1 received before being able to send the ping?

Answer: ARP Response

[Question 4.3] How many computers responded to the ping request?

Answer: 1

[Question 4.4] What is the name of the first device that responded to the first ARP Request?

Network Log
ROUTING: computer2 says computer5 is not on my local network sending to gateway: router
ARP REQUEST: Who has router tell computer2
ARP RESPONSE: Hey computer2, I am router
PING: Sending Ping Request packet from computer2 to computer5
ARP REQUEST: Who has computer5 tell router
ARP RESPONSE: Hey router I am computer5

Answer: router

[Question 4.5] What is the name of the first device that responded to the second ARP Request?

Answer: computer5

[Question 4.6] Send another Ping Request. Did it require new ARP Requests? (Y/N)

Answer: N

---

## Task 5 Nmap Host Discovery Using ARP

It is critical to avoid wasting time port-scanning an offline host or an IP address that is not in use since it is important to know which “hosts” are operational (online).

When no host discovery options are available, Nmap uses the following methods to find 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 to port 80, TCP SYN 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.

This means that there are significant distinctions between having a “privileged” and “unprivileged” user in the “terminal box” since the processes are different.

It’s worth noting that Nmap, by default, uses a ping scan to find live hosts, THEN proceeds to scan live hosts only.

If you wish to use Nmap to discover internet hosts without port-scanning live systems, use `nmap -sn TARGETS`.

Following that, “ARP scan” is only feasible if you are on the same subnet as the target systems.

The MAC address is required for the link-layer header, which includes the source and destination MAC addresses among other data.

The OS sends an ARP query to obtain the MAC address. A host that responds to ARP queries is operational. Again, the ARP query is only effective if the target is on the same subnet as you, i.e., on the same Ethernet/WiFi network. During a Nmap scan of a local network, you can expect to notice a lot of ARP queries.

If you simply want an ARP scan without port scanning

- use: `nmap -PR -sn <TARGETS>`
- where `-PR` specifies that you only want an ARP scan.

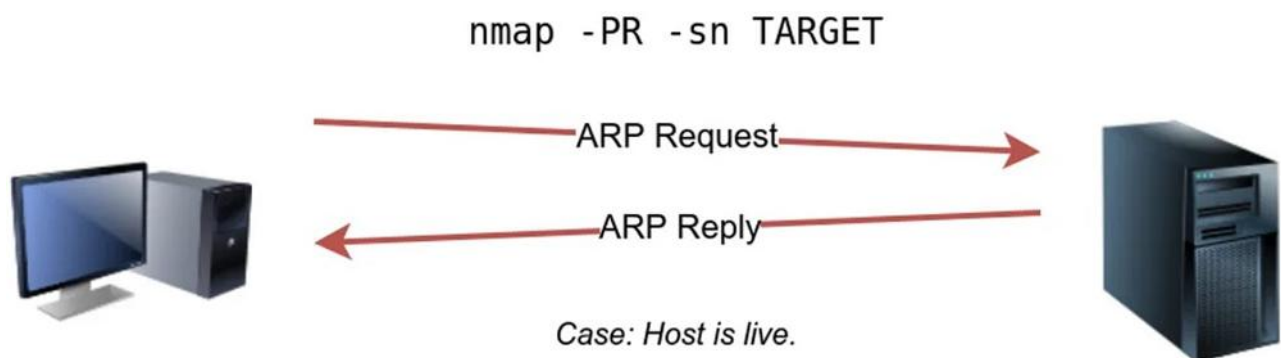
The sample below shows Nmap using ARP for host finding rather than port scanning. To discover all the live systems on the same subnet as our target machine, we use `nmap -PR -sn <MACHINE IP>/24`.

```
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
```

Based on the image below, if you do the scan info provided, this is what it looks like:

`nmap -PR -sn <target>`



When we examine the packets created by tools like tcpdump or Wireshark, we can witness network traffic similar to the figure below. Wireshark displays the source MAC address, destination MAC address, protocol, and query for each ARP request in the figure below.

The source address is our AttackBox's MAC address, but the destination is the broadcast address because we don't know the target's MAC address. However, we can see the target's IP address in the Info column. We can see in the diagram that we are seeking the MAC addresses of all IP addresses on the subnet, beginning with 10.10.210.1.

The host with the IP address in question will respond with an ARP response with its MAC address, indicating that it is online.



The image shows a Wireshark packet capture window titled "nmap-PR-sn-AttackBox.pcapng". The filter bar shows "arp". The packet list displays 18 ARP requests. The first packet is highlighted in blue.

Source	Destination	Protocol	Info
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.1? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.2? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.3? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.4? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.5? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.7? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.8? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.9? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.10? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.11? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.1? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.2? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.3? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.4? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.5? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.7? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.8? Tell 10.10.210.6

At the bottom of the window, it says: "nmap-PR-sn-AttackBox.pcapng Packets: 1480 · Displayed: 512 (34.6%) Profile: Default"

While addressing ARP scans, we should mention a scanner designed specifically for ARP queries:

`arp-scan --localnet`, or simply `arp-scan -l`, is a popular option. This command will send ARP requests to every valid IP address on your local network.

If your system has more than one interface and you want to discover the live hosts on one of them:

use `-I` to indicate the interface.

For example, will send ARP requests to all valid IP addresses on the `eth0` interface:

```
sudo arp-scan -I eth0 -l
```

In the following example, we used `arp-scan <AttackBox's IP>/24` to scan the AttackBox's subnet. We obtained the same three active targets because we did this scan close to the previous one:

```
nmap -PR -sn <ATTACKBOX IP>/24
```

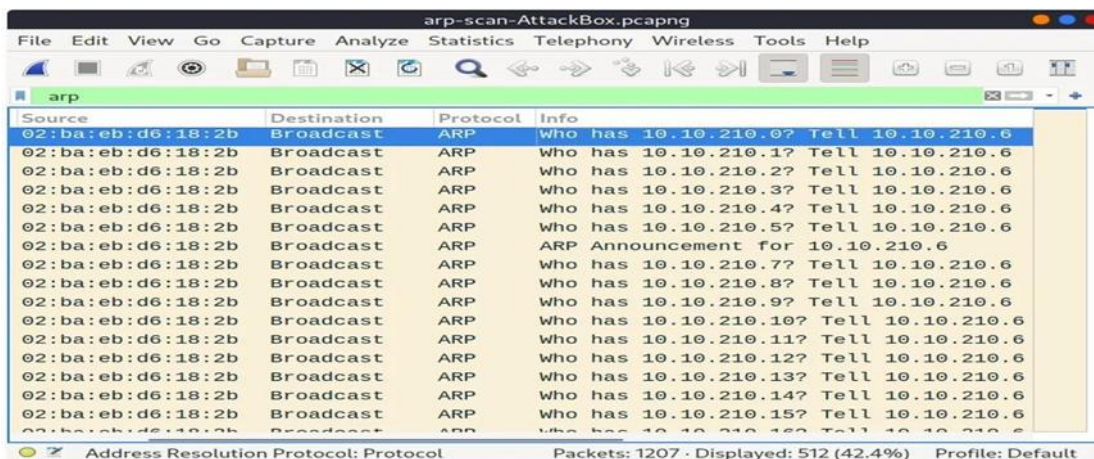
```

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 a large number of ARP inquiries, which we may inspect with tcpdump, Wireshark, or a comparable tool. We can see that the traffic patterns produced by arp-scan and nmap -PR -sn are comparable. The Wireshark output is seen below.



Source	Destination	Protocol	Info
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.0? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.1? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.2? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.3? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.4? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.5? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	ARP Announcement for 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.7? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.8? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.9? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.10? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.11? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.12? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.13? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.14? Tell 10.10.210.6
02:ba:eb:d6:18:2b	Broadcast	ARP	Who has 10.10.210.15? Tell 10.10.210.6

[Question 5.1] How many devices are you able to discover using ARP requests?

Network Log
ARP RESPONSE: Hey computer1, I am computer2
ARP RESPONSE: Hey computer1, I am computer3
ARP RESPONSE: Hey computer1, I am router

Answer: 3



## Task 6 Nmap Host Discovery Using ICMP

It's interesting to discover that we can "ping" every IP address on a target network and check who responds to our ping (ICMP Type 8/Echo) queries with a ping reply (ICMP Type 0), even though it's the simplest but not always reliable strategy.

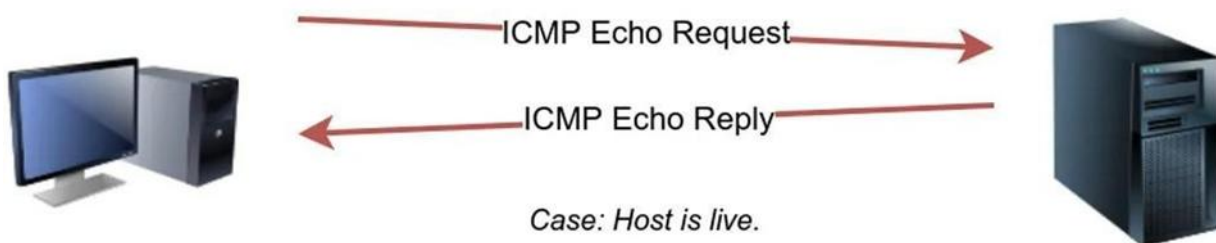
Why? — Because many firewalls block ICMP echo, new versions of Microsoft Windows include a host firewall that by default blocks ICMP echo requests. If your target is on the same subnet, an ARP query will come before an ICMP request.

Add the `-PE` option to utilize ICMP echo requests to discover live hosts. (Remember to add `-sn` if you don't want to run a port scan after that.)

An ICMP echo scan works by making an ICMP echo request and expecting the target to respond with an ICMP echo reply if it is online.

`nmap -PE -sn <Target>`

`nmap -PE -sn TARGET`



In the following example, we used `nmap -PE -sn MACHINE IP/24` to scan the target's subnet. This scan will send ICMP echo packets to all of the subnet's IP addresses. Again, we anticipate live hosts to respond; however, keep in mind that many firewalls block ICMP.

```
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 indicates that eight hosts are operational, as well as their MAC addresses. Unless the targets are on the same subnet as our system, we don't expect to learn their MAC addresses. The information above demonstrates that Nmap did not need to transmit ICMP packets because the ARP answers it received indicated that these hosts were operational.

We will perform the scan described above, but this time from a system on a different network. The outcomes are comparable, but without the MAC addresses.

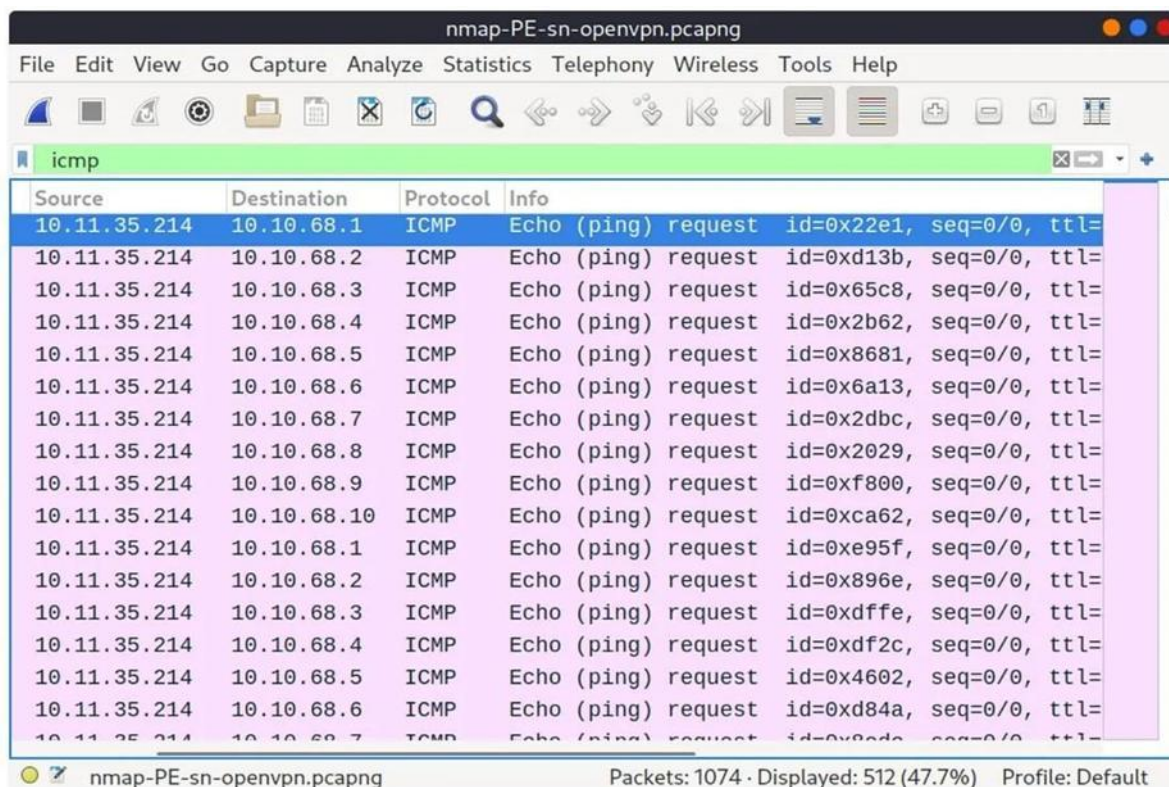
```

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 examine the network packets with a tool such as Wireshark, you will see something like the image below. As you can see, we have one source IP address on a different network than the destination subnet, which is issuing ICMP echo queries to all IP addresses in the target subnet to check which one responds.



The image shows a Wireshark packet capture window titled "nmap-PE-sn-openvpn.pcapng". The filter bar shows "icmp". The packet list table displays 20 ICMP Echo (ping) requests. All requests originate from the source IP 10.11.35.214 and are sent to various destinations within the 10.10.68.0/24 subnet. The packet details pane on the right shows the structure of an ICMP Echo (ping) request, including the ID, sequence number, and TTL.

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Echo (ping) request id=0x22e1, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Echo (ping) request id=0xd13b, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Echo (ping) request id=0x65c8, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Echo (ping) request id=0x2b62, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Echo (ping) request id=0x8681, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Echo (ping) request id=0x6a13, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Echo (ping) request id=0x2dbc, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Echo (ping) request id=0x2029, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Echo (ping) request id=0xf800, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Echo (ping) request id=0xca62, seq=0/0, ttl=
10.11.35.214	10.10.68.11	ICMP	Echo (ping) request id=0xe95f, seq=0/0, ttl=
10.11.35.214	10.10.68.12	ICMP	Echo (ping) request id=0x896e, seq=0/0, ttl=
10.11.35.214	10.10.68.13	ICMP	Echo (ping) request id=0xdffe, seq=0/0, ttl=
10.11.35.214	10.10.68.14	ICMP	Echo (ping) request id=0xdf2c, seq=0/0, ttl=
10.11.35.214	10.10.68.15	ICMP	Echo (ping) request id=0x4602, seq=0/0, ttl=
10.11.35.214	10.10.68.16	ICMP	Echo (ping) request id=0xd84a, seq=0/0, ttl=
10.11.35.214	10.10.68.17	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=
10.11.35.214	10.10.68.18	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=
10.11.35.214	10.10.68.19	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=
10.11.35.214	10.10.68.20	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=
10.11.35.214	10.10.68.21	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=
10.11.35.214	10.10.68.22	ICMP	Echo (ping) request id=0x80da, seq=0/0, ttl=



Because ICMP echo queries are frequently blocked, you should consider using ICMP Timestamp or ICMP Address Mask requests to determine whether a system is available.

Nmap sends a timestamp request (ICMP Type 13) and waits for a Timestamp response (ICMP Type 14). The -PP option instructs Nmap to use ICMP timestamp requests.

```
nmap -PP -sn TARGET
```

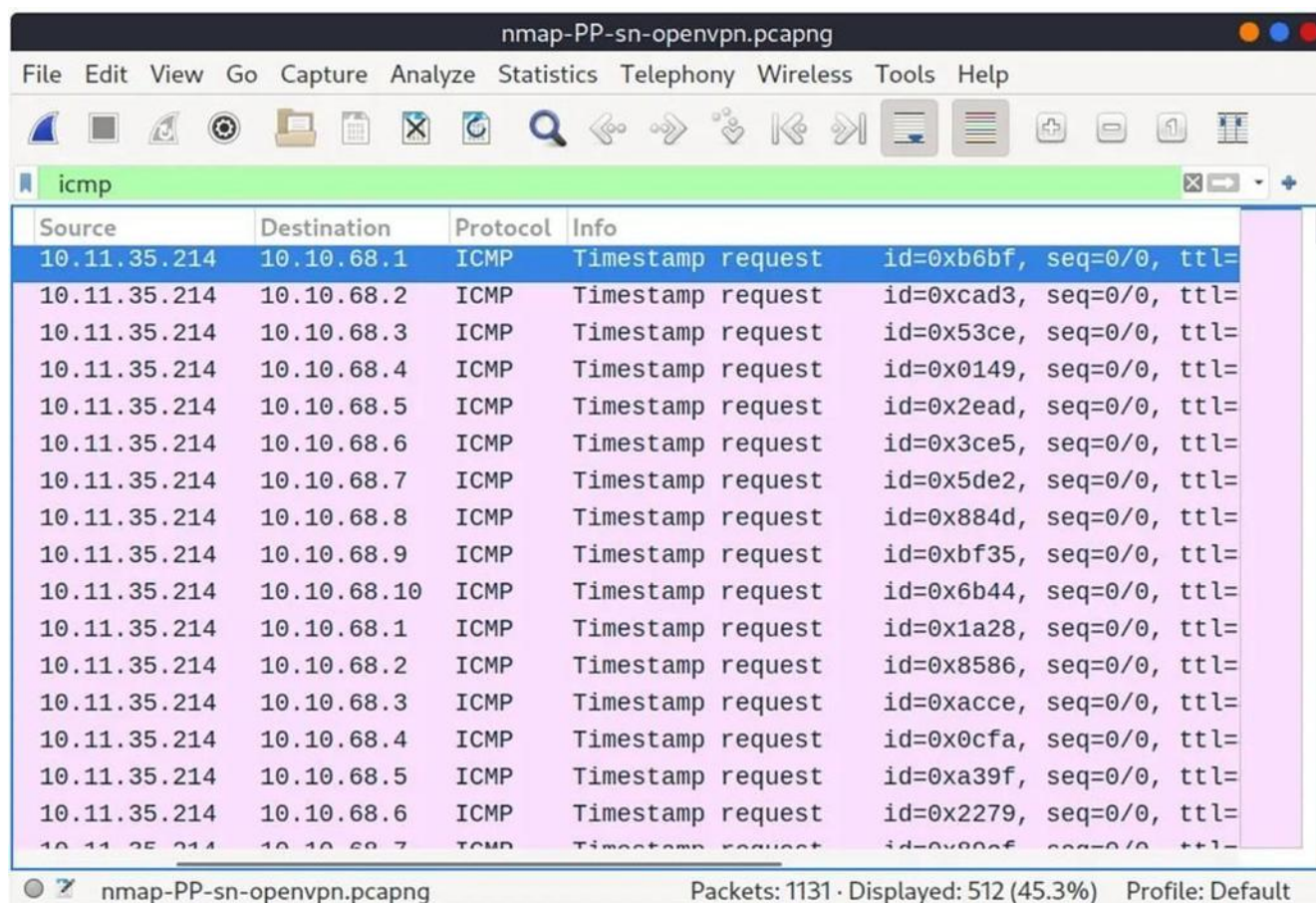


To discover the online computers on the target machine subnet, we run:  
`nmap -PP -sn MACHINE IP/24`

```
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
```

This scan, like the last ICMP scan, will send several ICMP timestamp queries to every valid IP address in the target subnet. The picture below shows one source IP address sending ICMP packets to every feasible IP address in order to locate online hosts.



The image shows a Wireshark packet capture window titled 'nmap-PP-sn-openvpn.pcapng'. The filter is set to 'icmp'. The packet list shows 17 ICMP timestamp requests from source 10.11.35.214 to destinations in the 10.10.68.0/24 subnet. The packet details pane shows the first packet's structure: ICMP header with type 3 (Timestamp request), id=0xb6bf, seq=0/0, and ttl=64.

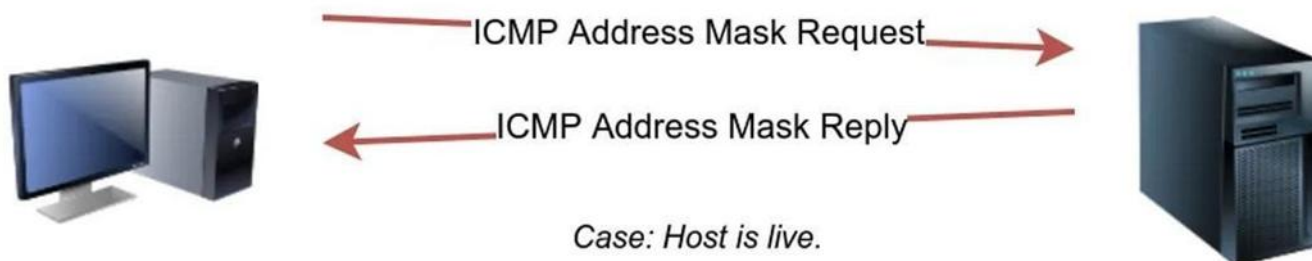
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Timestamp request id=0xb6bf, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Timestamp request id=0xcad3, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Timestamp request id=0x53ce, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Timestamp request id=0x0149, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Timestamp request id=0x2ead, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Timestamp request id=0x3ce5, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Timestamp request id=0x5de2, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Timestamp request id=0x884d, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Timestamp request id=0xbf35, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Timestamp request id=0x6b44, seq=0/0, ttl=
10.11.35.214	10.10.68.1	ICMP	Timestamp request id=0x1a28, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Timestamp request id=0x8586, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Timestamp request id=0xacce, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Timestamp request id=0x0cfa, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Timestamp request id=0xa39f, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Timestamp request id=0x2279, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Timestamp request id=0x80cf, seq=0/0, ttl=

At the bottom, it shows 'Packets: 1131 · Displayed: 512 (45.3%) Profile: Default'.

Similarly, Nmap sends address mask inquiries (ICMP Type 17) and looks for an address mask response (ICMP Type 18). With the -PM option, you can enable this scan. Live hosts are intended to respond to ICMP address mask requests.

`nmap -PM -sn MACHINE IP/24`

`nmap -PM -sn TARGET`





We use the command `nmap -PM -sn MACHINE IP/24` to try to find live hosts using ICMP address mask queries.

Despite the fact that we know at least eight hosts are online based on previous scans, this scan yielded none.

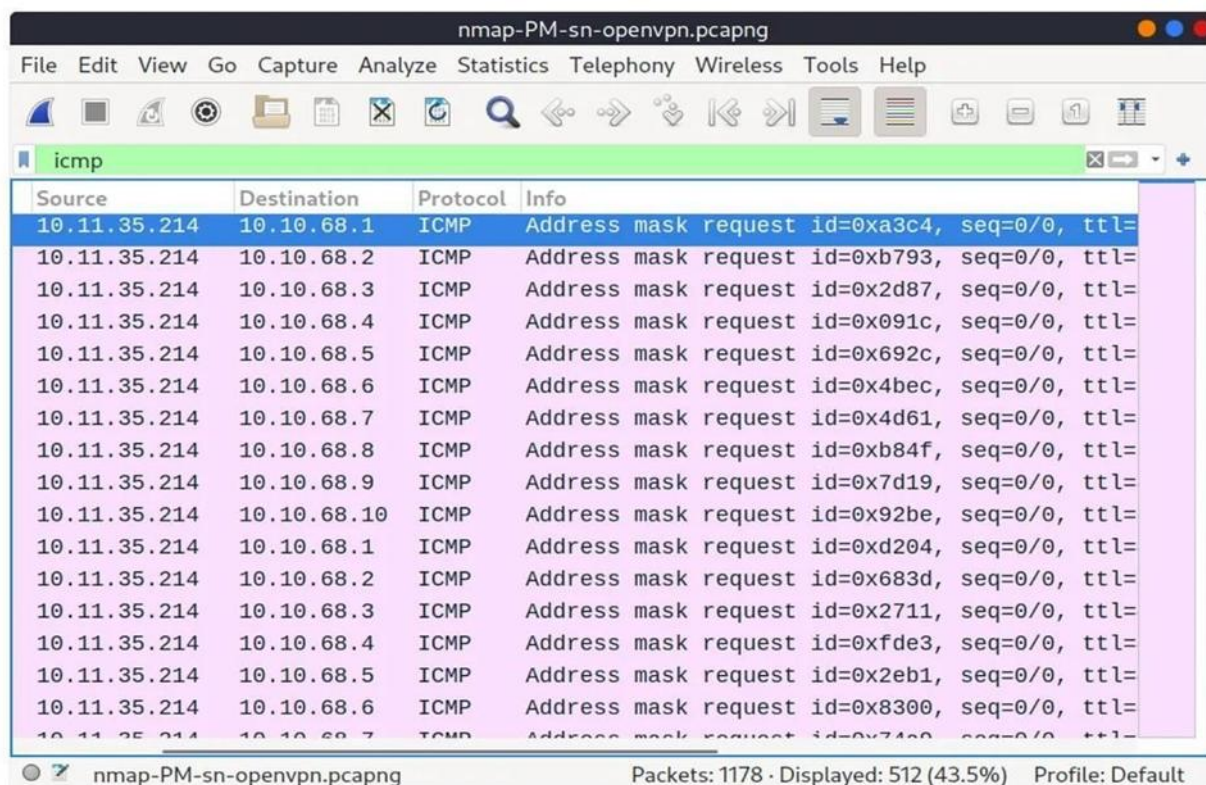
The reason for this is that the target system or a firewall on the route is preventing this sort of ICMP packet from being sent. As a result, learning multiple techniques to attain the same outcome is critical. If one sort of packet is banned, we may always try another to learn about the target network 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 did not receive a response and were unable to determine which hosts are available, it is important to note that this scan sent ICMP address mask queries to every valid IP address and waited for a response.

As shown in the screenshot below, each ICMP request was issued twice.



Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Address mask request id=0xa3c4, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Address mask request id=0xb793, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Address mask request id=0x2d87, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Address mask request id=0x091c, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Address mask request id=0x692c, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Address mask request id=0x4bec, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Address mask request id=0x4d61, seq=0/0, ttl=
10.11.35.214	10.10.68.8	ICMP	Address mask request id=0xb84f, seq=0/0, ttl=
10.11.35.214	10.10.68.9	ICMP	Address mask request id=0x7d19, seq=0/0, ttl=
10.11.35.214	10.10.68.10	ICMP	Address mask request id=0x92be, seq=0/0, ttl=
10.11.35.214	10.10.68.1	ICMP	Address mask request id=0xd204, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Address mask request id=0x683d, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Address mask request id=0x2711, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Address mask request id=0xfde3, seq=0/0, ttl=
10.11.35.214	10.10.68.5	ICMP	Address mask request id=0x2eb1, seq=0/0, ttl=
10.11.35.214	10.10.68.6	ICMP	Address mask request id=0x8300, seq=0/0, ttl=
10.11.35.214	10.10.68.7	ICMP	Address mask request id=0x74a0, seq=0/0, ttl=

[Question 6.1] What is the option required to tell Nmap to use ICMP Timestamp to discover live hosts?

Answer: -PP

[Question 6.2] What is the option required to tell Nmap to use ICMP Address Mask to discover live hosts?

Answer: -PM

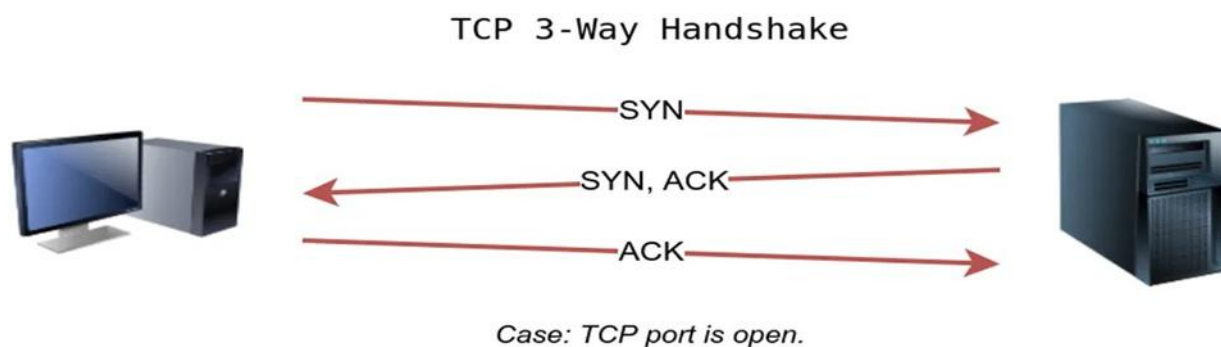
[Question 6.3] What is the option required to tell Nmap to use ICMP Echo to discover live hosts?

Answer: -PE

---

## Task 7 Nmap Host Discovery Using TCP and UDP

**TCP SYN Ping** – We can send a packet to a TCP port, 80 by default, with the SYN (Synchronize) flag set and wait for a response. An open port should receive a SYN/ACK (Acknowledgement); a closed port will receive a RST (Reset). In this situation, we simply check to see if we get a response to determine whether the host is up and running. The specific status of the port is unimportant in this context.

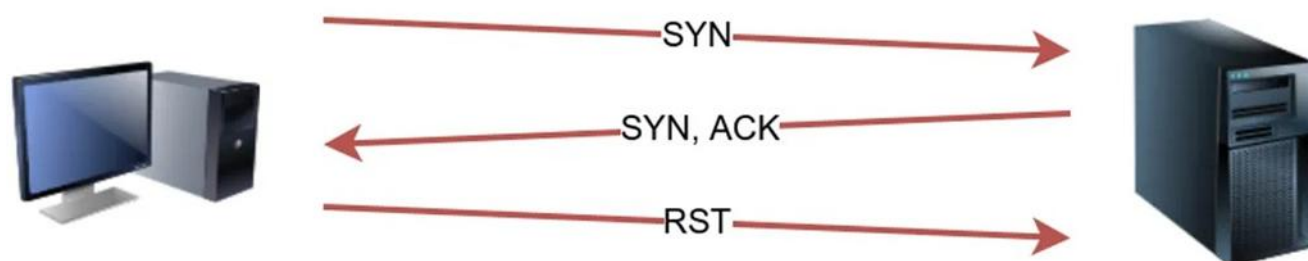


If you want Nmap to use TCP SYN ping, use -PS followed by the port number, range, list, or a combination of these options.

- PS21, for example, targets port 21 (specific)
- PS21-25 targets ports 21, 22, 23, 24, and 25. (range)
- PS80,443,8080 will direct traffic to the three ports 80, 443, and 8080. (direct)

Privileged users (root and sudoers) can transmit TCP SYN packets and do not need to finish the TCP 3-way handshake even if the port is open. If the port is open, unprivileged users are forced to finish the 3-way handshake.

`nmap -PS -sn TARGET`



*Case: TCP port is open.*

To scan the target VM subnet, we shall use:

`nmap -PS -sn MACHINE IP/24`

```
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
```

We were able to find five hosts, as seen in the report below. Remember to add “sudo” to leverage on “root user”.

Let’s check at the network traffic on Wireshark in the figure below to see what happened behind the scenes. Because we did not specify any TCP ports to utilize in the TCP ping scan, Nmap selected common ports; in this example, TCP port 80. Any service listening on port 80 is supposed to respond, implying that the host is up and running.



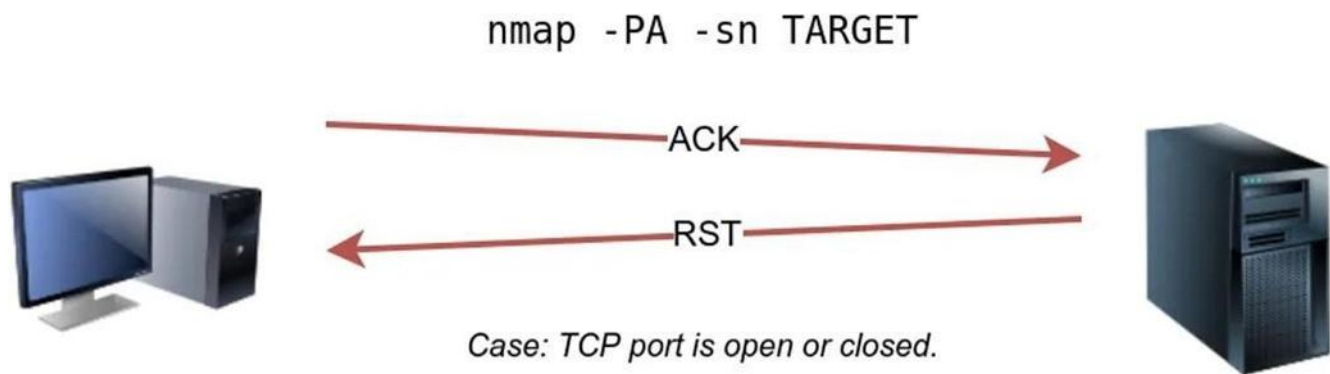
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.2	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.3	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.4	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.5	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.6	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.7	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.8	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.9	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.10	TCP	61429 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.1	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.2	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.3	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.4	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.5	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.6	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.11.35.214	10.10.68.7	TCP	61431 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

OpenVPN Protocol: Protocol      Packets: 1147 · Displayed: 623 (54.3%)      Profile: Default

**TCP ACK Ping** — As you might expect, this transmits a packet with the ACK flag set. To accomplish this, you must be running Nmap as a privileged user. When run as an unprivileged user, Nmap will attempt a three-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 diagram below indicates that any TCP transmission with an ACK flag should receive a TCP packet with a RST flag set. Because the TCP packet with the ACK flag is not part of any current connection, the target answers with the RST flag set. The predicted answer is used to determine whether or not the target host is operational.



To discover the online hosts on the target's subnet, we use:  
`sudo nmap -PA -sn MACHINE IP/24`

```
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
```

The TCP ACK ping scan revealed that five hosts were operational.

Looking at the network data, we can see that several packets have the ACK flag set and are being transmitted to port 80 of the target servers. Each packet is sent twice by Nmap. Systems that do not react are either down or unreachable.

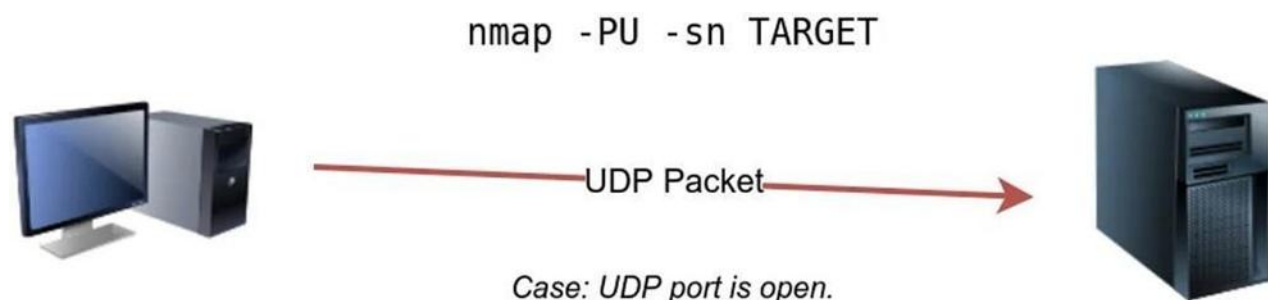


nmap-PA-sn-openvpn.pcapng			
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help			
!openvpn			
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.2	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.3	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.4	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.5	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.6	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.7	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.8	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.9	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.10	TCP	45492 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.1	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.2	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.3	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.4	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.5	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.6	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.7	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
nmap-PA-sn-openvpn.pcapng      Packets: 1079 · Displayed: 557 (51.6%)      Profile: Default			

UDP Ping – Finally, we may utilize UDP to determine whether or not the host is online. In contrast to TCP SYN ping, sending a UDP packet to an open port is unlikely to result in a response.

If we transmit a UDP packet to a closed UDP port, we should expect to receive an ICMP port unreachable packet, indicating that the target system is up and running.

In the picture below, we observe a UDP packet being sent to an open UDP port that receives no response. However, sending a UDP packet to any closed UDP port may result in an indirect answer indicating that the destination is online.



The port syntax is similar to that of TCP SYN and TCP ACK ping; Nmap uses -PU for UDP ping. In the following example, we utilize a UDP scan to find five active 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
```

Examine the UDP packets that were created. In the Wireshark picture below, we see Nmap sending UDP packets to UDP ports that are almost certainly closed. The graphic below illustrates Nmap's usage of an unusual UDP port to cause an ICMP destination unreachable (port unreachable) error.

9/11/25, 9:21 AM

Nmap Live Host Discovery | TryHackMe (THM) | by Aircon | Medium

nmap-PU-sn-openvpn.pcapng

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.2	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.3	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.4	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.5	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.6	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.7	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.8	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.9	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.10	UDP	57190 → 40125 Len=40
10.11.35.214	10.10.68.1	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.2	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.3	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.4	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.5	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.6	UDP	57192 → 40125 Len=40
10.11.35.214	10.10.68.7	UDP	57192 → 40125 Len=40

nmap-PU-sn-openvpn.pcapng      Packets: 1118 · Displayed: 602 (53.8%)      Profile: Default

Masscan – On the other hand, employs a similar approach to discover accessible systems. However, in order to complete its network scan as rapidly as possible, Masscan creates a high number of packets. The syntax is nearly identical: -p can be followed by a port number, list, or range. Consider the following scenarios:

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

[Question 7.1] Which TCP ping scan does not require a privileged account?  
Answer: TCP SYN Ping

[Question 7.2] Which TCP ping scan requires a privileged account?  
Answer: TCP ACK Ping

[Question 7.3] What option do you need to add to Nmap to run a TCP SYN ping scan on the telnet port?

```
-PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
```

Answer: -PS23

---

## Task 8 Using Reverse-DNS Lookup

The default behavior of Nmap is to use reverse-DNS on internet hosts. This can be a useful step because hostnames might disclose a lot.

If you don't wish to submit such DNS queries, use -n to bypass this step.

Nmap will seek up online hosts by default. However, you can use the -R option to query the DNS server even for offline hosts. You can use the --dns-servers DNS\_SERVER option to specify a specific DNS server.

[Question 8.1] We want Nmap to issue a reverse DNS lookup for all the possible hosts on a subnet, hoping to get some insights from the names. What option should we add?

Answer: -R

Task 1	✔	Introduction	▼
Task 2	✔	Subnetworks	📄 ▼
Task 3	✔	Enumerating Targets	▼
Task 4	✔	Discovering Live Hosts	▼
Task 5	✔	Nmap Host Discovery Using ARP	▼
Task 6	✔	Nmap Host Discovery Using ICMP	▼
Task 7	✔	Nmap Host Discovery Using TCP and UDP	▼
Task 8	✔	Using Reverse-DNS Lookup	▼