

Acknowledgment

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

Nmap

Room 1: Nmap Live Host Discovery

Task 1: Introduction

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. 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. The second question about discovering running services is answered in the next Nmap rooms that focus on port-scanning.

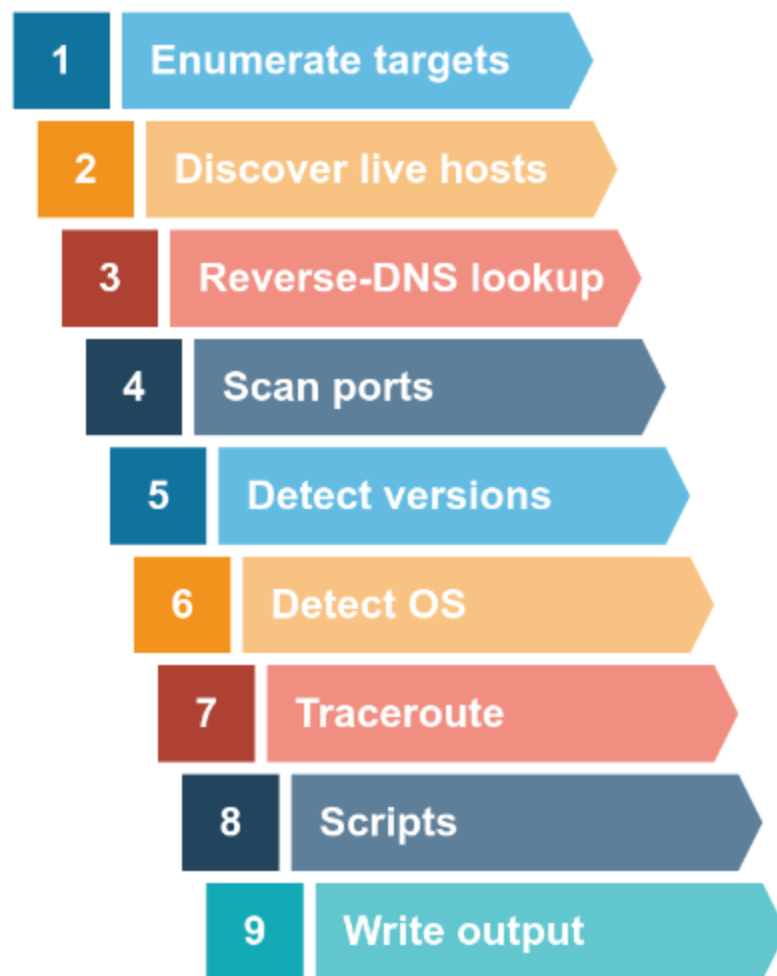
This room explains the steps that Nmap carries out to discover the systems that are online before port-scanning. This stage is crucial because trying to port-scan offline systems will only waste time and create unnecessary noise on the network.

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.

We also introduce two scanners, `arp-scan` and `masscan`, and explain how they overlap with part of Nmap’s host discovery.

As already mentioned, starting with this room, we will use Nmap to discover systems and services actively. Nmap was created by Gordon Lyon (Fyodor), a network security expert and open source programmer. It was released in 1997. Nmap, short for Network Mapper, is free, open-source software released under GPL license. Nmap is an industry-standard tool for mapping networks, identifying live hosts, and discovering running services. Nmap’s scripting engine can further extend its functionality, from fingerprinting services to exploiting vulnerabilities. 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.



Task 2: 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.



he 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.

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.

Click on the “View Site” button to start the network simulator. We will use this simulator to answer the questions in tasks 2, 4, and 5.

Task 3: Enumerating Targets

We mentioned the different *techniques* we can use for scanning in Task 1. Before we explain each in detail and put it into use against a live target, we need to specify the targets we want to scan. 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`.

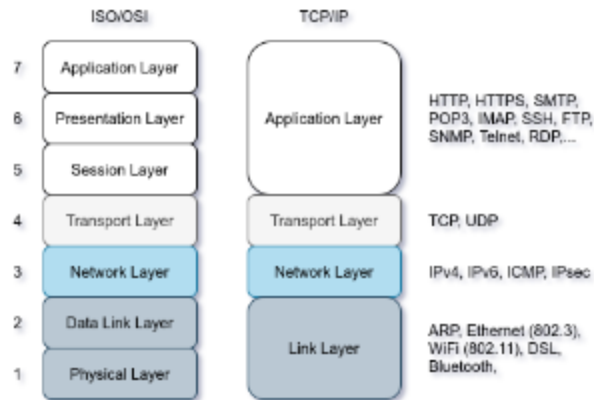
If you want to check the list of hosts that Nmap will scan, you can use `nmap -sL TARGETS`. This option will give you a detailed list of the hosts that Nmap will scan without scanning them; however, Nmap will attempt a reverse-DNS resolution on all the targets to obtain their names. Names might reveal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

Launch the AttackBox using the Start AttackBox button, open the terminal when the AttackBox is ready, and use Nmap to answer the following.

Task 4: 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



Before we discuss how scanners can use each in detail, we will briefly review these four protocols. 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.

If you have closed the network simulator, click on the “View Site” button in Task 2 to display it again.

Task 5: Nmap Host Discovery Using ARP

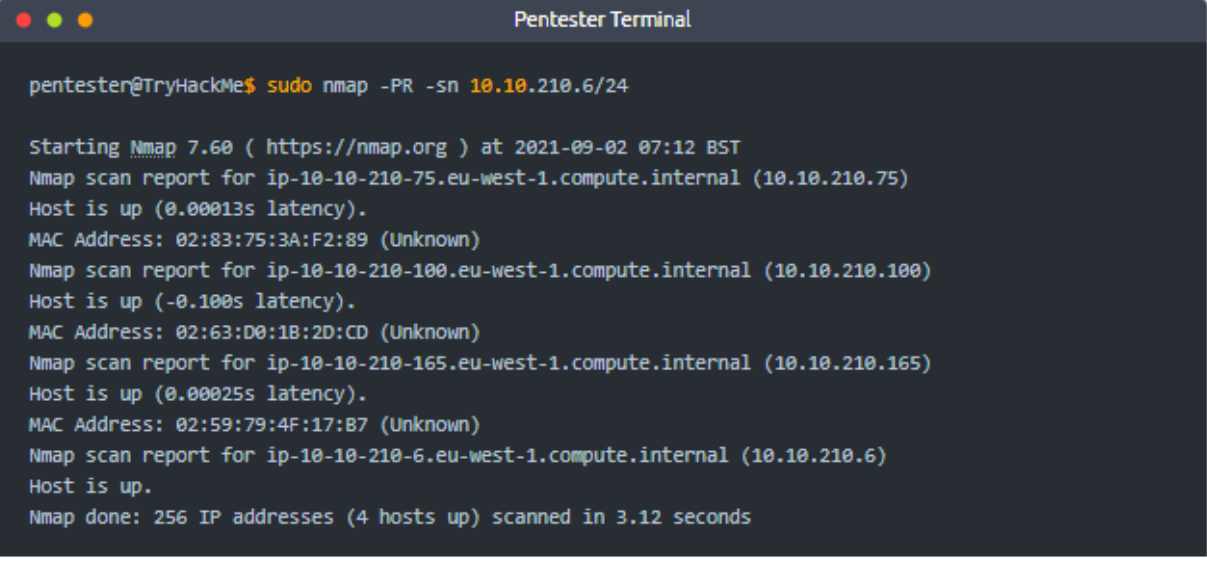
How would you know which hosts are up and running? It is essential to avoid wasting our time port-scanning an offline host or an IP address not in use. There are various ways to discover online hosts. When no host discovery options are provided, Nmap follows the following approaches to discover live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

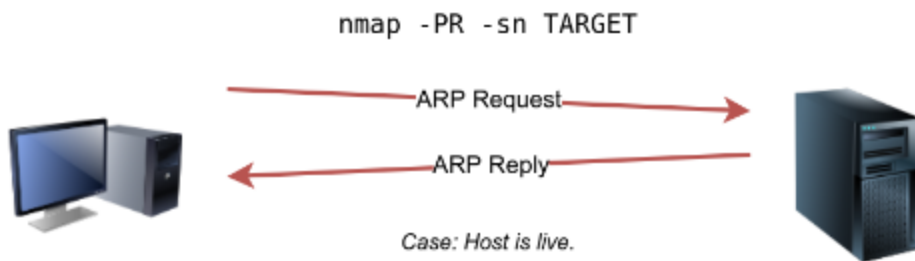


```
Pentester Terminal

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.

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 10.10.210.6? 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 10.10.210.12? 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
02:ba:eb:d6:18:2b	Broadcast	ARP	who has 10.10.210.16? Tell 10.10.210.6

Talking about ARP scans, we should mention a scanner built around ARP queries: `arp-scan`; it provides many options to customize your scan. 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`.

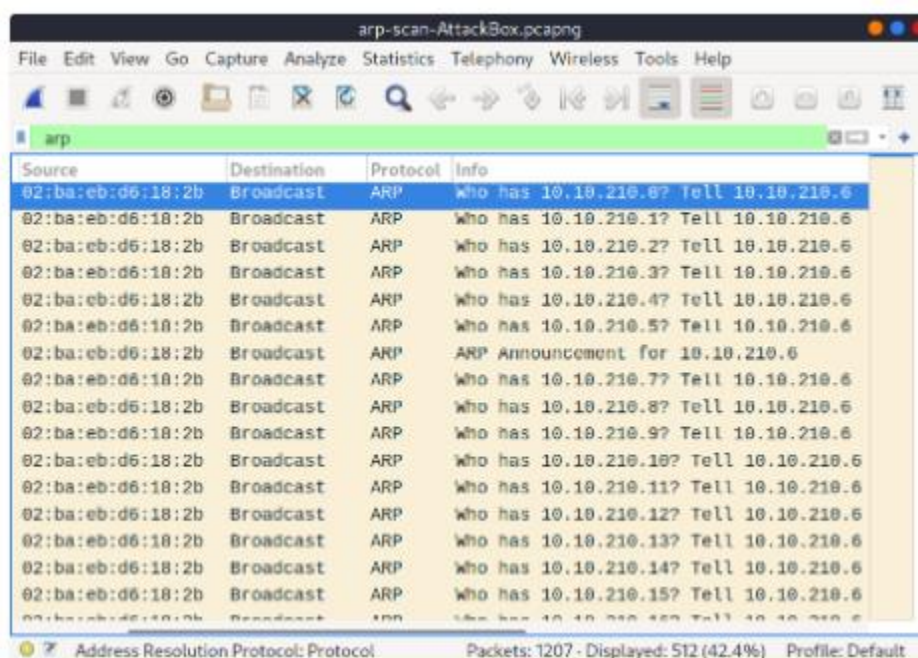
In the example below, we scanned the subnet of the AttackBox using `arp-scan` `ATTACKBOX_IP/24`. Since we ran this scan at a time frame close to the previous one `nmap -PR -sn ATTACKBOX_IP/24`, we obtained the same three live targets.

```
Pentester Terminal

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.

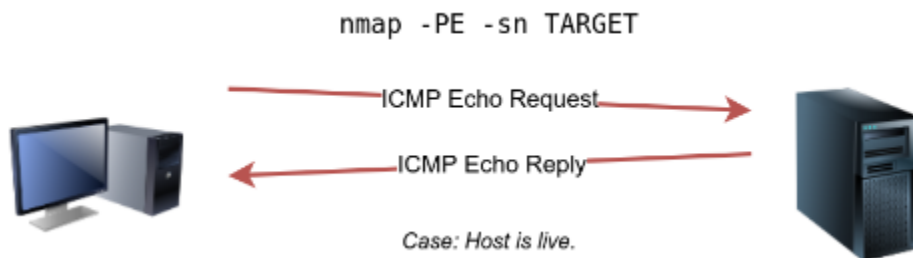


If you have closed the network simulator, click on the "Visit Site" button in Task 2 to display it again.

Task 6: 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` from the AttackBox.

```
Pentester Terminal

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: 02:95:36:71:58:87 (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.

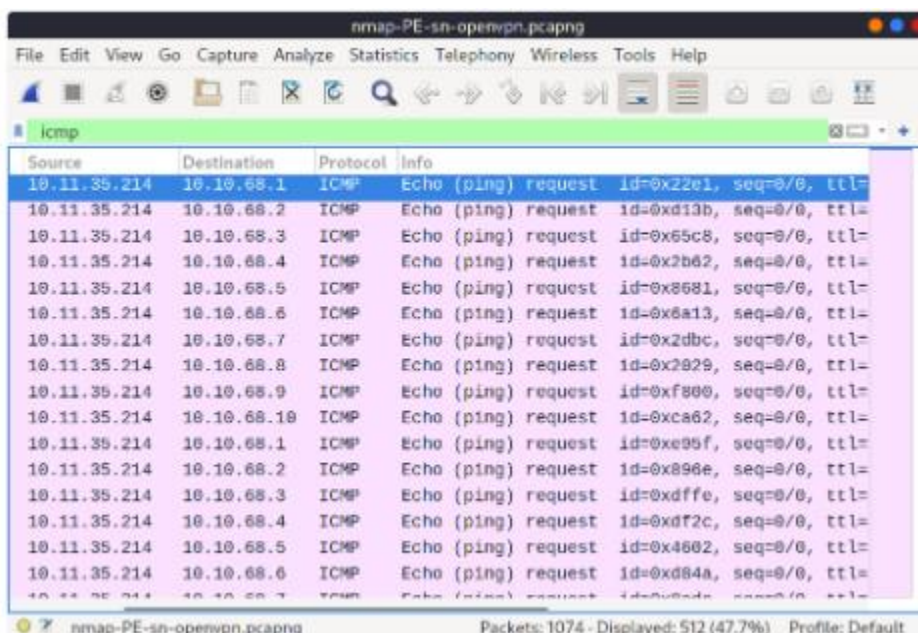
We will repeat the scan above; however, this time, we will scan from a system that belongs to a different subnet. The results are similar but without the MAC addresses.

```
Pentester Terminal

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.

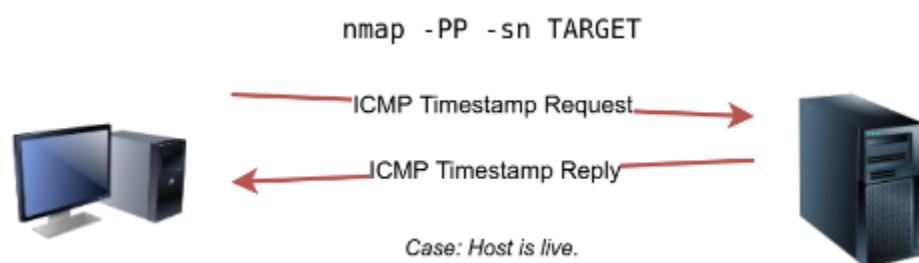


The image shows a Wireshark packet capture window titled 'nmap-PE-sn-openvpn.pcapng'. The 'Filter' bar is set to 'icmp'. The packet list shows 20 ICMP Echo (ping) requests. The source IP for all requests is 10.11.35.214, and the destination IP addresses range from 10.10.68.1 to 10.10.68.10. The packet details pane shows the first packet's structure: ICMP Echo (ping) request, ID=0x22e1, seq=0/0, ttl=64.

Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Echo (ping) request id=0x22e1, seq=0/0, ttl=64
10.11.35.214	10.10.68.2	ICMP	Echo (ping) request id=0xd13b, seq=0/0, ttl=64
10.11.35.214	10.10.68.3	ICMP	Echo (ping) request id=0x65c8, seq=0/0, ttl=64
10.11.35.214	10.10.68.4	ICMP	Echo (ping) request id=0x2b62, seq=0/0, ttl=64
10.11.35.214	10.10.68.5	ICMP	Echo (ping) request id=0x8681, seq=0/0, ttl=64
10.11.35.214	10.10.68.6	ICMP	Echo (ping) request id=0x6a13, seq=0/0, ttl=64
10.11.35.214	10.10.68.7	ICMP	Echo (ping) request id=0x2dbc, seq=0/0, ttl=64
10.11.35.214	10.10.68.8	ICMP	Echo (ping) request id=0x2029, seq=0/0, ttl=64
10.11.35.214	10.10.68.9	ICMP	Echo (ping) request id=0xf800, seq=0/0, ttl=64
10.11.35.214	10.10.68.10	ICMP	Echo (ping) request id=0xca62, seq=0/0, ttl=64
10.11.35.214	10.10.68.1	ICMP	Echo (ping) request id=0xe95f, seq=0/0, ttl=64
10.11.35.214	10.10.68.2	ICMP	Echo (ping) request id=0x896e, seq=0/0, ttl=64
10.11.35.214	10.10.68.3	ICMP	Echo (ping) request id=0xdffe, seq=0/0, ttl=64
10.11.35.214	10.10.68.4	ICMP	Echo (ping) request id=0xd72c, seq=0/0, ttl=64
10.11.35.214	10.10.68.5	ICMP	Echo (ping) request id=0x4602, seq=0/0, ttl=64
10.11.35.214	10.10.68.6	ICMP	Echo (ping) request id=0xd84a, seq=0/0, ttl=64
10.11.35.214	10.10.68.7	ICMP	Echo (ping) request id=0x0000, seq=0/0, ttl=64

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 Terminal

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
```

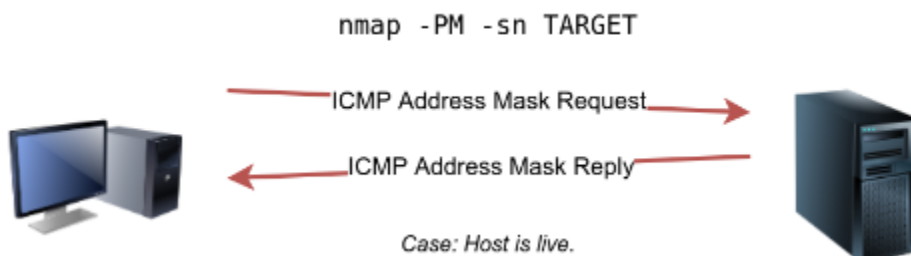
Similar to the previous ICMP scan, this scan will send many ICMP timestamp requests to every valid IP address in the target subnet. In the Wireshark screenshot below, you can see one source IP address sending ICMP packets to every possible IP address to discover online hosts.

Wireshark packet capture showing ICMP Timestamp requests. The table below represents the data visible in the packet list pane.

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=0xbf3b, 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=0xcfa, 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=0x000f, seq=0/0, ttl=

Wireshark status bar: Packets: 1131 - Displayed: 512 (45.3%) Profile: Default

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.

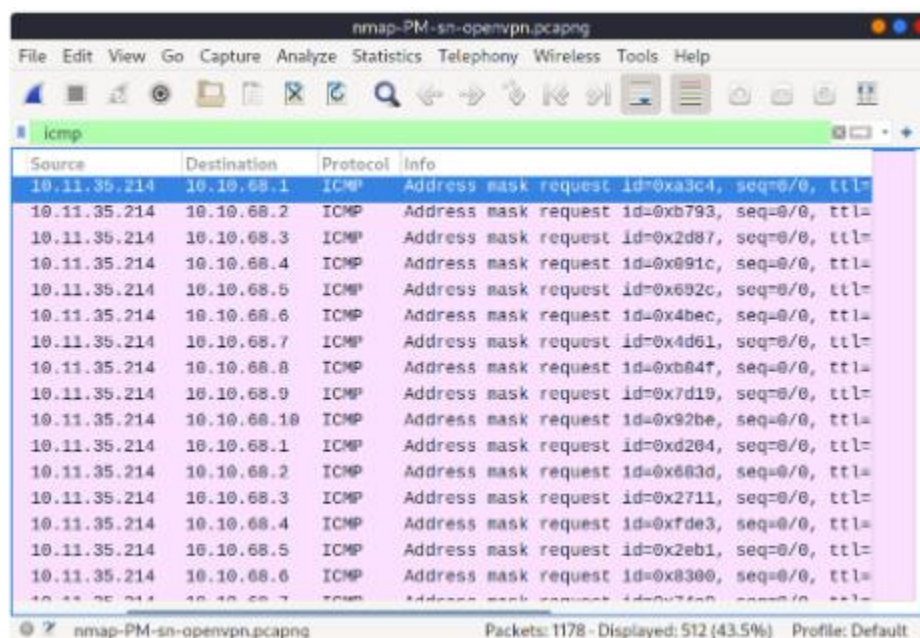
```

Pentester Terminal

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.



The screenshot shows a Wireshark capture of ICMP traffic. The packet list table is as follows:

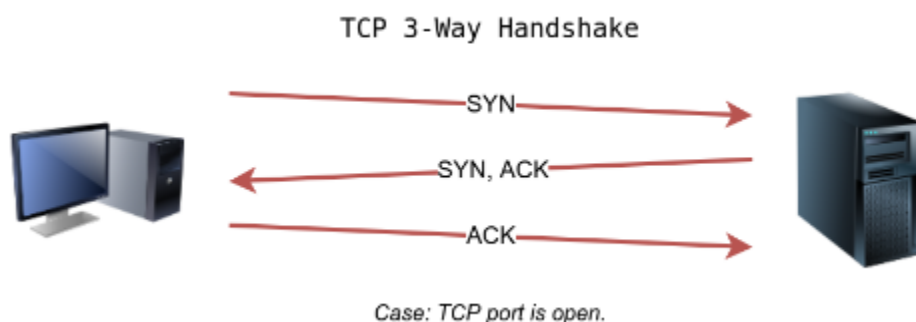
Source	Destination	Protocol	Info
10.11.35.214	10.10.68.1	ICMP	Address mask request id=0xa3c4, seq=0/0, ttl=
10.11.35.214	10.10.68.2	ICMP	Address mask request id=0xb793, seq=0/0, ttl=
10.11.35.214	10.10.68.3	ICMP	Address mask request id=0x2d87, seq=0/0, ttl=
10.11.35.214	10.10.68.4	ICMP	Address mask request id=0x891c, 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=0x7f00, seq=0/0, ttl=

At the bottom, it says: Packets: 1178 - Displayed: 512 (43.5%) Profile: Default

Task 7: Nmap Host Discovery Using TCP and UDP

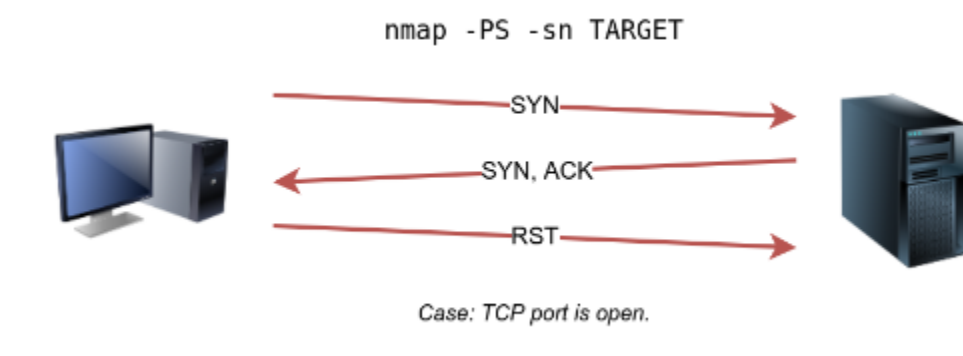
TCP SYN Ping

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.



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.



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 Terminal

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.

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

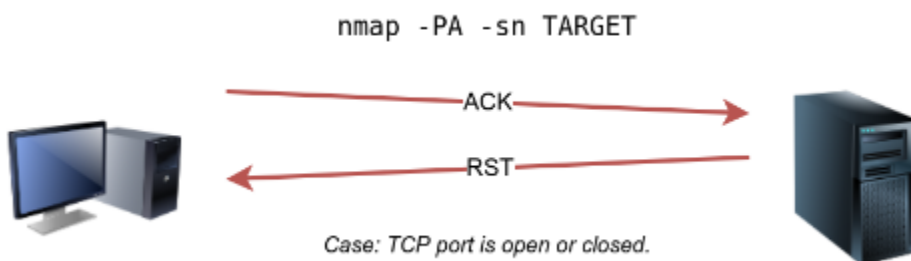
OpenVPN Protocol: Protocol Packets: 1147 - Displayed: 623 (54.3%) Profile: Default

TCP ACK Ping

As you 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 Terminal

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
```

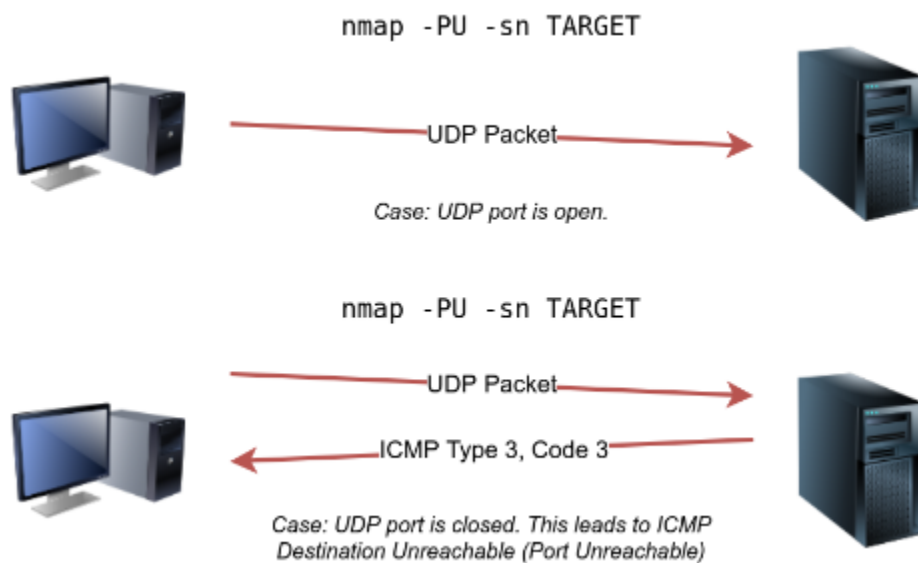
If we peek at the network traffic as shown in the figure below, we will discover many packets with the ACK flag set and sent to port 80 of the target systems. Nmap sends each packet twice. The systems that don't respond are offline or inaccessible.

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	10.10.68.11	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.12	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.13	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.14	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.15	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.16	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
10.11.35.214	10.10.68.17	TCP	45494 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0

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.

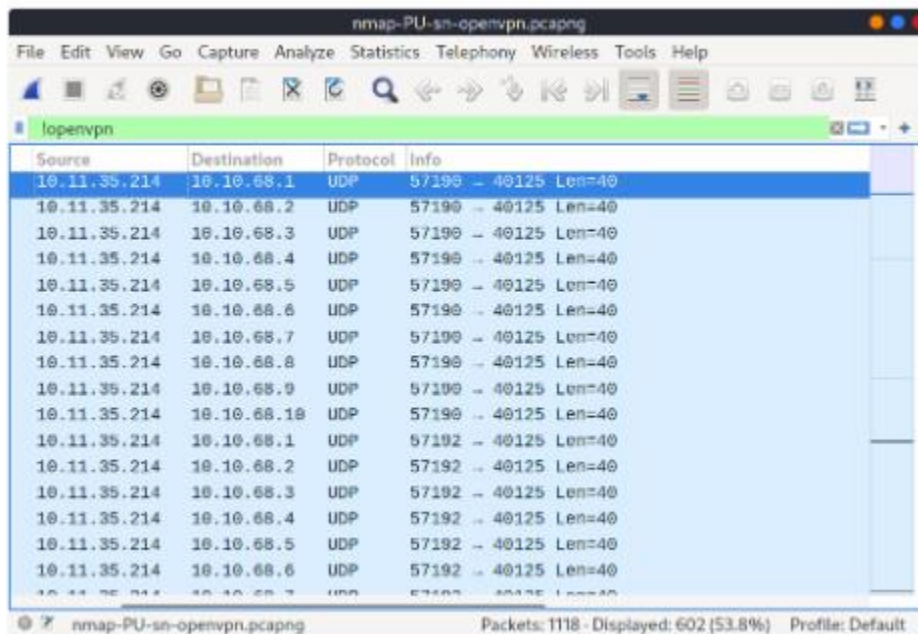


The syntax to specify the ports is similar to that of TCP SYN ping and TCP ACK ping; Nmap uses `-PU` for UDP ping. In the following example, we use a UDP scan, and we discover five live hosts.

```
Pentester Terminal

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.



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	10.10.68.11	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.12	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.13	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.14	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.15	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.16	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.17	UDP	57192 -> 40125 Len=40
10.11.35.214	10.10.68.18	UDP	57192 -> 40125 Len=40

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 is not installed on the AttackBox; however, it can be installed using `apt install masscan`.

Task 8: Using 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.

Task 9: Summary

You have learned how ARP, ICMP, TCP, and UDP can detect live hosts by completing this room. 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	<code>sudo nmap -PR -sn MACHINE_IP/24</code>
ICMP Echo Scan	<code>sudo nmap -PE -sn MACHINE_IP/24</code>
ICMP Timestamp Scan	<code>sudo nmap -PP -sn MACHINE_IP/24</code>
ICMP Address Mask Scan	<code>sudo nmap -PM -sn MACHINE_IP/24</code>
TCP SYN Ping Scan	<code>sudo nmap -PS22,80,443 -sn MACHINE_IP/30</code>
TCP ACK Ping Scan	<code>sudo nmap -PA22,80,443 -sn MACHINE_IP/30</code>
UDP Ping Scan	<code>sudo nmap -PU53,161,162 -sn MACHINE_IP/30</code>

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
<code>-n</code>	no DNS lookup
<code>-R</code>	reverse-DNS lookup for all hosts
<code>-sn</code>	host discovery only

Room 2: Nmap Basic Port Scans

Task 1: Introduction

In the previous room, we focused on discovering online systems. 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, in this room and the next one, we focus on port scanning and the different types of port scans used by `nmap`. This room explains:

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

Moreover, we discuss the different options to specify the ports, the scan rate, and the number of parallel probes.

Answer the questions below

Launch the AttackBox by using the Start AttackBox button. You will launch different types of scans against the target VM to gain a solid knowledge of Nmap basic scan types.

Task 2: 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.

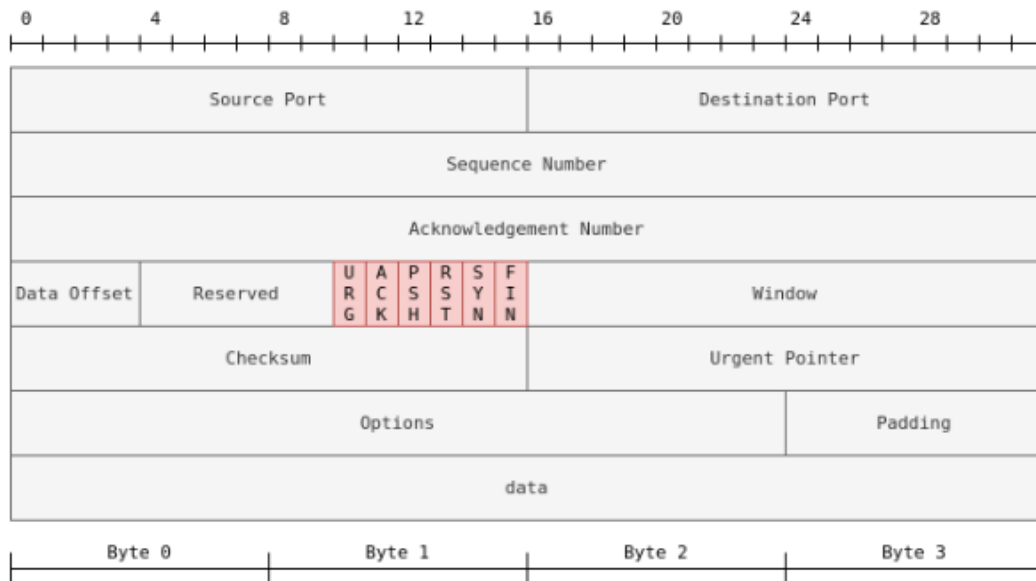
However, in practical situations, we need to consider the impact of firewalls. For instance, a port might be open, but a firewall might be blocking the packets. Therefore, Nmap considers the following six states:

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.

Task 3: 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 as defined in [RFC 793](#). This figure looks sophisticated at first; however, it is pretty simple to understand. 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.

TCP Header (RFC793)



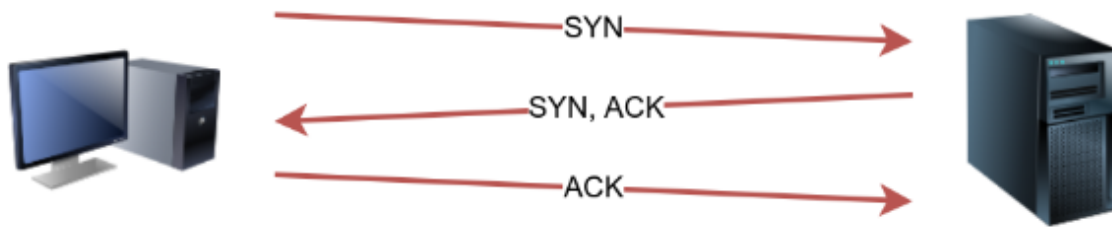
In particular, we need to focus on the flags that Nmap can set or unset. We have highlighted the TCP flags in red. Setting a flag bit means setting its value to 1. From left to right, the TCP header flags are:

1. **URG:** Urgent flag indicates that the urgent pointer field 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.

Task 4: 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.

TCP 3-Way Handshake



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`.

`nmap -sT TARGET`



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.

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	40900 → 256 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	49446 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	34466 → 143 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	50188 → 8888 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	36154 → 5900 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	57134 → 445 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	40246 → 113 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	55014 → 111 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	39962 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.113.174	10.10.252.27	TCP	54696 → 53 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
10.10.252.27	10.10.113.174	TCP	443 → 49446 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.174	TCP	256 → 40900 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.174	TCP	143 → 34466 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0
10.10.113.174	10.10.252.27	TCP	34466 → 143 [ACK] Seq=1 Ack=1 Win=62848 Len=0
10.10.252.27	10.10.113.174	TCP	8888 → 50188 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.174	TCP	5900 → 36154 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.174	TCP	445 → 57134 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

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.

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

Below (TCP connect scan) the following command example returned a detailed list of the open ports.

To illustrate the `-sT` (TCP connect scan), the following command example returned a detailed list of the open ports.

```
Pentester Terminal

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)

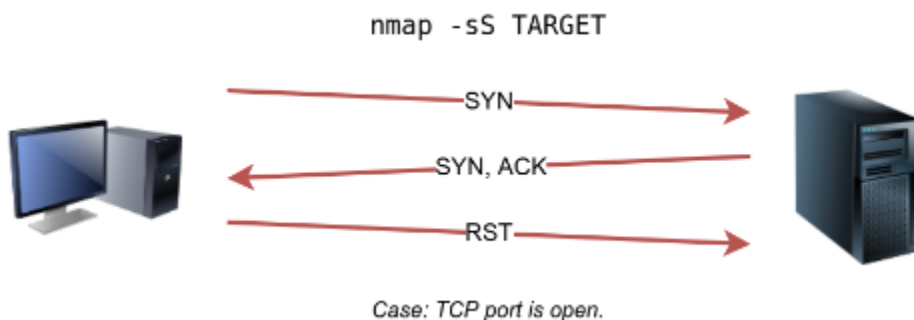
Nmap done: 1 IP address (1 host up) scanned in 0.40 seconds
```

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.

Task 5: 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.



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.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==10.10.252.27

Source	Destination	Protocol	Info
10.10.113.174	10.10.252.27	TCP	46095 → 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 554 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 100 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.113.174	10.10.252.27	TCP	46095 → 3309 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10.10.252.27	10.10.113.1	TCP	1720 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.1	TCP	25 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174	10.10.252.27	TCP	46095 → 25 [RST] Seq=1 Win=0 Len=0
10.10.252.27	10.10.113.1	TCP	5900 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.10.252.27	10.10.113.1	TCP	80 → 46095 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0
10.10.113.174	10.10.252.27	TCP	46095 → 80 [RST] Seq=1 Win=0 Len=0
10.10.252.27	10.10.113.1	TCP	135 → 46095 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

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

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.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

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

nmap-sS-AttackBox.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

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

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 Terminal

pentester@TryHackMe$ sudo nmap -sS 10.10.19.165

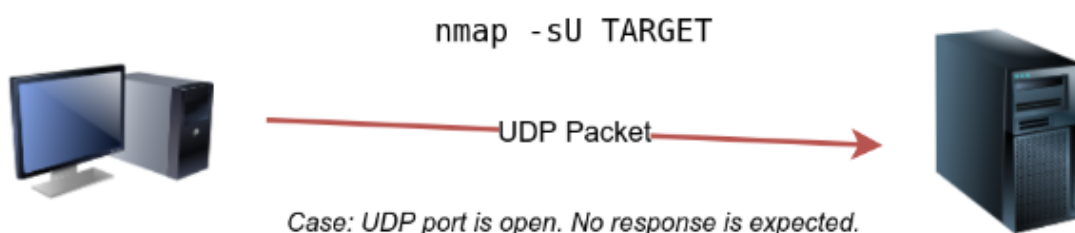
Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:53 BST
Nmap scan report for 10.10.19.165
Host is up (0.0073s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open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

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

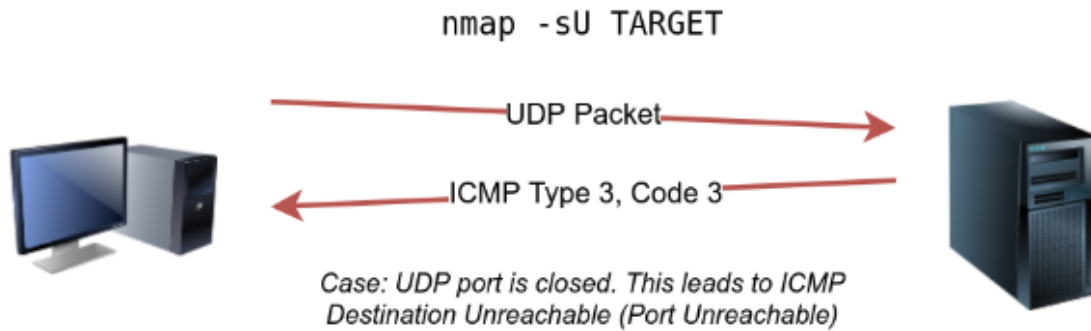
Task 6: UDP Scan

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).

nmap-sU-AttackBox.pcapng

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

nmap-sU-AttackBox.pcapng Packets: 87775 · Displayed: 2580 (2.9%) Profile: Default

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 Terminal

pentester@TryHackMe$ sudo nmap -sU 10.10.19.165

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:54 BST
Nmap scan report for 10.10.19.165
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
```

Task 7: 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 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.

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

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.

Task 8: Summary

This room covered three types of scans.

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

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

Room 3: Nmap Advanced Port Scans

Task 1: Introduction

Task2: TCP Null Scan, Full Scan and Xmas Scan

Task3: TCP Maimon Scan

Task4: TCP ACK, Window, Custom Scan

Task5: Spoofing and Decoys

Task6: Fragmented Packets

Task7: Idle/Zombie Scan

Task8: Getting More Details

Task9: Summary

Room 4: Nmap Post Port Scans

Task 1: Introduction

Task2: TCP Null Scan, Full Scan and Xmas Scan

Task3: TCP Maimon Scan

Task4: TCP ACK, Window, Custom Scan

Task5: Spoofing and Decoys

Task6: Fragmented Packets

Task7: Idle/Zombie Scan

Task8: Getting More Details

Task9: Summary

