

# Nmap: The Basics

---

## Learned:

- Nmap is used to scan devices on a network for issues, whether security-related or for troubleshooting.
- Scanning IP addresses can be done with many options, such as a range using the `-` symbol `192.168.1.1-50` or `-sn` used for ping scans.
- Port scanning can also be done using `-sT` for viewing completed TCP three-way handshakes or `-sU` for UDP scanning.
- You can choose how fast to scan targets by setting a minimum or maximum packet rate, or by using the timing templates `-T0` through `-T5`, which range from 0 (slow) to 5 (fast).

## Host Discovery: Who Is Online

Nmap uses multiple ways to specify its targets:

- IP range using `-`: If you want to scan all the IP addresses from 192.168.0.1 to 192.168.0.10, you can write `192.168.0.1-10`
- IP subnet using `/`: If you want to scan a subnet, you can express it as `192.168.0.1/24`, and this would be equivalent to `192.168.0.0-255`
- Hostname: You can also specify your target by hostname, for example, `example.thm`

Let's say you want to discover the online hosts on a network. Nmap offers the `-sn` option, i.e., ping scan.

We are either running `nmap` as `root` or using `sudo` because we don't want to restrict Nmap's abilities with our account privileges. Running Nmap as a local (non-root) user would limit us to fundamental types of scans such as ICMP echo and TCP connect scans.

## Scanning a "Local" Network

In this context, we use the term "local" to refer to the network we are directly connected to, such as an Ethernet or WiFi network. In the first demonstration, we will scan the WiFi network to which we are connected. Our IP address is `192.168.66.89`, and we are scanning the `192.168.66.0/24` network. The `nmap -sn 192.168.66.0/24` command and its output are shown in the terminal below.

AttackBox Terminal

```
root@tryhackme:~# nmap -sn 192.168.66.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2024-08-07 13:49 EEST
Nmap scan report for XiaoQiang (192.168.66.1)
```

```
Host is up (0.0069s latency).
MAC Address: 44:DF:65:D8:FE:6C (Unknown)
Nmap scan report for S190023240007 (192.168.66.88)
Host is up (0.090s latency).
MAC Address: 7C:DF:A1:D3:8C:5C (Espressif)
Nmap scan report for wlan0 (192.168.66.97)
Host is up (0.20s latency).
MAC Address: 10:D5:61:E2:18:E6 (Tuya Smart)
Nmap scan report for 192.168.66.179
Host is up (0.10s latency).
MAC Address: E4:AA:EC:8F:88:C9 (Tianjin Hualai Technology)
[...]
Nmap done: 256 IP addresses (7 hosts up) scanned in 2.64 seconds
```

Because we are scanning the local network, where we are connected via Ethernet or WiFi, we can look up the MAC addresses of the devices. Consequently, we can figure out the network card vendors, which is beneficial information as it can help us guess the type of target device(s).

When scanning a directly connected network, Nmap starts by sending ARP requests. When a device responds to the ARP request, Nmap labels it with “Host is up”.

## Scanning a “Remote” Network

Consider the case of a “remote” network. In this context, “remote” means that at least one router separates our system from this network. As a result, all our traffic to the target systems must go through one or more routers. Unlike scanning a local network, we cannot send an ARP request to the target.

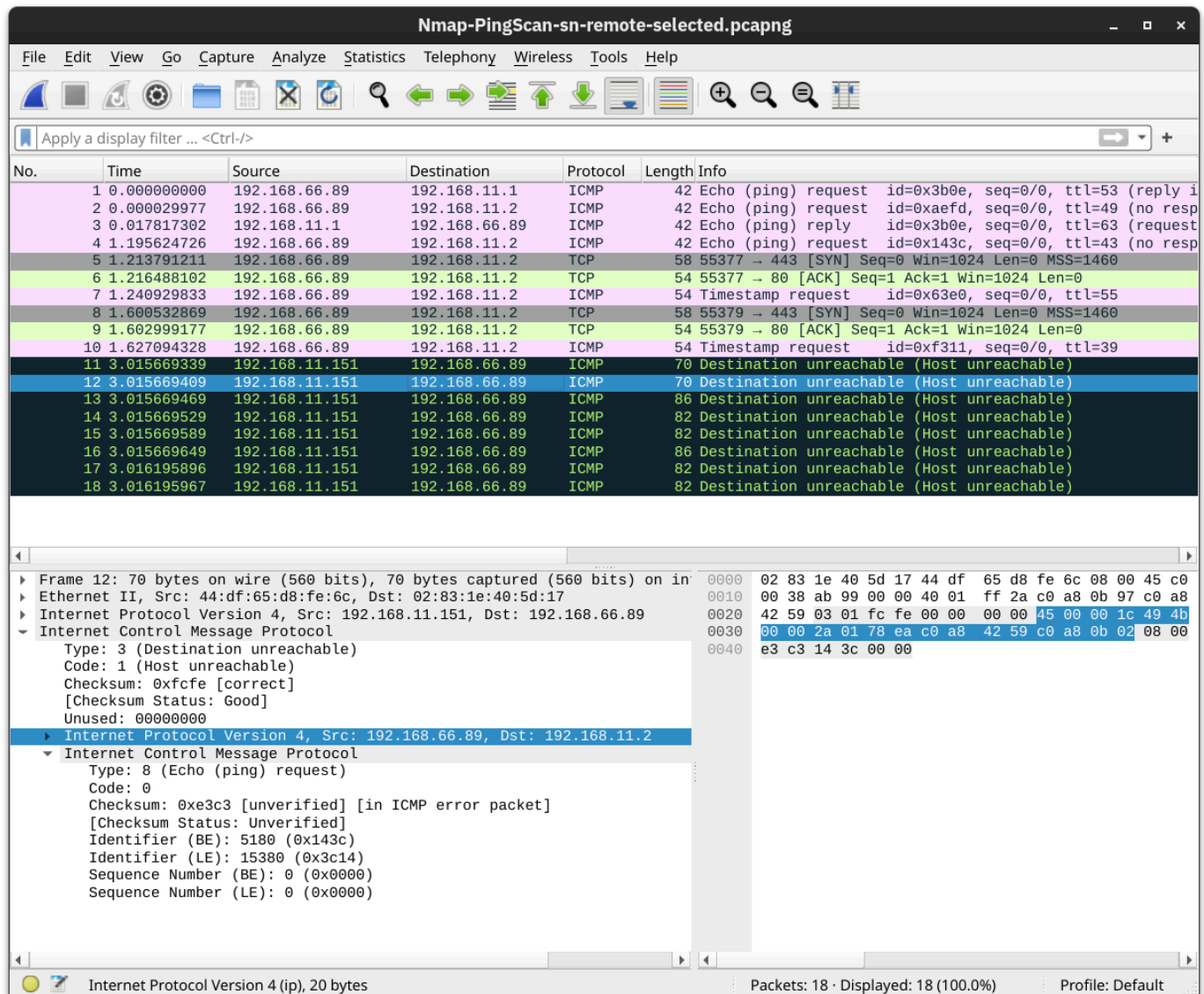
Our system has the IP address `192.168.66.89` and belongs to the `192.168.66.0/24` network. In the terminal below we scan the target network `192.168.11.0/24` where there are two or more routers (hops) separate our local system from the target hosts.

AttackBox Terminal

```
root@tryhackme:~# nmap -sn 192.168.11.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2024-08-07 14:05 EEST
Nmap scan report for 192.168.11.1
Host is up (0.018s latency).
Nmap scan report for 192.168.11.151
Host is up (0.0013s latency).
Nmap scan report for 192.168.11.152
Host is up (0.13s latency).
Nmap scan report for 192.168.11.154
Host is up (0.22s latency).
Nmap scan report for 192.168.11.155
```

Host is up (2.3s latency).

Nmap done: 256 IP addresses (5 hosts up) scanned in 10.67 seconds



## Scanning TCP Ports

The easiest and most basic way to know whether a TCP port is open would be to attempt to `telnet` to the port. If you are inclined to scan with a Telnet client, try to establish a TCP connection with every target port. In other words, you attempt to complete the TCP three-way handshake with every target port; however, only open TCP ports would respond appropriately and allow a TCP connection to be established. This procedure is not very different from Nmap's connect scan.

### Connect Scan

The connect scan can be triggered using `-sT`. It tries to complete the TCP three-way handshake with every target TCP port. If the TCP port turns out to be open and Nmap connects successfully, Nmap will tear down the established connection.

In the screenshot below, our scanning machine has the IP address `192.168.124.148` and the target system has TCP port 22 open and port 23 closed. In the part marked with 1, you can see how the TCP

three-way handshake was completed and later torn down with a TCP RST-ACK packet by Nmap. The part marked with 2 shows a connection attempt to a closed port, and the target system responded with a TCP RST-ACK packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	76	47984 → 22 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK
2	0.000185092	192.168.124.211	192.168.124.148	TCP	76	22 → 47984 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
3	0.000206693	192.168.124.148	192.168.124.211	TCP	68	47984 → 22 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=
4	0.000241360	192.168.124.148	192.168.124.211	TCP	68	47984 → 22 [RST, ACK] Seq=1 Ack=1 Win=32128 Len=0
5	0.400055798	192.168.124.148	192.168.124.211	TCP	76	51792 → 23 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK
6	0.400240119	192.168.124.211	192.168.124.148	TCP	56	23 → 51792 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

## SYN Scan (Stealth)

Unlike the connect scan, which tries to **connect** to the target TCP port, i.e., complete a three-way handshake, the SYN scan only executes the first step: it sends a TCP SYN packet. Consequently, the TCP three-way handshake is never completed. The advantage is that this is expected to lead to fewer logs as the connection is never established, and hence, it is considered a relatively stealthy scan. You can select the SYN scan using the `-sS` flag.

In the screenshot below, we scan the same system with port 22 open. The part marked with 1 shows the listening service replying with a TCP SYN-ACK packet. However, Nmap responded with a TCP RST packet instead of completing the TCP three-way handshake. The part marked with 2 shows a TCP connection attempt to a closed port. In this case, the packet exchange is the same as in the connect scan.

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	60	56186 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2	0.000225431	192.168.124.211	192.168.124.148	TCP	60	22 → 56186 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
3	0.000247493	192.168.124.148	192.168.124.211	TCP	56	56186 → 22 [RST] Seq=1 Win=0 Len=0
4	0.400493120	192.168.124.148	192.168.124.211	TCP	60	56186 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.400734380	192.168.124.211	192.168.124.148	TCP	56	23 → 56186 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

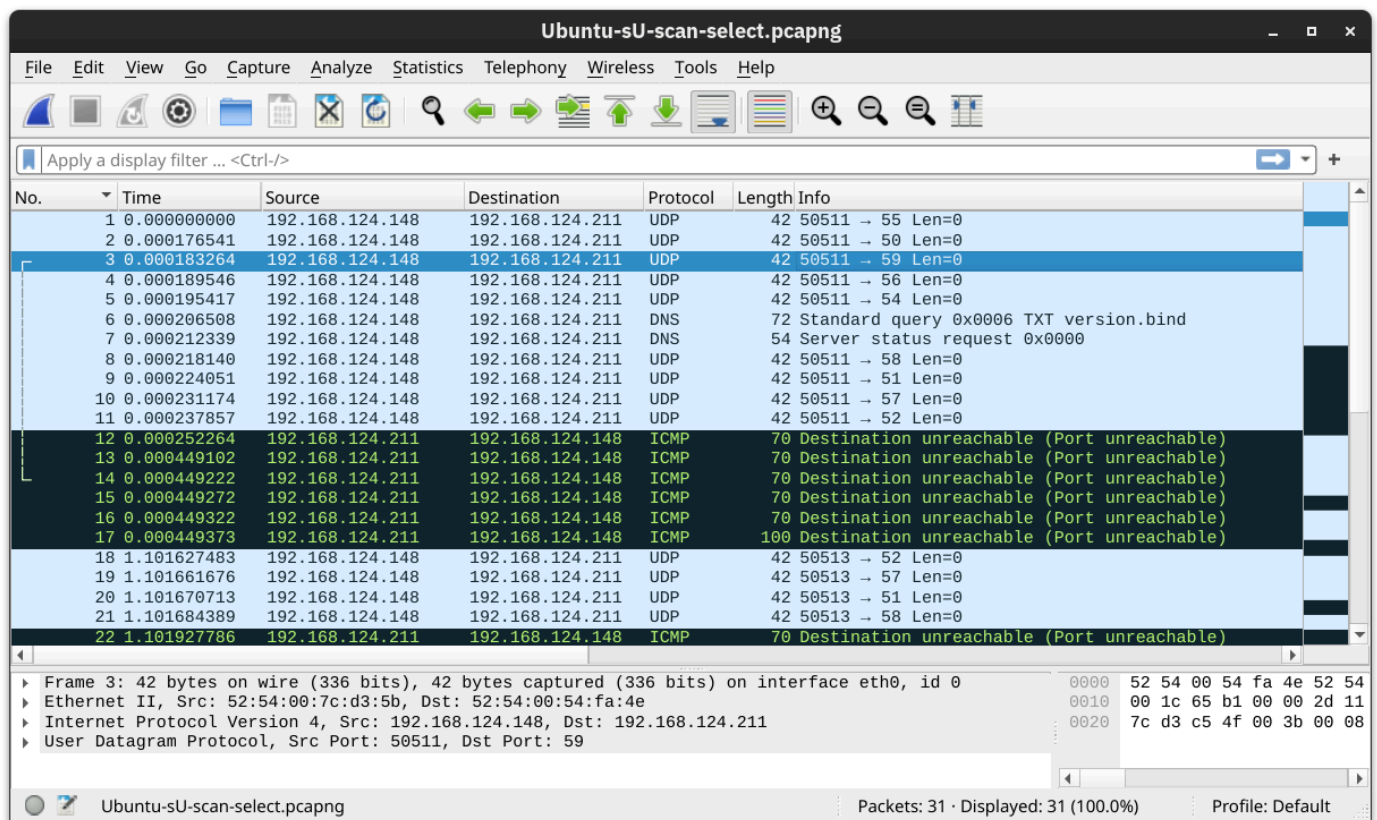
Ubuntu-sS-scan-select-order.pcapng

Packets: 5 · Displayed: 5 (100.0%) Profile: Default

## Scanning UDP Ports

Although most services use TCP for communication, many use UDP. Examples include DNS, DHCP, NTP (Network Time Protocol), SNMP (Simple Network Management Protocol), and VoIP (Voice over IP). UDP does not require establishing a connection and tearing it down afterwards. Furthermore, it is very suitable for real-time communication, such as live broadcasts.

Nmap offers the option `-sU` to scan for UDP services. Because UDP is simpler than TCP, we expect the traffic to differ. The screenshot below shows several ICMP destination unreachable (port unreachable) responses as Nmap sends UDP packets to closed UDP ports.



Nmap scans the most common 1,000 ports by default. However, this might not be what you are looking for. Therefore, Nmap offers you a few more options.

- `-F` is for Fast mode, which scans the 100 most common ports (instead of the default 1000).
- `-p[range]` allows you to specify a range of ports to scan. For example, `-p10-1024` scans from port 10 to port 1024, while `-p-25` will scan all the ports between 1 and 25. Note that `-p-` scans all the ports and is equivalent to `-p1-65535` and is the best option if you want to be as thorough as possible.

## Summary

Option	Explanation
<code>-sT</code>	TCP connect scan – complete three-way handshake
<code>-sS</code>	TCP SYN – only first step of the three-way handshake
<code>-sU</code>	UDP scan
<code>-F</code>	Fast mode – scans the 100 most common ports
<code>-p[range]</code>	Specifies a range of port numbers – <code>-p-</code> scans all the ports

## OS Detection

You can enable OS detection by adding the `-O` option. As the name implies, the OS detection option triggers Nmap to rely on various indicators to make an educated guess about the target OS. In this case, it is detecting the target has Linux 4.x or 5.x running. That's actually true. However, there is no perfectly accurate OS detector. The statement that it is between 4.15 and 5.8 is very close as the target host's OS is 5.15.

#### AttackBox Terminal

```
root@tryhackme:~# nmap -sS -O 192.168.124.211
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-13 13:37 EEST
Nmap scan report for ubuntu22lts-vm (192.168.124.211)
Host is up (0.00043s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 52:54:00:54:FA:4E (QEMU virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.44 seconds
```

### Service and Version Detection

You discovered several open ports and want to know what services are listening on them. `-sV` enables version detection. This is very convenient for gathering more information about your target with fewer keystrokes. The terminal output below shows an additional column called "VERSION", indicating the detected SSH server version.

#### AttackBox Terminal

```
root@tryhackme:~# nmap -sS -sV 192.168.124.211
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-13 13:33 EEST
Nmap scan report for ubuntu22lts-vm (192.168.124.211)
Host is up (0.000046s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
MAC Address: 52:54:00:54:FA:4E (QEMU virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```



Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 0.25

What if you can have both `-O`, `-sV` and some more in one option? That would be `-A`. This option enables OS detection, version scanning, and traceroute, among other things.

## Forcing the Scan

When we run our port scan, such as using `-sS`, there is a possibility that the target host does not reply during the host discovery phase (e.g. a host doesn't reply to ICMP requests). Consequently, Nmap will mark this host as down and won't launch a port scan against it. We can ask Nmap to treat all hosts as online and port scan every host, including those that didn't respond during the host discovery phase. This choice can be triggered by adding the `-Pn` option.

## Summary

Option	Explanation
<code>-O</code>	OS detection
<code>-sV</code>	Service and version detection
<code>-A</code>	OS detection, version detection, and other additions
<code>-Pn</code>	Scan hosts that appear to be down

Nmap provides various options to control the scan speed and timing.

Running your scan at its normal speed might trigger an IDS or other security solutions. It is reasonable to control how fast a scan should go. Nmap gives you six timing templates, and the names say it all: paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4), and insane (5). You can pick the timing template by its name or number. For example, you can add `-T0` (or `-T 0`) or `-T paranoid` to opt for the slowest timing.

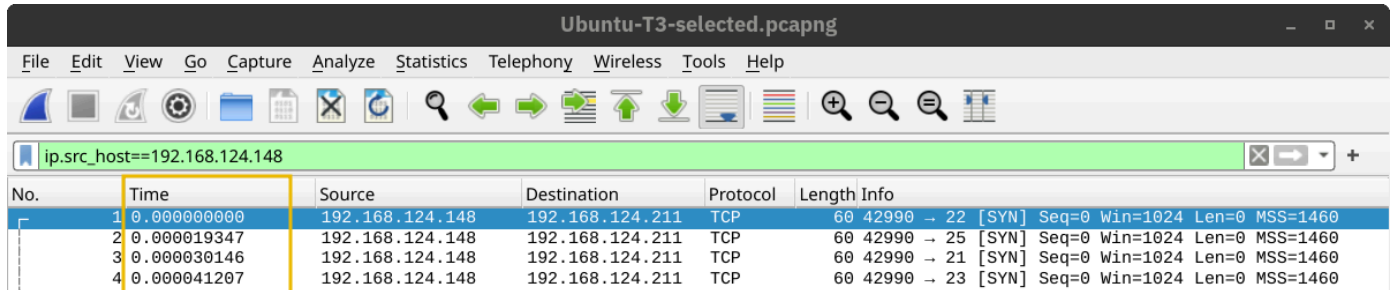
In the Nmap scans below, we launch a SYN scan targeting the 100 most common TCP ports, `nmap -sS 10.201.116.94 -F`. We repeated the scan with different timings: T0, T1, T2, T3, and T4. In our lab setup, Nmap took different amounts of time to scan the 100 ports. The table below should give you an idea, but you will get different results depending on the network setup and target system.

Timing	Total Duration
T0 (paranoid)	9.8 hours
T1 (sneaky)	27.53 minutes



Timing	Total Duration
T2 (polite)	40.56 seconds
T3 (normal)	0.15 seconds
T4 (aggressive)	0.13 seconds

In the following screenshots, we can see the time when Nmap sent the different packets. In this screenshot below, with the scan timing being **T0**, we can see that Nmap waited 5 minutes before moving to the next port.



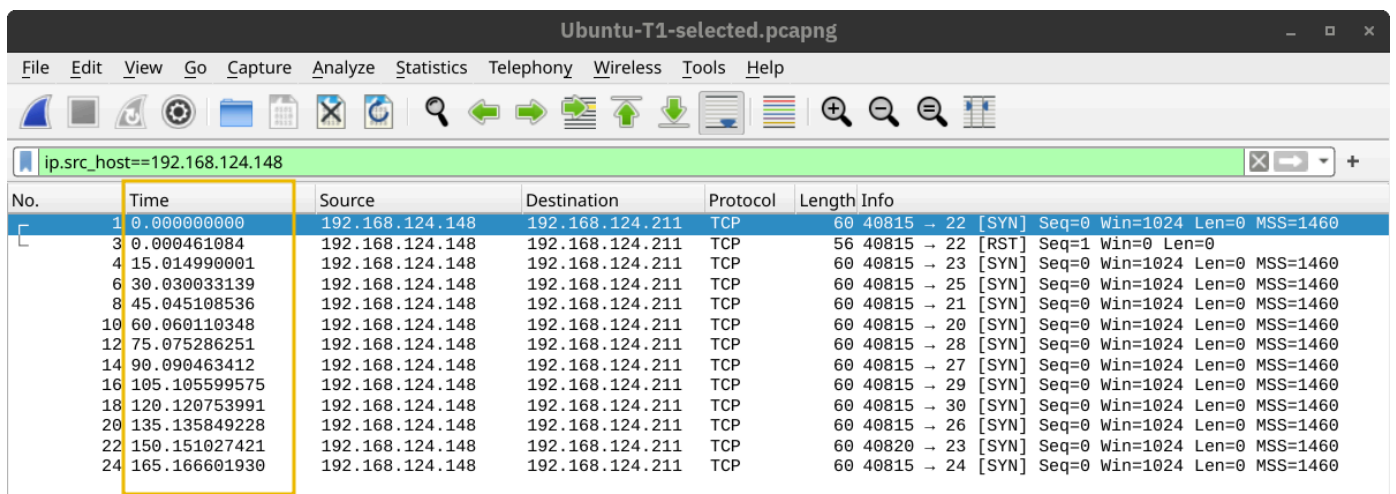
Ubuntu-T3-selected.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src\_host==192.168.124.148

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	60	42990 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2	0.000019347	192.168.124.148	192.168.124.211	TCP	60	42990 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000030146	192.168.124.148	192.168.124.211	TCP	60	42990 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000041207	192.168.124.148	192.168.124.211	TCP	60	42990 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

In the screenshot below, Nmap waited 15 seconds between every two ports when we set the timing to **T1**.



Ubuntu-T1-selected.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.src\_host==192.168.124.148

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	60	40815 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000461084	192.168.124.148	192.168.124.211	TCP	56	40815 → 22 [RST] Seq=1 Win=0 Len=0
4	15.014990001	192.168.124.148	192.168.124.211	TCP	60	40815 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	30.030033139	192.168.124.148	192.168.124.211	TCP	60	40815 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	45.045108536	192.168.124.148	192.168.124.211	TCP	60	40815 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10	60.060110348	192.168.124.148	192.168.124.211	TCP	60	40815 → 20 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	75.075286251	192.168.124.148	192.168.124.211	TCP	60	40815 → 28 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
14	90.090463412	192.168.124.148	192.168.124.211	TCP	60	40815 → 27 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	105.105599575	192.168.124.148	192.168.124.211	TCP	60	40815 → 29 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	120.120753991	192.168.124.148	192.168.124.211	TCP	60	40815 → 30 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	135.135849228	192.168.124.148	192.168.124.211	TCP	60	40815 → 26 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
22	150.151027421	192.168.124.148	192.168.124.211	TCP	60	40820 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
24	165.166601930	192.168.124.148	192.168.124.211	TCP	60	40815 → 24 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Then, the waiting dropped to 0.4 seconds for **T2** as shown below.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	60	62177 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.400617264	192.168.124.148	192.168.124.211	TCP	60	62177 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.801103901	192.168.124.148	192.168.124.211	TCP	60	62177 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	1.201808968	192.168.124.148	192.168.124.211	TCP	60	62177 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	1.202321728	192.168.124.148	192.168.124.211	TCP	56	62177 → 22 [RST] Seq=1 Win=0 Len=0
10	1.602305554	192.168.124.148	192.168.124.211	TCP	60	62177 → 20 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	2.003089587	192.168.124.148	192.168.124.211	TCP	60	62177 → 28 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
14	2.403667835	192.168.124.148	192.168.124.211	TCP	60	62177 → 26 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	2.804154051	192.168.124.148	192.168.124.211	TCP	60	62177 → 27 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	3.204791920	192.168.124.148	192.168.124.211	TCP	60	62177 → 29 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	3.605332658	192.168.124.148	192.168.124.211	TCP	60	62177 → 24 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
22	4.005888721	192.168.124.148	192.168.124.211	TCP	60	62177 → 30 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Finally, in the default case, `T3`, Nmap appeared to be running as fast as it could, as shown below. It is worth repeating that this would look different on a different lab setup. However, in this particular case, Nmap considered the connection to the target to be fast and reliable, as no packet loss was incurred.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.124.148	192.168.124.211	TCP	60	42990 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2	0.000019347	192.168.124.148	192.168.124.211	TCP	60	42990 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000030146	192.168.124.148	192.168.124.211	TCP	60	42990 → 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000041207	192.168.124.148	192.168.124.211	TCP	60	42990 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.000051466	192.168.124.148	192.168.124.211	TCP	60	42990 → 24 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	0.000059321	192.168.124.148	192.168.124.211	TCP	60	42990 → 30 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	0.000067897	192.168.124.148	192.168.124.211	TCP	60	42990 → 28 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	0.000077335	192.168.124.148	192.168.124.211	TCP	60	42990 → 26 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	0.000086001	192.168.124.148	192.168.124.211	TCP	60	42990 → 29 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10	0.000094277	192.168.124.148	192.168.124.211	TCP	60	42990 → 27 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
13	0.000215864	192.168.124.148	192.168.124.211	TCP	56	42990 → 22 [RST] Seq=1 Win=0 Len=0
22	0.000319799	192.168.124.148	192.168.124.211	TCP	60	42990 → 20 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

A second helpful option is the number of parallel service probes. The number of parallel probes can be controlled with `--min-parallelism <numprobes>` and `--max-parallelism <numprobes>`. These options can be used to set a minimum and maximum on the number of TCP and UDP port probes active simultaneously for a host group. By default, `nmap` will automatically control the number of parallel probes. If the network is performing poorly, i.e., dropping packets, the number of parallel probes might fall to one; furthermore, if the network performs flawlessly, the number of parallel probes can reach several hundred.

A similar helpful option is the `--min-rate <number>` and `--max-rate <number>`. As the names indicate, they can control the minimum and maximum rates at which `nmap` sends packets. The rate is provided as the *number of packets per second*. It is worth mentioning that the specified rate applies to the whole scan and not to a single host.

The last option we will cover in this task is `--host-timeout <time>`. This option specifies the maximum time you are willing to wait, and it is suitable for slow hosts or hosts with slow network connections.

Option	Explanation
<code>-T&lt;0-5&gt;</code>	Timing template – paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4), and insane (5)
<code>--min-parallelism &lt;numprobes&gt;</code> and <code>--max-parallelism &lt;numprobes&gt;</code>	Minimum and maximum number of parallel probes
<code>--min-rate &lt;number&gt;</code> and <code>--max-rate &lt;number&gt;</code>	Minimum and maximum rate (packets/second)
<code>--host-timeout</code>	Maximum amount of time to wait for a target host

## Verbosity and Debugging

In some cases, the scan takes a very long time to finish or to produce any output that will be displayed on the screen. Furthermore, sometimes you might be interested in more real-time information about the scan progress. The best way to get more updates about what's happening is to enable verbose output by adding `-v`.

### AttackBox Terminal

```
root@tryhackme:~# nmap -sS 192.168.139.1/24
Starting Nmap 7.92 ( https://nmap.org ) at 2024-08-13 18:57 EEST
Nmap scan report for 192.168.139.254
Host is up (0.000030s latency).
All 1000 scanned ports on 192.168.139.254 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:E0:FC:AE (VMware)

Nmap scan report for g5000 (192.168.139.1)
Host is up (0.000010s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
902/tcp   open  iss-realsecure

Nmap done: 256 IP addresses (2 hosts up) scanned in 41.84 seconds
```

Then, we repeated the same scan; however, the second time, we used the `-v` option for verbosity. The amount of details present below can be very useful, especially when you are learning about Nmap and exploring the different options. In the terminal output below, we can see how Nmap is moving from one stage to another: ARP ping scan, parallel DNS resolution, and finally, SYN stealth scan for every live host.

### AttackBox Terminal

```
root@tryhackme:~# nmap 192.168.139.1/24 -v
Starting Nmap 7.92 ( https://nmap.org ) at 2024-08-13 19:01 EEST
Initiating ARP Ping Scan at 19:01
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 19:01, 7.94s elapsed (255 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:01
Completed Parallel DNS resolution of 1 host. at 19:02, 13.00s elapsed
Nmap scan report for 192.168.139.0 [host down]
Nmap scan report for 192.168.139.2 [host down]
[...]
Nmap scan report for 192.168.139.253 [host down]
Nmap scan report for 192.168.139.255 [host down]
Initiating Parallel DNS resolution of 1 host. at 19:02
Completed Parallel DNS resolution of 1 host. at 19:02, 0.05s elapsed
Initiating SYN Stealth Scan at 19:02
Scanning 192.168.139.254 [1000 ports]
[...]
Initiating SYN Stealth Scan at 19:02
Scanning g5000 (192.168.139.1) [1000 ports]
Discovered open port 902/tcp on 192.168.139.1
Completed SYN Stealth Scan at 19:02, 0.03s elapsed (1000 total ports)
Nmap scan report for g5000 (192.168.139.1)
Host is up (0.0000090s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
902/tcp   open  iss-realsecure

Read data files from: /usr/bin/./share/nmap
Nmap done: 256 IP addresses (2 hosts up) scanned in 42.19 seconds
Raw packets sent: 3512 (146.336KB) | Rcvd: 2005 (84.156KB)
```

Most likely, the `-v` option is more than enough for verbose output; however, if you are still unsatisfied, you can increase the verbosity level by adding another “v” such as `-vv` or even `-vvvv`. You can also specify the verbosity level directly, for example, `-v2` and `-v4`. You can even increase the verbosity level by pressing “v” after the scan already started.

If all this verbosity does not satisfy your needs, you must consider the `-d` for debugging-level output. Similarly, you can increase the debugging level by adding one or more “d” or by specifying the debugging level directly. The maximum level is `-d9`; before choosing that, make sure you are ready for thousands of information and debugging lines.

## Saving Scan Report

In many cases, we would need to save the scan results. Nmap gives us various formats. The three most useful are normal (human-friendly) output, XML output, and grepable output, in reference to the `grep` command. You can select the scan report format as follows:

- `-oN <filename>` - Normal output
- `-oX <filename>` - XML output
- `-oG <filename>` - `grep`-able output (useful for `grep` and `awk`)
- `-oA <basename>` - Output in all major formats

In the terminal below, we can see an example of using the `-oA` option. It resulted in three reports with the extensions `nmap`, `xml`, and `gnmap` for normal, XML, and `grep`-able output.

AttackBox Terminal

```
root@tryhackme:~# nmap -sS 192.168.139.1 -oA gateway
Starting Nmap 7.92 ( https://nmap.org ) at 2024-08-13 19:35 EEST
Nmap scan report for g5000 (192.168.139.1)
Host is up (0.0000070s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
902/tcp   open  iss-realsecure

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
# ls
gateway.gnmap  gateway.nmap  gateway.xml
```

All Options

Option	Explanation
<code>-sL</code>	List scan – list targets without scanning
<b>Host Discovery</b>	
<code>-sn</code>	Ping scan – host discovery only
<b>Port Scanning</b>	
<code>-sT</code>	TCP connect scan – complete three-way handshake
<code>-sS</code>	TCP SYN – only first step of the three-way handshake
<code>-sU</code>	UDP Scan
<code>-F</code>	Fast mode – scans the 100 most common ports

Option	Explanation
<code>-p[range]</code>	Specifies a range of port numbers – <code>-p-</code> scans all the ports
<code>-Pn</code>	Treat all hosts as online – scan hosts that appear to be down
<b>Service Detection</b>	
<code>-O</code>	OS detection
<code>-sV</code>	Service version detection
<code>-A</code>	OS detection, version detection, and other additions
<b>Timing</b>	
<code>-T&lt;0-5&gt;</code>	Timing template – paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4), and insane (5)
<code>--min-parallelism &lt;numprobes&gt;</code> and <code>-max-parallelism &lt;numprobes&gt;</code>	Minimum and maximum number of parallel probes
<code>--min-rate &lt;number&gt;</code> and <code>--max-rate &lt;number&gt;</code>	Minimum and maximum rate (packets/second)
<code>--host-timeout</code>	Maximum amount of time to wait for a target host
<b>Real-time output</b>	
<code>-v</code>	Verbosity level – for example, <code>-vv</code> and <code>-v4</code>
<code>-d</code>	Debugging level – for example <code>-d</code> and <code>-d9</code>
<b>Report</b>	
<code>-oN &lt;filename&gt;</code>	Normal output
<code>-oX &lt;filename&gt;</code>	XML output
<code>-oG &lt;filename&gt;</code>	<code>grep</code> -able output
<code>-oA &lt;basename&gt;</code>	Output in all major formats