Practical Guide for Using Nmap

Live Host Discovery

When trying to attack or test a network, the first step is to **identify which systems (computers or devices) are online** and later **figure out what services (like websites or file sharing) they are running**.

To help with this, we use a tool called **Nmap**.

Why Discover Live Hosts First?

Before scanning ports, it's important to know which devices are **actually online**.
Scanning offline devices is a waste of time and can create **unnecessary traffic** on the network (called "noise").

How Nmap Finds Live Hosts

Nmap uses several methods to detect which devices are active:

- **ARP Scan** – Sends ARP requests (used in local networks) to check if devices respond.
- **ICMP Scan** – Sends ping-like messages (like the ping command) to check for replies.
- **TCP/UDP Ping Scan** – Sends packets to specific TCP or UDP ports to see if there's a response.

Other Tools You'll Learn About

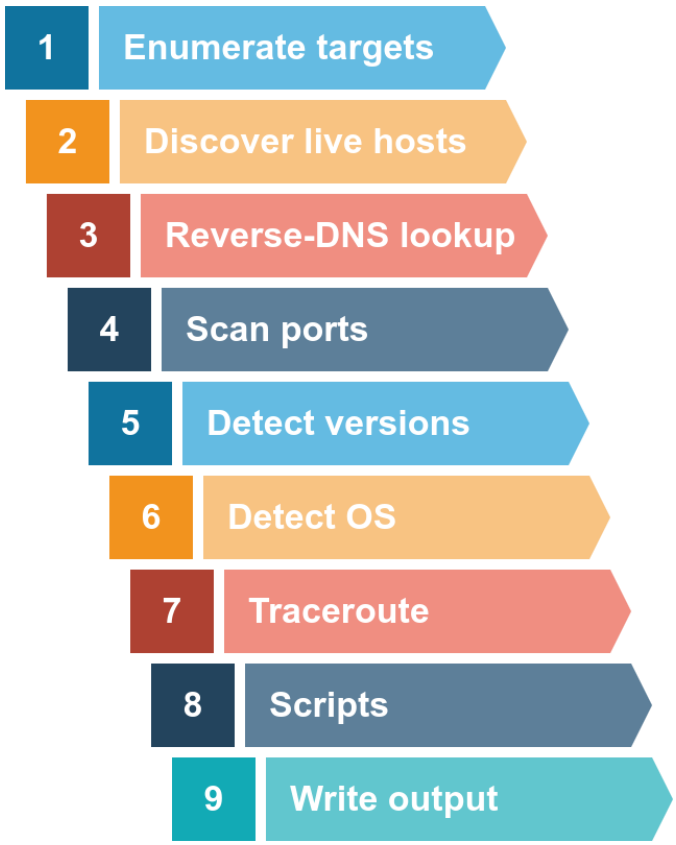
- **arp-scan** – Specialized tool for ARP scanning.
- **masscan** – A very fast scanner that can find live hosts (and ports).

These tools do some of the same things as Nmap.

About Nmap

- **Created by**: Gordon Lyon (nickname: Fyodor)
- **Released in**: 1997
- **Free and open source** under the GPL license
- **Stands for**: Network Mapper
- **Used for**:
    - Finding live systems
    - Scanning open ports
    - Detecting services
    - Running scripts to do more (like identifying software or finding vulnerabilities)

Nmap is a widely-used and powerful tool in cybersecurity.



🔁 Network Segment vs. Subnetwork

- **Network Segment** = A **physical group** of computers connected through something like an **Ethernet switch** or **WiFi access point**.
- **Subnetwork (or Subnet)** = A **logical group** of computers that **share the same IP address range** and are usually connected through a **router**.

👉 In most cases, a **subnet** includes one or more **network segments**.

🔍 Why Subnets Matter in Reconnaissance

When scanning for live devices (recon), you might want to scan a **whole subnet**.
Here's how that works:

📍 If you're connected to the same subnet:

- Your system can use **ARP (Address Resolution Protocol)** to find other devices.
- ARP asks: "Who has this IP? Tell me your MAC address!"
- If a device replies, it means it's online.

🚫 But if you're scanning a different subnet:

- Your scanner sends packets through your **default gateway (router)**.
- **ARP does not work** across routers – it's **only for local subnet use**.
- So, ARP won't help you find devices in other subnets.

🎯 How to Tell Nmap What to Scan

Before using any scanning technique, you need to **tell Nmap what targets to scan**. There are a few different ways to do this:

✅ Ways to Specify Targets

1. **List of Targets**

nmap MACHINE_IP scanme.nmap.org example.com

This will scan **3 hosts**: an IP address and two domain names.

2. **Range of IPs**

nmap 10.11.12.15-20

This will scan:

10.11.12.15

10.11.12.16

10.11.12.17

10.11.12.18

10.11.12.19

10.11.12.20

✅ **6 total IPs**

3. **Subnet**

nmap MACHINE_IP/30

This will scan a subnet of 4 IP addresses, based on CIDR notation.

4. **From a File**

nmap -iL list_of_hosts.txt

This tells Nmap to read a **list of targets from a file**.

🔍 Preview Targets Without Scanning

If you just want to see what Nmap **would scan**, but without actually sending any probes:

nmap -sL TARGETS

- This **lists** all IPs Nmap would scan
- Nmap will try to do **reverse DNS lookups** on each IP (to get hostnames)

🧠 Goal: Find Which Hosts Are Online (Live Hosts)

To figure out which computers (hosts) are **currently active on a network**, we can use different protocols from the **TCP/IP model**.

📚 Which Protocols Can Help?

From **different layers** of the network stack, we can use:

1. **ARP** – Link Layer
2. **ICMP** – Network Layer
3. **TCP** – Transport Layer
4. **UDP** – Transport Layer

🔎 Quick Review of Each Protocol

🔹 1. ARP (Address Resolution Protocol)

- Works **only in local subnets** (cannot cross routers).
- Asks: “Who has this IP address? Tell me your MAC address.”
- If a host replies, it’s **definitely online**.
- Used by default before any communication in the same subnet.

🔹 **2. ICMP (Internet Control Message Protocol)**

- Used by the ping command.
- Sends:
  - **Type 8** = Echo Request
  - **Type 0** = Echo Reply
- If a host replies, it's live.
- Some systems **block ICMP**, so this method isn't always reliable.

🔁 **Note**: If you're pinging a system on the **same subnet**, your computer usually sends an **ARP request first** to get the MAC address.

🔹 **3. TCP**

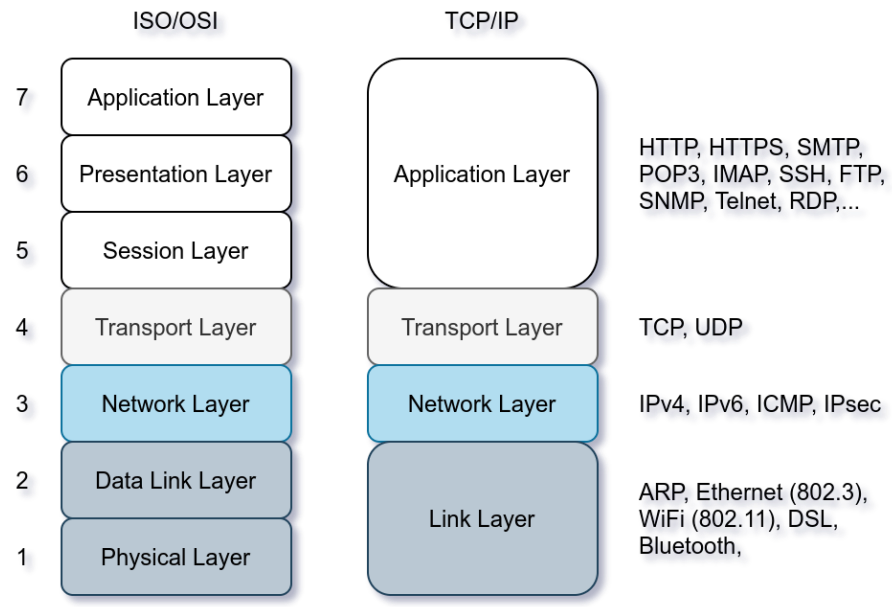
- Scanner sends a packet (like SYN) to a **common port** (e.g. 80, 443).
- If a host responds, it's probably online.
- Works **even if ICMP is blocked**.

🔹 **4. UDP**

- Scanner sends a packet to a UDP port (e.g. 53 for DNS).
- UDP is **less reliable** because it doesn't guarantee replies.
- But sometimes you'll get responses that prove the host is live.

✅ **Why Use Different Methods?**

- **If ICMP is blocked** → Try **TCP or UDP**
- **If you're on the same subnet** → Use **ARP**
- **If you're scanning from another subnet** → Use **ICMP, TCP, or UDP** (ARP won't work)



🧠 Why Do We Use ARP Scans?

Before scanning ports, we need to know **which devices are online**. There's no point in scanning ports on an **offline system** — it's a waste of time and creates unnecessary noise on the network.

🔍 How Nmap Detects Live Hosts (by default)

🛡️ If you're a privileged user (root/sudo):

- **On a local network** (same subnet):
  👉 Nmap uses **ARP requests** to find live hosts.
- **On a different network** (remote targets):
  👉 Nmap sends:
  - ICMP Echo Request (ping)
  - TCP ACK to port 80
  - TCP SYN to port 443
  - ICMP timestamp request

🙅 **If you're not root:**

- Nmap uses TCP SYN scans to ports 80 and 443 to see if a host replies (3-way handshake).

✅ **Nmap's Default Behavior**

By default, Nmap:

1. First **pings** hosts (to see who's online)
2. Then **scans ports** on those live systems

🔧 **If You Only Want to Find Live Hosts (no port scan):**

Use:

nmap -sn TARGETS

This is a **"ping scan" only** — it won't scan ports.

📡 **If You Want to Use ARP Only**

Use:

sudo nmap -PR -sn 10.10.210.6/24

- -PR → use **ARP scan only**
- -sn → skip port scan

✅ This only works on the **same subnet** (local network).
ARP doesn't work across routers.

⚙️ How ARP Scans Work (Under the Hood)

- Your machine sends an **ARP broadcast** to ask:
  "Who has IP 10.10.210.75?"
- Any live machine replies with its **MAC address**
- If there's a reply → the host is **up**

You can **see this traffic** using tools like **Wireshark** or **tcpdump**.

🛠️ Alternative Tool: arp-scan

Another great tool for ARP scanning:

✅ Example:

sudo arp-scan 10.10.210.6/24

- Scans all IPs in the subnet
- Sends ARP requests and lists who replies

📌 If you want to scan a specific network interface:

sudo arp-scan -I eth0 -l

- -I eth0 → use the eth0 interface
- -l → scan the local subnet

🛑 Note: arp-scan is **not installed** by default on the AttackBox.
You can install it with:

sudo apt install arp-scan

🔁 Summary

Tool	Command Example	Description
Nmap	sudo nmap -PR -sn 10.10.210.6/24	ARP scan using Nmap (no port scan)
arp-scan	sudo arp-scan 10.10.210.6/24	ARP scan with a dedicated tool
	sudo arp-scan -I eth0 -l	ARP scan on specific interface

Both tools generate **ARP traffic**, and you'll see similar packets in tools like Wireshark.

📊 Comparison of ICMP Scan Types

Scan Type	Option	ICMP Type Used	Reliability
Echo	-PE	8 (Echo) / 0	🔸 Common, but often blocked
Timestamp	-PP	13 / 14	✅ Better chance than echo
Address Mask	-PM	17 / 18	🔴 Rarely useful, often blocked

🧠 Why Use TCP and UDP for Host Discovery?

Sometimes **ICMP (ping)** is **blocked by firewalls**, so we use **TCP or UDP** instead.
The idea is simple: send packets to **specific ports** and see if **any kind of response** comes back. If something replies, the host is likely **online**.

🔹 **TCP SYN Ping (-PS)**

- Sends a **SYN** (like the start of a connection) to one or more TCP ports.
- **If the port is open** → you get a **SYN/ACK**.
- **If the port is closed** → you get an **RST** (reset).
- Either response means the host is **up**.

Command Example:

sudo nmap -PS -sn 10.10.68.220/24

You can specify ports:

-PS21 # Port 21

-PS21-25 # Ports 21 to 25

-PS80,443,8080 # Common HTTP/HTTPS ports

🔑 **Root is needed** for sending raw SYN packets. Otherwise, a full TCP connection (3-way handshake) is used.

🔸 **TCP ACK Ping (-PA)**

- Sends a **TCP ACK** (used in active connections).
- The target replies with **RST**, since there's no connection.
- Getting an RST = host is **alive**.

Command Example:

sudo nmap -PA -sn 10.10.68.220/24

You can use custom ports, just like with -PS:

-PA21,80,443

🔑 Also requires **root**.

🔹 **UDP Ping (-PU)**

- Sends a **UDP packet** to a port.
- If the port is **closed**, the target sends back an **ICMP "port unreachable"** error.
- If the port is **open**, usually you get **no reply**, but that still may indicate the host is online.

Command Example:

sudo nmap -PU -sn 10.10.68.220/24

⚠️ UDP is **unreliable** — not getting a reply doesn't always mean the host is offline.

🧪 **Summary Table**

Type	Nmap Flag	What It Sends	Expected Reply	Good For
TCP SYN	-PS	SYN to port(s)	SYN/ACK or RST	Fast and reliable (root only)
TCP ACK	-PA	ACK to port(s)	RST	Bypassing firewalls
UDP	-PU	UDP to port(s)	ICMP "port unreachable"	Firewalled or stealthy hosts

Use -sn with each to **skip port scanning** and do only host discovery.

⚡ Bonus Tool: Masscan

- Like Nmap, but **extremely fast**.
- Syntax is similar:

masscan 10.10.68.220/24 -p80

masscan 10.10.68.220/24 -p22-25

masscan 10.10.68.220/24 --top-ports 100

🚀 Great for **massive networks** or **time-sensitive scans**, but can flood networks — be careful.

🔧 Command Cheat Sheet

Scan Type	Example Command
🔌 **ARP Scan**	sudo nmap -PR -sn MACHINE_IP/24
📶 **ICMP Echo (Ping)**	sudo nmap -PE -sn MACHINE_IP/24
⏱️ **ICMP Timestamp**	sudo nmap -PP -sn MACHINE_IP/24
🎭 **ICMP Address Mask**	sudo nmap -PM -sn MACHINE_IP/24
🔐 **TCP SYN Ping**	sudo nmap -PS22,80,443 -sn MACHINE_IP/30
🔁 **TCP ACK Ping**	sudo nmap -PA22,80,443 -sn MACHINE_IP/30
📡 **UDP Ping**	sudo nmap -PU53,161,162 -sn MACHINE_IP/30

✅ -sn: Only perform **host discovery**, skip port scan.

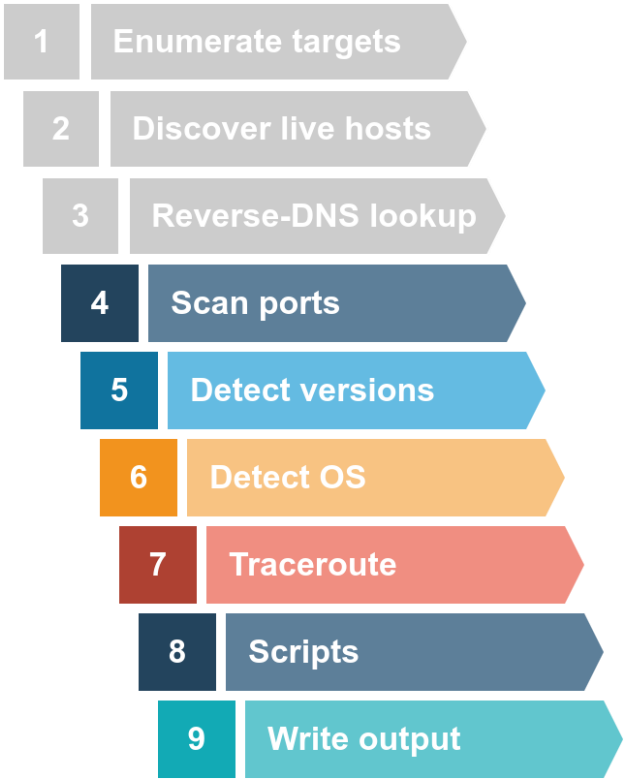
⚙️ **Additional Options**

Option	What it does
-n	Skip DNS lookups (faster + stealthier)
-R	Perform reverse-DNS on **all** hosts (even offline)
--dns-servers IP	Use a specific DNS server (e.g., 8.8.8.8)

Nmap Basic Port Scans

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.

📦 What are TCP and UDP Ports?

- Think of a **port** as a "door" into a computer.
- Each port is used by a **specific service**.
- Like an IP address identifies a **device**, a **port number** identifies a **service** on that device.

🔢 Examples of Common Ports:

Service	Port	Protocol
HTTP (web)	80	TCP
HTTPS (secure web)	443	TCP
DNS	53	UDP (and sometimes TCP)
SSH	22	TCP
FTP	21	TCP

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:

🚦 Nmap Port States Explained

State	What it Means
🔓 **Open**	A service is listening on this port.
🚪 **Closed**	No service is listening, but the port is reachable (no firewall blocking it).
🔒 **Filtered**	Nmap can’t tell if the port is open or closed — something (like a firewall) is blocking it.
❓ **Unfiltered**	Port is reachable, but Nmap couldn't determine if it’s open or closed (e.g., with ACK scan).
🔄 **Open	Filtered**
🚫 **Closed	Filtered**

🔑 Key Takeaways:

- A port can’t be shared by more than one service on the **same IP**.
- **Firewalls** and **security tools** often block or filter ports, which is why Nmap might not always get a clear answer.
- Nmap adapts its scanning strategy depending on how the target responds (or doesn’t).

📦 What’s the TCP Header?

- A TCP header is the **first 24 bytes** of a TCP segment.
- It includes source & destination port numbers, sequence numbers, and **flags** that control the connection.
- These **flags** help manage how a connection is opened, maintained, or closed.

🚩 TCP Flags — What They Do

Flag	Full Name	Purpose
URG	Urgent	Tells the receiver there's urgent data that should be processed immediately.
ACK	Acknowledgement	Confirms receipt of data or another packet.
PSH	Push	Requests immediate delivery to the application layer.
RST	Reset	Forcefully ends the connection or responds when no service is listening.
SYN	Synchronize	Starts a connection (1st step in 3-way handshake).
FIN	Finish	Gracefully closes a connection.

🧪 Why Does This Matter for Nmap?

Nmap uses these flags to create **different types of scans**. For example:

Scan Type	Flag(s) Used	Behavior
TCP SYN Scan (-sS)	SYN	Starts the handshake, but doesn't finish it. Stealthy and fast.
TCP Connect Scan (-sT)	SYN, ACK	Uses full 3-way handshake. Easier but noisier.
ACK Scan (-sA)	ACK	Used to map firewall rules (doesn't tell if port is open).
FIN Scan (-sF)	FIN	Sends FIN packet without handshake; used to bypass some firewalls.

Scan Type	Flag(s) Used	Behavior
Xmas Scan (-sX)	FIN, URG, PSH	Sends “lit-up” packets. Some systems reply differently to weird flag combos.
Null Scan (-sN)	*(No flags)*	No flags set — if target replies, the port might be open or closed depending on system.

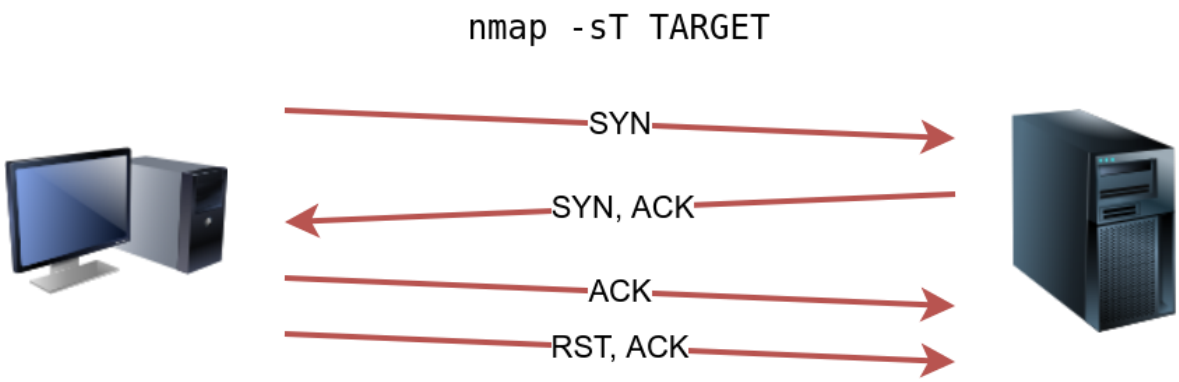
✅ **Summary:**

- **TCP flags** are tiny on/off switches in the TCP header.
- Nmap manipulates these flags to trigger **different responses** from target machines.
- These responses help **identify open/closed/filtered ports** or **bypass firewalls**.

TCP Connect Scan

🔍 **What is TCP Connect Scan?**

- Uses the **standard TCP 3-way handshake**:
    1. SYN → sent from Nmap to the target.
    2. SYN/ACK → received from the target (if port is open).
    3. ACK → sent from Nmap to complete the handshake.
    4. Followed by a **RST/ACK** to **tear down** the connection.



Case: TCP port is open.

⚙️ When is it used?

- **Default** for **unprivileged users** (non-root).
- Cannot craft raw SYN packets without special permissions, so it **uses the OS's connect() system call** instead.

🛠️ Syntax

nmap -sT [target]

🧪 Example:

nmap -sT 10.10.169.34

🚩 Flags & Options You Can Add

Option	Purpose
-sT	TCP Connect scan
-F	Fast mode: scan top 100 ports
-r	Scan ports in numeric order (not random)
-p	Specify ports to scan (e.g., -p 22,80)

📊 Sample Output

PORT STATE SERVICE

22/tcp open ssh

25/tcp open smtp

80/tcp open http

111/tcp open rpcbind

143/tcp open imap

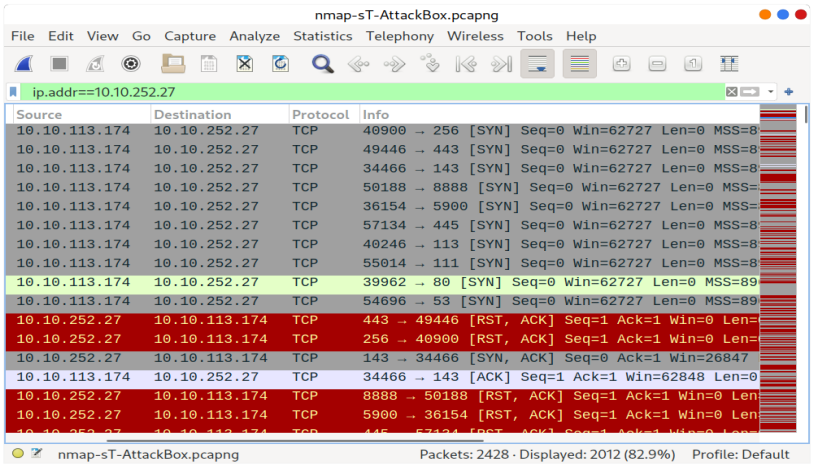
993/tcp open imaps

995/tcp open pop3s

🧠 Key Takeaways

- **Connect scan is more detectable**, as it completes full handshakes.
- Use it **only when you can't run SYN scans** (-sS, which is stealthier).
- Add -F for a faster scan, and -r for ordered port checks.

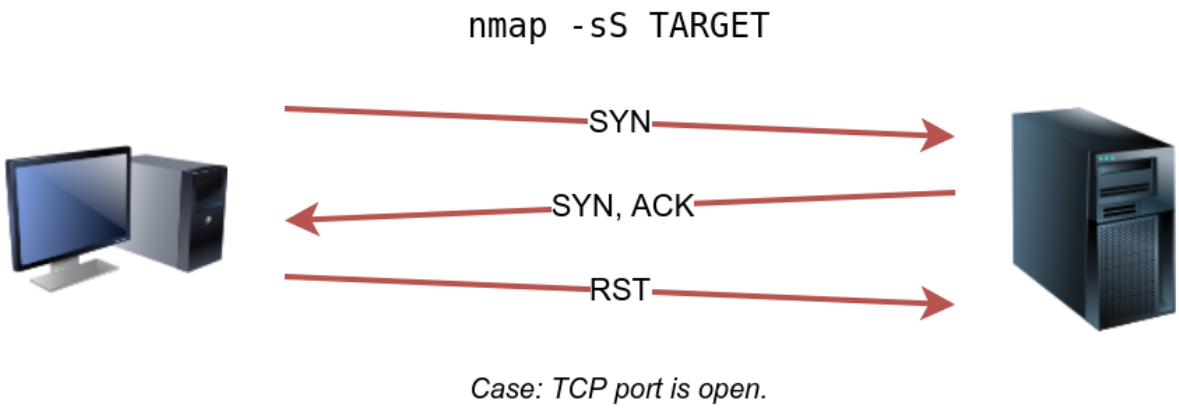
In the following Wireshark packet capture window, we see Nmap sending TCP packets with SYN flag set to various ports, 256, 443, 143, and so on. By default, Nmap will attempt to connect to the 1000 most common ports. A closed TCP port responds to a SYN packet with RST/ACK to indicate that it is not open. This pattern will repeat for all the closed ports as we attempt to initiate a TCP 3-way handshake with them.



We notice that port 143 is open, so it replied with a SYN/ACK, and Nmap completed the 3-way handshake by sending an ACK. The figure below shows all the packets exchanged between our Nmap host and the target system's port 143. The first three packets are the TCP 3-way handshake being completed. Then, the fourth packet tears it down with an RST/ACK packet.

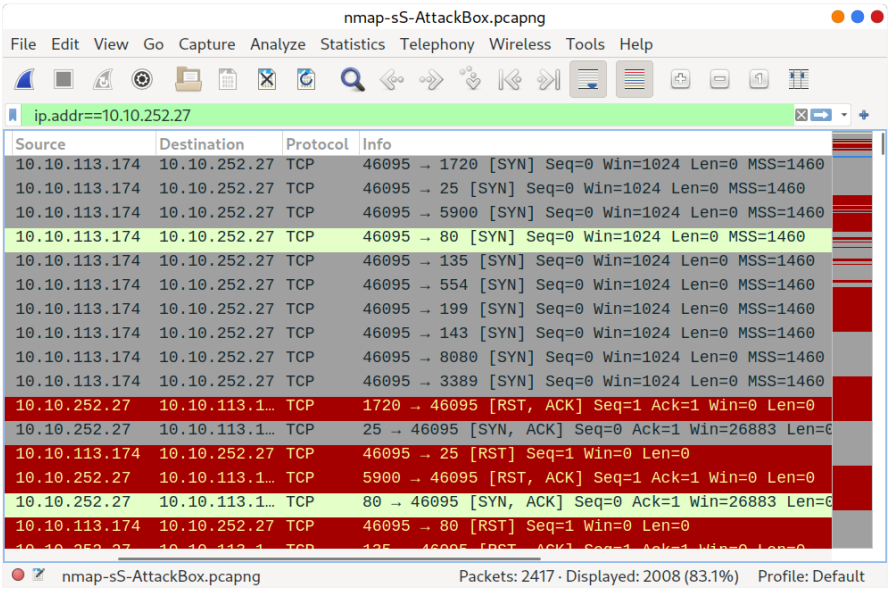
TCP SYN Scan

Unprivileged users are limited to connect scan. However, the default scan mode is SYN scan, and it requires a privileged (root or sudoer) user to run it. SYN scan does not need to complete the TCP 3-way handshake; instead, it tears down the connection once it receives a response from the server. Because we didn't establish a TCP connection, this decreases the chances of the scan being logged. We can select this scan type by using the -sS option. The figure below shows how the TCP SYN scan works without completing the TCP 3-way handshake.

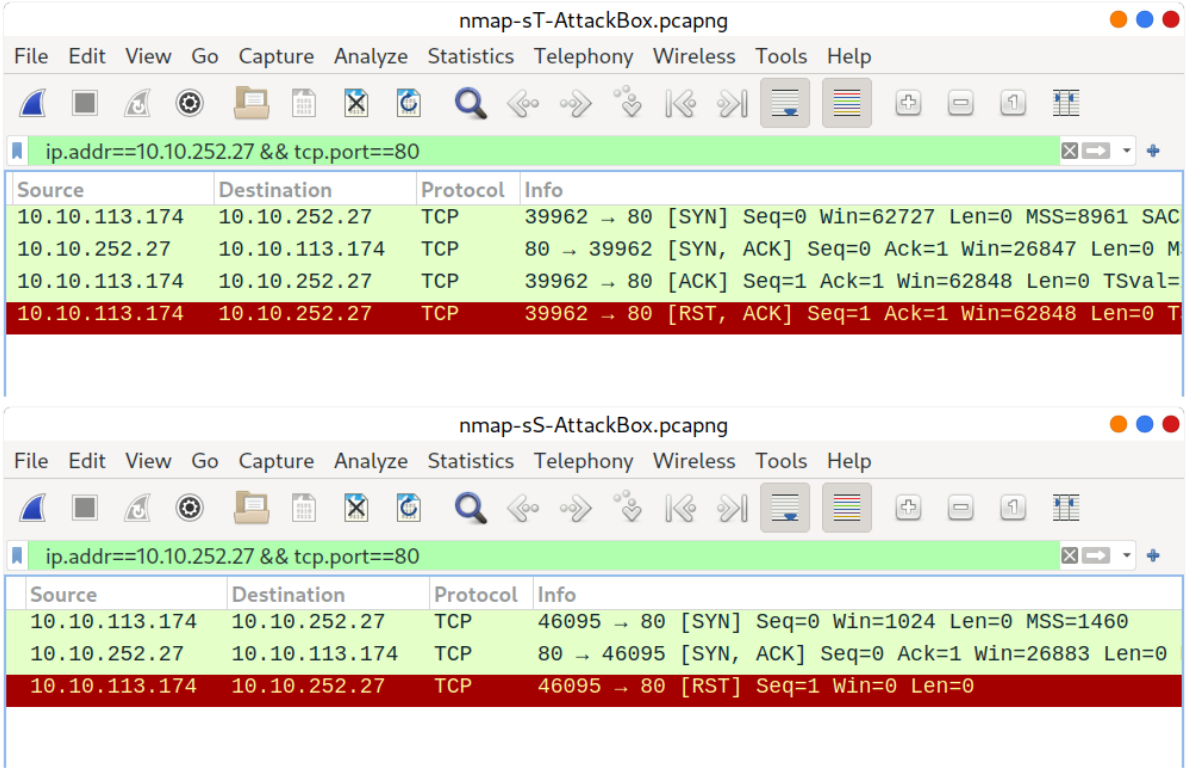


Case: TCP port is open.

The following screenshot from Wireshark shows a TCP SYN scan. The behaviour in the case of closed TCP ports is similar to that of the TCP connect scan.



To better see the difference between the two scans, consider the following screenshot. In the upper half of the following figure, we can see a TCP connect scan -sT traffic. Any open TCP port will require Nmap to complete the TCP 3-way handshake before closing the connection. In the lower half of the following figure, we see how a SYN scan -sS does not need to complete the TCP 3-way handshake; instead, Nmap sends an RST packet once a SYN/ACK packet is received.

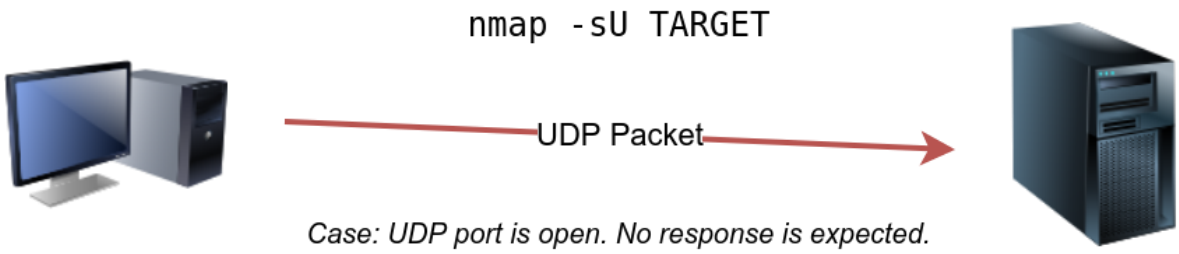


TCP SYN scan is the default scan mode when running Nmap as a privileged user, running as root or using sudo, and it is a very reliable choice. It has successfully discovered the open ports you found earlier with the TCP connect scan, yet no TCP connection was fully established with the target

Example syntax : sudo nmap -Ss 10.10.10.12

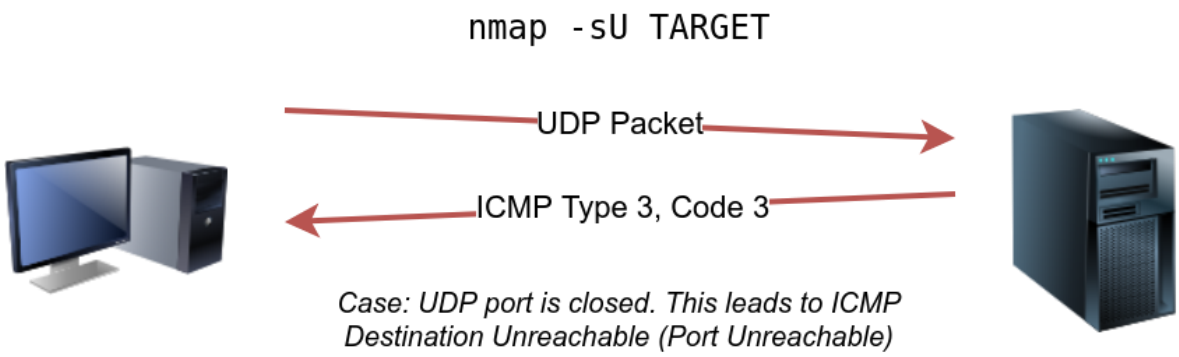
UDP is a connectionless protocol, and hence it does not require any handshake for connection establishment. We cannot guarantee that a service listening on a UDP port would respond to our packets. However, if a UDP packet is sent to a closed port, an ICMP port unreachable error (type 3, code 3) is returned. 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.

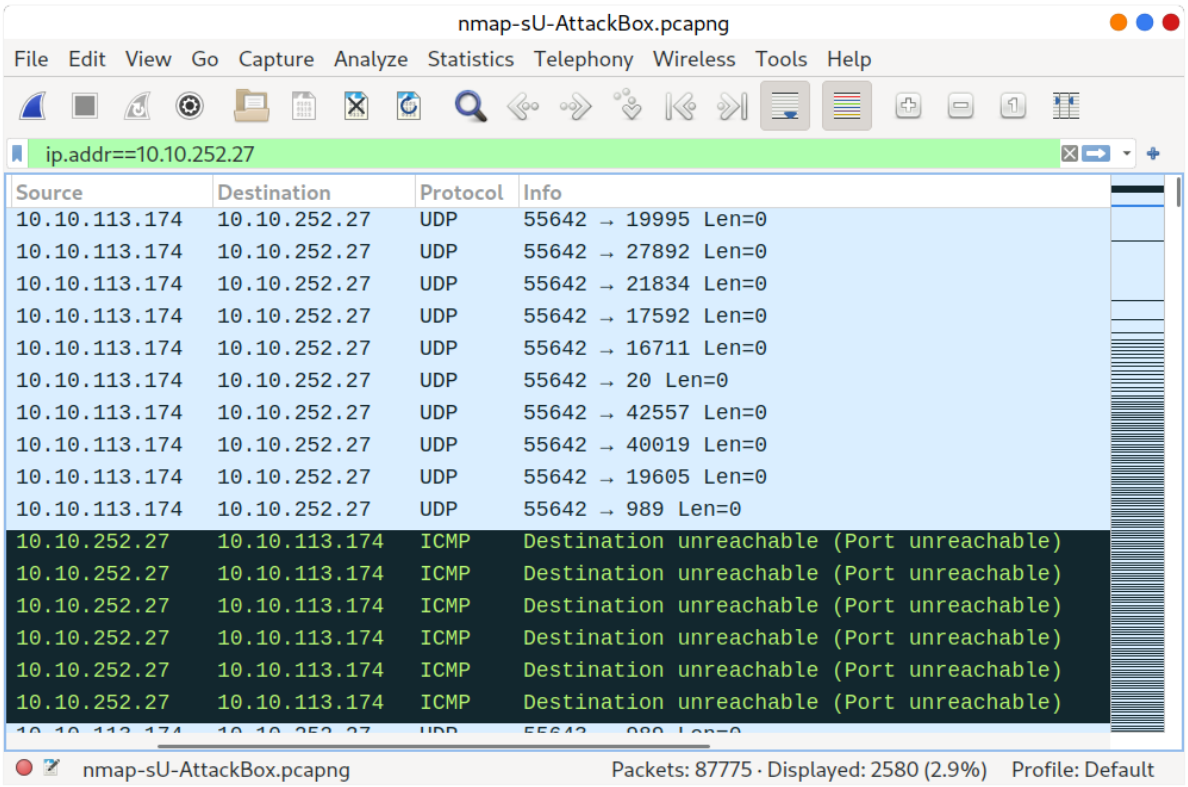


Case: UDP port is open. No response is expected.

However, as shown in the figure below, we expect to get an ICMP packet of type 3, destination unreachable, and code 3, port unreachable. In other words, the UDP ports that don't generate any response are the ones that Nmap will state as open.



In the Wireshark capture below, we can see that every closed port will generate an ICMP packet destination unreachable (port unreachable).



Launching a UDP scan against this Linux server proved valuable, and indeed, we learned that port 111 is open. On the other hand, Nmap cannot determine whether UDP port 68 is open or filtered.

```
pentester@TryHackMe$ sudo nmap -sU 10.10.169.34

Starting Nmap 7.60 ( https://nmap.org ) at 2021-08-30 09:54 BST
Nmap scan report for 10.10.169.34
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
```

🔢 Port Scan Customization

Option	Description	Example
-p22,80,443	Specific ports	Scan only those ports
-p1-1023	Port range	Scan all "well-known" ports
-p-	All 65535 ports	Full port scan
-F	Fast mode	Top 100 ports only
--top-ports 10	Top N common ports	Prioritizes likelihood of being open

---

🕒 Timing Templates

Option	Speed	Use Case
-T0	🐢 Paranoid	Max stealth, evades IDS but **extremely slow**
-T1	🐌 Sneaky	Slower, low-profile
-T2	🚶 Polite	Reduces load on target

Option	Speed	Use Case
-T3	⚖️ Normal	Default
-T4	⚡ Aggressive	Fast, **good for CTFs**
-T5	🔥 Insane	**Very fast**, may cause dropped packets/inaccuracy

🔧 Tip: For **practice labs and CTFs**, -T4 is often the sweet spot.

📶 Rate Control

Use these when you need **precise bandwidth control** (e.g., slow scans, or avoiding network overload):

- --min-rate 10 → Minimum 10 packets/second
- --max-rate 100 → Cap to 100 packets/second

🎛️ Parallelization Control

Advanced control of scan concurrency:

- --min-parallelism 100 → At least 100 probes at a time
- --max-parallelism 300 → No more than 300 concurrent probes

🧠 Practical Examples

Fast scan of top 1000 ports (default), aggressive timing

sudo nmap -sS -T4 10.10.10.10

Full port scan with fast timing

sudo nmap -sS -p- -T4 10.10.10.10

Scan top 10 ports slowly to evade detection

sudo nmap -sS --top-ports 10 -T1 10.10.10.10

Throttle packet rate to max 50/sec

sudo nmap -sS -p1-1023 --max-rate 50 10.10.10.10

What are flags

In **networking**, a **flag** refers to a specific **bit or set of bits** in a **protocol header** (such as TCP or IP) used to **signal control information** about a packet or connection. Flags tell the receiving system how to handle or interpret the packet.

🔑 Example: TCP Flags

TCP uses flags in its header to manage the state of connections. Each flag is 1 bit and can be set (1) or unset (0). Here are the most common TCP flags:

Flag	Name	Purpose
SYN	Synchronize	Initiates a TCP connection (part of the 3-way handshake).
ACK	Acknowledge	Acknowledges received data or connection setup.
FIN	Finish	Gracefully closes a connection.
RST	Reset	Immediately terminates a connection (used for errors or rejection).
PSH	Push	Tells the receiver to push the data to the application immediately.
URG	Urgent	Indicates urgent data (rarely used).

🧠 Why Flags Matter

Flags are critical for:

- **Connection management** (e.g., establishing, maintaining, and tearing down connections)
- **Detecting network behavior** (e.g., in scans or intrusion detection)
- **Performing specific types of Nmap scans** (e.g., SYN scans set only the SYN flag)

🔍 TCP Scan Types and What They Mean

Scan Type	Nmap Option	Flag(s) Used	Closed Port Response	Open Port Response	State Shown
Null Scan	-sN	None (no flags set)	RST	No response	`open
FIN Scan	-sF	FIN	RST	No response	`open
Xmas Scan	-sX	FIN, PSH, URG	RST	No response	`open

🛡️ Why Use These Scans?

- **To bypass stateless firewalls**: These firewalls only inspect SYN packets. Null, FIN, and Xmas scans use non-SYN flags, making them potentially stealthier.
- **To check for basic firewall configurations**: These scans help determine if firewalls or filters drop unexpected TCP flag combinations.
- **When SYN scan (-sS) is not an option**: For example, when SYN packets are filtered or blocked, or you're trying to avoid detection by IDS.

🚫 Limitations

- **Cannot definitively confirm "open"** — only "not closed" (open|filtered).
- **Doesn't work well against Windows targets**: Microsoft's TCP/IP stack doesn't conform to RFC 793 in this case; it responds with RST regardless of port state.
- **Blocked by stateful firewalls and IDS/IPS**: These scans are ineffective against more advanced protections.

🧠 What is a TCP Maimon Scan?

- **Discovered by:** Uriel Maimon (1996)
- **Nmap flag:** -sM
- **TCP flags used:** FIN + ACK
- **Purpose:** Probe TCP ports using an uncommon flag combination to bypass certain defenses and identify open ports.

⚙️ How It Works

Port State	Expected Behavior (RFC-compliant BSD)	Most Systems' Behavior
Open	No response (dropped silently)	Sends **RST**
Closed	Sends **RST**	Sends **RST**

🔍 If the target **does not respond** (i.e., drops the packet), the port is inferred as **open or filtered**.
👄 But on **most modern systems**, **both open and closed ports send RST**, making the scan **ineffective**.

🛑 Limitations

- Rarely useful on modern networks
- Only useful against certain BSD-based or legacy systems
- Won't detect open ports on most Linux/Windows systems

✅ Why Learn It Anyway?

- **Broadens your scanning toolbox** — sometimes obscure scans work in niche environments.
- **Useful for evading basic intrusion detection or stateless firewalls** that look for SYN or standard probes.
- **Highlights how different OS implementations affect recon strategies.**

✅ TCP ACK Scan (-sA)

Purpose:

- Not for finding open ports directly
- Used for **firewall rule mapping**

Behavior:

- Sends TCP packets with **only the ACK flag**
- If a port replies with **RST**, it's **unfiltered** (i.e., not blocked by a firewall)
- If no reply, it's **filtered** (firewall likely blocked it)

When useful:

- Identifying which ports are being **blocked or allowed by a firewall**

✅ TCP Window Scan (-sW)

Purpose:

- Like ACK scan, but uses **TCP Window Size** field for inference

Behavior:

- Still sends ACK packets
- If the **TCP Window size > 0**, port may be **open**
- If it's **0**, port is **closed**

Limitations:

- Works **only on specific OS implementations** (e.g., older Windows)
- Not widely reliable, but can expose **firewall-filtered open ports**

✅ Custom TCP Flag Scan (--scanflags)

Purpose:

- Build **your own scan type** using any combination of TCP flags

Example:

sudo nmap --scanflags RSTSYNFIN <target>

Useful for:

- **Bypassing stateful/stateless firewalls**
- **Evasion scenarios**
- **Deep research** or very specific **target behavior testing**

Caution:

- You need to **interpret results yourself** — Nmap will not always provide helpful state labels for custom scans

---

🧠 Key Concept

ACK and Window scans **don't show service status**, they **map firewall filtering behavior**.

If a port is:

- **Unfiltered** in ACK scan → Not blocked by firewall
- **Closed** in Window scan → Firewall allows packets but no service listens

🔥 Firewalls (What They Do)

A **firewall** is like a security guard for your network. It decides what data packets are allowed through and what should be blocked. It uses a set of **rules**, which can be either:

- **Block everything, allow only exceptions** (e.g. allow only HTTP/HTTPS)
- **Allow everything, block only threats** (less secure, rarely used)

🛡️ **Types of checks it performs**:

- Basic firewalls: inspect **IP header** and **TCP/UDP headers**.
- Advanced firewalls: also analyze **payloads** (actual data being sent).

🕵️‍♂️ IDS (Intrusion Detection System)

An **IDS** watches all traffic that comes into or out of a network. Unlike a firewall, it doesn't block packets—it just **alerts** when it sees something suspicious.

🚨 What it looks for:

- Known malicious patterns (like shellcode or port scans)
- Suspicious behavior (e.g. too many SYN packets too fast)
- Signatures of tools like **Nmap scans**

🧩 Fragmented Packets – What & Why?

When using **Nmap**, you might want to avoid detection by **firewalls** or **IDS**. One trick is to break your scan packets into smaller fragments so that security systems don't easily recognize them.

✅ Why Fragment?

- Firewalls/IDS often expect to see **entire packets**.
- By **splitting up** packets, you make it harder for these tools to detect or understand the scan.

💉 How Fragmentation Works

- Normally, a packet contains both an **IP header** (20 bytes) and a **TCP header** (usually 20-24 bytes).
- When using -f, Nmap **splits the packet** into chunks of 8 bytes or less.
- If you use -f -f (or -ff), it uses chunks of 16 bytes instead.

So, a 24-byte TCP header would be split like this:

- -f → three fragments: 8 + 8 + 8
- -ff → two fragments: 16 + 8

These fragments are **reassembled** by the target system, but **firewalls and IDS** might struggle to analyze them.

🛠️ Nmap Fragmentation Example

Normal Scan:

sudo nmap -sS -p80 10.20.30.144

- Sends a **normal SYN scan** to port 80.
- Easy for IDS/firewalls to detect.

Fragmented Scan:

sudo nmap -sS -p80 -f 10.20.30.144

- Same scan, but **packets are fragmented** (into ≤ 8-byte pieces).
- Harder for detection systems to spot the SYN scan.

More fragmentation:

sudo nmap -sS -p80 -ff 10.20.30.144

- Uses **16-byte fragments**, useful if you want fewer fragments but still avoid detection.

📦 Custom Payload Size – Look Less Suspicious

If you want your packets to **look bigger and more normal**, you can pad them with extra bytes using:

--data-length NUM

Example:

sudo nmap -sS --data-length 50 10.20.30.144

This adds **50 random bytes** of data to each packet. It doesn't affect functionality but **makes the packets less obvious** to simple signature-based IDS tools.

⚠️ Important Notes:

- Fragmentation doesn't guarantee stealth. **Modern IDS/IPS** can still **reassemble packets** and spot Nmap.
- Fragmenting too much might lead to **dropped packets** or scan failures on certain networks.
- Works best on **older or simpler firewalls/IDS systems**.

🧠 Summary

Technique	Purpose	Nmap Option
IP Fragmentation	Bypass packet filters / IDS pattern detection	-f, -ff, --mtu
Add Random Data	Make scan packets look less suspicious	--data-length NUM
Change fragment size	Customize packet structure	--mtu <value>

Spoofing & Idle (Zombie) Scan – Simplified

IP spoofing means faking your IP address so the scan looks like it's coming from another device.

However, spoofing only works if you can still **see the replies**. If not, you won't know the results.

That's where **idle scanning** helps.

In an **idle scan**, you trick the target into thinking another device (the "zombie") is scanning it. You then watch how that zombie behaves to figure out if a port is **open or closed** on the target.

Steps:

1. Ask the zombie for its current IP ID.
2. Spoof a scan to the target, making it look like the zombie sent it.
3. Ask the zombie again.
    - If the IP ID increased by **2**, the port is **open**.
    - If it increased by **1**, it's **closed or filtered**.

Command example:

nmap -sI ZOMBIE_IP TARGET_IP

🛠️ Getting More Details with Nmap

By default, Nmap gives a summary of open/closed ports. But you can **add extra flags** to learn **why** it thinks a port is open or how it reached that conclusion.

✅ 1. --reason – Show Why

Use --reason to see *why* a port is reported as open/closed.
It shows things like:

- **“syn-ack”** – means the port replied as expected → it's **open**
- **“reset”** – means the port denied the request → it's **closed**
- **“arp-response”** – means the **host is up**

🔧 Example:

sudo nmap -sS --reason 10.10.252.27

🔍 2. -v and -vv – Verbose Output

These options give **live progress updates** and **more scan details**, like which ports were discovered first.

- -v = verbose
- -vv = very verbose

🔧 Example:

sudo nmap -sS -vv 10.10.252.27

It shows:

- When host discovery starts and ends

- When each scan phase runs (e.g., SYN scan)
- Which ports are discovered during the scan

🧪 3. -d and -dd – Debugging Details

Use these if you want to see **low-level packet behavior** and internal logic. This output is **very detailed**, best used for deep troubleshooting.

- -d = debug
- -dd = even more debug

⚠️ These can produce **huge outputs**, often more than one screen.

Summary of Useful Flags:

Flag	Purpose
--reason	Show why Nmap labeled a port open/closed
-v / -vv	Show more scan progress and discoveries
-d / -dd	Debug info for in-depth troubleshooting

Use these when you want to **understand the "why" behind your scan results** – great for learning or investigating unexpected behavior.

In the context of **network scanning**, **"idle"** means **not active** — a system that is **quiet** on the network, not sending or receiving much data.

When we talk about an **idle host** (especially in an **Idle/Zombie scan**), we're looking for:

- A system that is **on** and connected to the network.
- A system that is **not actively communicating** (not browsing, downloading, or serving).
- A system with a **predictable IP ID sequence** — meaning each time it sends a packet, its IP ID increases by 1.

Why is this useful?

In an **Idle scan**, you use this quiet, predictable system to send spoofed packets. If it's really idle, any unexpected change in its IP ID means someone (like the target you're scanning) responded **to it**, which helps you detect open ports **without revealing your own IP**.

So, **"idle" = quiet, predictable, and not doing much** — perfect for sneaky scanning.

🔹 **Common Examples of Idle Devices:**

1. **Old network printers**
    - Especially ones that aren't used frequently.
    - Many printers respond to basic network probes but don't generate much outgoing traffic.
2. **Unmonitored IoT devices**
    - Smart plugs, sensors, or basic IoT lightbulbs.
    - Devices that sit quietly waiting for a command.
3. **Legacy systems**
    - Old servers or machines that are powered on but rarely accessed.
    - Think of lab computers, backup servers, or unused virtual machines.

4. **Networked appliances**
    - Things like IP cameras (when not actively viewed), digital thermostats, or refrigerators with network capability.
5. **Home routers or access points (not heavily used)**
    - Secondary routers or range extenders on a quiet segment of the network.
6. **Embedded systems**
    - Devices running things like vending machines, control systems, or digital signage that are online but rarely interact.

🔍 Scan Types and Examples

Scan Type	Command Example	Description
TCP Null Scan	sudo nmap -sN 10.10.30.57	Sends a TCP packet with no flags. Closed ports reply with RST.
TCP FIN Scan	sudo nmap -sF 10.10.30.57	Sends a TCP packet with the FIN flag. Closed ports send RST.
TCP Xmas Scan	sudo nmap -sX 10.10.30.57	Sends FIN, PSH, and URG flags. Closed ports reply with RST.
TCP Maimon Scan	sudo nmap -sM 10.10.30.57	Sends FIN + ACK. Sometimes helps with older BSD systems.
TCP ACK Scan	sudo nmap -sA 10.10.30.57	Useful for **firewall rule discovery**. Doesn't determine port state directly.

Scan Type	Command Example	Description
TCP Window Scan	sudo nmap -sW 10.10.30.57	Like ACK scan but checks TCP window size in RST to infer state.
Custom TCP Scan	sudo nmap --scanflags URGACKPSHRSTSYNFIN 10.10.30.57	Set any combination of TCP flags manually.
Spoofed IP Scan	sudo nmap -S SPOOFED_IP 10.10.30.57	Spoofs your IP; only useful if you can **capture the response**.
MAC Spoofing	--spoof-mac SPOOFED_MAC	Spoof your MAC address (same subnet only).
Decoy Scan	nmap -D 10.10.0.1,ME 10.10.30.57	Use fake IPs to hide your real one. “ME” places your IP among decoys.
Idle (Zombie) Scan	sudo nmap -sI ZOMBIE_IP 10.10.30.57	Uses an idle system to scan stealthily. Tracks IP ID changes to infer port status.
Fragmented Packets	-f or -ff	Splits packets into 8 or 16 byte chunks to bypass packet inspection.

⚙️ Additional Scan Options

Option	Purpose
--source-port PORT	Specify which source port to use (can help bypass basic filtering).
--data-length NUM	Add random data to make packets a specific size — makes scans less obvious.
--reason	Shows **why** Nmap marked a port open/closed (e.g., syn-ack response).
-v, -vv	Adds **verbose output** — shows scanning phases and early discoveries.
-d, -dd	Enables **debug mode** — shows internal logic and packet details.

📌 Quick Tips

- Null, FIN, and Xmas scans rely on **lack of response** from open ports.
- Maimon, ACK, and Window scans **rely on behavior of RST packets**.
- **Idle scan** is very stealthy but only works if the zombie is truly idle.
- **Fragmentation** and **decoys** are tricks to bypass firewalls or hide your real IP.
- Use --reason and -vv for learning *how* Nmap interprets responses.

🔍 What is Service Detection?

After finding open ports, Nmap can try to **identify which services are running** (and which versions). This helps you check for **known vulnerabilities**.

🛠️ How to Use It

- Use **-sV** to enable service detection.
- Use **--version-intensity LEVEL** to control how aggressive Nmap is:
    - 0: very light and fast
    - 9: very detailed, slower
    - Example: nmap -sV --version-intensity 5 10.10.202.25

⚠️ Important Notes:

- Running -sV **requires a full connection** (TCP 3-way handshake).
- This **disables stealth scanning** (-sS), since Nmap must communicate directly with the service.

🖥️ Example Output:

sudo nmap -sV 10.10.202.25

Port	State	Service	Version Info
22/tcp	open	ssh	OpenSSH 6.7p1 Debian
25/tcp	open	smtp	Postfix smtpd
80/tcp	open	http	nginx 1.6.2
110/tcp	open	pop3	Dovecot pop3d
111/tcp	open	rpcbind	version 2-4

🧠 Why It Matters

By knowing **exact versions**, you can:

- Check for **vulnerabilities**.
- Choose the right **exploits**.
- Understand the target's **tech stack**.

🖥️ OS Detection (-O)

Nmap can **guess the operating system** (like Linux or Windows) by analyzing how the target responds to certain packets.

✅ Use:

sudo nmap -sS -O 10.10.202.25

📋 Output Example:

- OS detected: Linux 3.X
- Guessed: Linux kernel 3.13
- Real OS: Linux 3.16

⚠️ Notes:

- Nmap needs **at least one open & one closed port** for best accuracy.
- It may get the **OS version wrong**, especially on virtual machines.
- Results are **approximate**, not 100% reliable.

🌍 Traceroute (--traceroute)

Traceroute finds the **path (hops/routers)** between you and the target.

✅ Use:

sudo nmap -sS --traceroute 10.10.202.25

📋 **Output Example:**

HOP RTT ADDRESS

1 1.48 ms 10.10.202.25

This means the target is **1 hop away** — directly connected or on the same network.

⚠️ **Notes:**

- Nmap's traceroute works **in reverse** from standard traceroute (starts with high TTL).
- If a router blocks ICMP replies, **some hops may not appear**.

💡 **Summary:**

Feature	Option	Purpose
OS Detection	-O	Guess OS type and version
Traceroute	--traceroute	Show network path to the target

🛠 **Nmap Scripting Engine (NSE) — Simplified Notes**

What is it?
NSE lets Nmap use small programs (scripts) to automate and extend its scanning features. These scripts are written in **Lua**, but **you don't need to know Lua** to use them.

📁 **Where are the scripts?**

On the AttackBox, scripts are found at:
/usr/share/nmap/scripts

There are **hundreds** of them, grouped by what they scan (e.g., HTTP, FTP, etc.).

⚙️ Script Categories

Scripts are organized into categories. Examples:

Category	Purpose
default	Basic info (used with -sC)
auth	Login and auth checks
brute	Brute force attacks
vuln	Checks for vulnerabilities
discovery	Info gathering (e.g. DNS, DBs)
safe	Will not crash or attack
intrusive	Might be noisy or harmful
exploit	Try to exploit known flaws

🚀 How to use them:

- **Default scripts:**
  sudo nmap -sS -sC <IP>
  (Runs basic scripts after a SYN scan)
- **Specific script by name:**
  --script=http-date
- **Pattern match:**
  --script="ftp*" (all FTP scripts)

🔐 **Example Output:**

Running -sC reveals details like:

- SSH key types
- HTTP server title
- POP3/IMAP capabilities

🧪 **Example of a safe script:**

sudo nmap -sS -n --script "http-date" <IP>

Checks the server's HTTP date/time and compares it to local time.

⚠️ **Caution:**

Some scripts can **crash services** or **perform attacks** (like brute-force or DoS).
Only use them on systems you're allowed to test.

Don't blindly trust or run scripts downloaded from the Internet.

💾 **Saving Nmap Scan Results**

When you run an Nmap scan, it's smart to save the output. Nmap offers **4 formats**:

🟢 **1. Normal Output**

- Like what you see in the terminal.
- Use: -oN filename
- Good for reading and reviewing.

Example:

nmap -sS -sV -O -oN scan.nmap <IP>

🟠 **2. Grepable Output**

- Easy to filter/search using grep.
- Use: -oG filename
- Each line has all the info, including the IP — useful when scanning many hosts.

Example:

nmap -sS -sV -O -oG scan.gnmap <IP>

Grep Example:

grep http scan.gnmap

🔵 **3. XML Output**

- Good for importing into other tools.
- Use: -oX filename

⚡ **4. All Formats at Once**

- Saves Normal, Grepable, and XML all together.
- Use: -oA basefilename

Example:

nmap -sS -sV -O -oA fullscan <IP>

Creates:

- fullscan.nmap

- fullscan.gnmap
- fullscan.xml

❌ 5. Script Kiddie Output

- Useless for real analysis. Just a joke format.
- Use: -oS filename
- Looks “leet” but don’t use it seriously.

🔍 What You Learned

- How to detect running **services**, their **versions**, and the **host OS**
- How to use **traceroute**
- How to choose and run **Nmap scripts** for pentesting
- How to **save** scan results in various formats

📌 Key Nmap Options

Option	Meaning
-sV	Detect service and version info on open ports
-sV --version-light	Use only the most likely probes (faster)
-sV --version-all	Use **all** available probes (slower, thorough)
-O	Detect the target operating system
--traceroute	Run a traceroute to the target
--script=<SCRIPTS>	Run specified Nmap scripts (e.g., http-*, ftp-anon, etc.)
-sC or --script=default	Run the default script set (same as --script=default)

Option	Meaning
-A	Aggressive mode: -sV -O -sC --traceroute all in one

💾 Saving Output

Option	Format	Description
-oN	Normal	Human-readable terminal output
-oG	Grepable	Easily searchable with grep
-oX	XML	Best for automated tools or importing elsewhere
-oA	All formats	Saves in .nmap, .gnmap, and .xml formats