NETWORK SCANNING & RECONNAISSANCE

Module 3 — Network Scanning & Enumeration (Full, Detailed)

Goal: discover live hosts, open ports, services, versions, OS, and basic configuration information so you can build safe, repeatable attack plans.

> Important rule: Only scan systems you are authorized to test. Scanning without permission is illegal and unethical.

---

1. Scanning Workflow (High-level)

1. **Confirm scope & authorization** (IP ranges, domains, exclusions, time window).
2. **Passive discovery** (OSINT, Shodan, DNS, public sources).
3. **Light host discovery** (ARP / ping sweep inside local network).
4. **Fast breadth scan** (masscan / zmap) to find responsive hosts and open ports.
5. **Focused deep scan** (nmap) on discovered hosts/ports for service/version/OS detection and NSE scripts.
6. **Service-specific enumeration** (SMB, HTTP, SSH, SNMP, etc.) with specialized tools.
7. **Analyze, filter, and prioritize** results.
8. **Repeat / re-scan** as necessary with timing and stealth adjustments.

---

2. Pre-scan checklist & safety

- Get written authorization (scope, IPs, time window).
- Inform defenders if necessary (to avoid incident response).

- Respect rate limits and DoS risk.
- Use non-destructive options unless exploitation is permitted.
- Log your scans and timestamps for reports.

3. Host discovery (Who is alive?)

3.1 Ping sweep with nmap

```
nmap -sn 192.168.1.0/24
```

What it does: Sends ICMP echo, TCP/ARP probes to determine live hosts.

Interpretation: Lines like `Nmap scan report for 192.168.1.10` indicate a live host. `Host is up (0.0031s latency).` means responsive.

Notes: `-sn` avoids port scanning; it's safe and fast. For local Ethernet, ARP-based discovery is often the most reliable.

3.2 ARP discovery (netdiscover / arp-scan)

```
netdiscover -r 192.168.1.0/24
arp-scan --interface=eth0 192.168.1.0/24
```

What it does: Sends ARP requests — effective on layer-2 networks even when ping is blocked.

Interpretation: Returns IP, MAC, and vendor (useful to identify routers/printers/IoT).

3.3 Fast host discovery for large ranges (masscan)

```
sudo masscan 10.0.0.0/8 -p80,443 --rate=10000 -oL masscan_results.txt
```

What it does: Extremely fast TCP SYN scanner (can scan millions of IPs).

Important: masscan spoofs source ports and uses raw packets — use with care and permission.

Output: `-oL` produces a simple list of hits; you typically feed this into nmap for deeper scans.

3.4 Combining masscan -> nmap pipeline

1. Run masscan for speed:

```
sudo masscan 10.0.1.0/24 -p0-65535 --rate=5000 -oG masscan.gnmap
```

1. Extract IPs and ports for nmap:

```
cat masscan.gnmap | awk '/open/ {print $2 " -p" $4}' | sed 's/,/,/g' > target
s.txt
```

1. Run nmap against a specific IP:port list (example format: `192.168.1.10 -p22,80`)

```
while read ip portstr; do nmap -sS -sV -O -p ${portstr#-p} $ip -oA nmap_
${ip}; done < targets.txt
```

Why this: masscan finds things fast; nmap confirms, fingerprints, and runs NSE scripts reliably.

---

4. Port scanning (what's open?)

4.1 Full TCP port scan (nmap)

```
nmap -p- --min-rate 1000 -T4 192.168.1.10 -oA full_tcp_scan
```

Flags explained:

- `p-` scan all 65535 ports
- `-min-rate 1000` try to send packets faster (use cautiously)

- `T4` aggressive timing for speed
- `oA` save in all formats (`.nmap` , `.gnmap` , `.xml`)

Interpretation: Open ports shown as `22/tcp open ssh` .

4.2 SYN Stealth scan (fast + stealthy)

```
nmap -sS -Pn -p22,80,443 192.168.1.10 -oN syn_scan.txt
```

- `sS` : SYN scan (doesn't complete TCP handshake)
- `Pn` : treat host as up (skip host discovery)

  Use case: Less noisy than connect scans, often bypasses basic logging.

4.3 Connect scan (when SYN is blocked)

```
nmap -sT 192.168.1.10 -p22,80
```

What: Uses OS connect() syscall; more detectable but works when privileges limited.

4.4 UDP scanning

```
nmap -sU -p 53,67,68,161 192.168.1.10 -sV --script=vulners -oN udp_scan.tx
t
```

Notes: UDP is slow and unreliable; responses may be absent. Combine with service probes (`-sV`) and specific NSE scripts. Increase timing only if safe.

4.5 Timing templates (nmap -T)

- `T0` paranoid (very slow)
- `T1` sneaky
- `T2` polite

- `T3` normal
- `T4` aggressive
- `T5` insane (very loud, fast — risky)

Use `-T0/-T1` for stealth in sensitive environments.

---

5. Service & version detection (what is running?)

5.1 Basic service detection

```
nmap -sV 192.168.1.10 -p22,80,443 -oN svc_version.txt
```

What it does: Connects to open ports, probes banners and fingerprints services.

Example output:

```
22/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.6 ((CentOS))
```

Use versions to map to known CVEs.

5.2 Aggressive detection (NSE + OS + traceroute)

```
nmap -A -p80,443 192.168.1.10 -oA aggressive_scan
```

- `A` = `sV -O --traceroute --script=default,safe`

5.3 Targeted NSE scripts

```
nmap --script http-enum,smb-enum-shares -p80,445 192.168.1.10 -oN nse_scan.txt
```

NSE (Nmap Scripting Engine): powerful for vulnerability checks, enumeration, brute force, etc. Always check script categories (safe / intrusive) before running.

6. OS fingerprinting

6.1 nmap OS detection

```
nmap -O 192.168.1.10
```

What: Sends TCP/IP probes to determine OS and kernel versions. Output: `OS details: Linux 3.10 - 3.16`.

Caveats: Firewalls and packet normalization can mislead fingerprinting.

6.2 TCP/IP fingerprinting with p0f (passive)

```
sudo p0f -i eth0
```

Use case: Passive fingerprinting by listening to packets — stealthy.

7. Enumeration — service-specific deep dives

After scanning, enumerate each exposed service with tools suited to it.

7.1 HTTP / Web enumeration

7.1.1 Banner & tech detection (WhatWeb)

```
whatweb -v http://192.168.1.10
```

Output: Frameworks, CMS, server versions, plugins.

7.1.2 Directory brute force (gobuster / dirsearch)

```
gobuster dir -u http://192.168.1.10 -w /usr/share/wordlists/dirb/common.txt -
t 50 -o gobuster_out.txt
```

- `dir` mode enumerates directories/files.
- `t` threads.

7.1.3 Web vulnerability scan

```
nikto -h http://192.168.1.10 -o nikto_report.txt
```

What: Finds common misconfigs: dangerous headers, outdated server, default files.

7.1.4 Web app scanning & fingerprinting (Burp / OWASP ZAP)

Use proxies (Burp) for interactive testing, mapping inputs, and fuzzing.

7.2 SSH

7.2.1 Banner grab

```
nc 192.168.1.10 22
```

Expect: `SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u7`

7.2.2 Brute forcing (only if permitted)

```
hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.10 -t 4 -f
```

Notes: Rate-limit, lockout detection, and legality concerns — use only in scope.

7.3 SMB / Windows

7.3.1 Quick share enumeration

```
smbclient -L //192.168.1.10 -N
```

What: Lists shares if null sessions allowed.

7.3.2 Detailed enumeration (enum4linux)

```
enum4linux -a 192.168.1.10
```

Output: Users, groups, shares, OS, domain info — extremely useful for AD environments.

7.3.3 SMB mapping (smbmap)

```
smbmap -H 192.168.1.10
```

What: Shows accessible shares and permissions (read/write). Good for finding writeable shares for initial footholds.

7.4 SNMP

7.4.1 Community string check

```
snmpwalk -v2c -c public 192.168.1.10
```

What: If returns data, `public` is valid — reveals processes, config, sometimes credentials.

7.5 FTP

```
ftp 192.168.1.10
# or
nmap --script ftp-anon -p21 192.168.1.10
```

Check for anonymous login and writeable directories.

7.6 RDP

```
nmap -p 3389 --script rdp-enum-encryption 192.168.1.10
```

What: Enumerates encryption level and can reveal user lists via certain RDP implementations.

7.7 Databases: MySQL, PostgreSQL, Redis, ElasticSearch

7.7.1 MySQL

```
mysql -u root -h 192.168.1.10 -p
```

If anonymous access or weak creds allowed, you can enumerate databases and users.

7.7.2 Redis (dangerous if misconfigured)

```
redis-cli -h 192.168.1.10
# try INFO command or check for CONFIG GET
```

Misconfigured Redis can allow remote code execution via save/load module tricks — treat carefully and only if in scope.

7.7.3 Elasticsearch

```
curl http://192.168.1.10:9200/_cluster/health?pretty
```

Open ES often exposes indices and PII.

8. Advanced scanning & evasion techniques (use carefully)

> These techniques can trigger IDS/IPS and are noisy. Use only with authorization and for learning.

8.1 Decoy scans

```
nmap -sS -D RND:10 192.168.1.10
```

- `D RND:10` sends decoy traffic from random addresses. Makes attribution harder for defenders.

8.2 Fragment packets

```
nmap -f 192.168.1.10
```

- `f` fragments packets — may bypass simplistic firewalls; often blocked by modern devices.

8.3 Source port spoofing

```
nmap --source-port 53 192.168.1.10
```

Uses source port 53 (DNS) to attempt bypass of poorly configured filters.

8.4 Idle scan (stealthy, advanced)

```
nmap -sI zombie_ip 192.168.1.10
```

Uses an idle host (zombie) to probe target; stealthy but requires a suitable zombie host with predictable IP ID behavior.

8.5 Timing & rate controls

- Lower speed: `T1` or `-scan-delay 500ms`
- Rate limiting in masscan: `-rate 1000`

Caution: Evasion often increases likelihood of false negatives or breaking services.

---

9. Parsing & analyzing results

9.1 Nmap XML analysis

```
nmap -oX results.xml 192.168.1.0/24
xsltproc /usr/share/nmap/nmap.xsl results.xml -o report.html
```

Convert XML to HTML for readable reports.

9.2 Grepable output

```
nmap -oG quick.gnmap 192.168.1.0/24
grep "open" quick.gnmap | awk '{print $2 ":" $4}' > open_ports_list.txt
```

9.3 Importing into tools

- **Dradis**, **Faraday**, **Serpico** for reporting and collaboration.
- **Nmap XML** is commonly ingested.

---

10. Practical example — Full scan + enumeration pipeline

1. **Host discovery (ARP)**:

```
sudo arp-scan --interface=eth0 192.168.100.0/24 -o arp_results.txt
```

1. **Fast port discovery (masscan for top ports)**:

```
sudo masscan 192.168.100.0/24 -p80,443,22,445,3389 --rate 5000 -oG m
asscan_hits.gnmap
```

1. **Parse hits into an nmap file**:

```
cat masscan_hits.gnmap | awk '/open/ {print $2}' | sort -u > hosts_alive.txt
```

1. **Nmap service & version + NSE (per-host)**:

```
for ip in $(cat hosts_alive.txt); do
  nmap -sS -sV -O --script=default,safe -p- $ip -oA nmap_${ip}
done
```

1. **Service-specific enumeration**: if 445 open, run enum4linux/smbmap; if 80 open, run gobuster/nikto; etc.

---

11. Common interpretation patterns & red flags

- **Many open high-numbered TCP ports** → possible host running many services or dev machine.
- **SNMP readable (`public`)** → gather system info, routing, sometimes credentials.
- **SMB anonymous shares** → possible misconfig allowing data exfiltration.
- **Web servers returning sensitive files (/.env, /config.php)** → high priority.
- **RDP with weak encryption** → potential credential harvesting or downgrade attacks.
- **NTP/UDP amplification services** → possible DDoS vector (report to owners).

---

12. Documentation for the report

For every scan, save:

- Date & time & timezone
- Command used (exact)
- Target IPs / ranges

- Output files (`.nmap`, `.xml`, `.gnmap`)
- Screenshots / banners (copy-paste)
- Any potential impact (false positives, service disruption risk)

Example report snippet:

> nmap -sS -sV -O --script=default -p22,80,443 192.168.1.10 -oA nmap_192.168.1.10
>
> Findings: `22/tcp open ssh OpenSSH 7.4` — recommended to check for weak authentication and outdated version.

---

13. Extras — Helpful one-liners & tips

- **Run nmap from a socks proxy (pivot):**

```
proxychains4 nmap -sT -Pn -p22 10.0.2.15
```

- **Check for HTTP methods allowed:**

```
curl -I -X OPTIONS http://192.168.1.10
```

- **Fetch robots.txt quickly:**

```
curl http://192.168.1.10/robots.txt
```

- **Extract banner with netcat:**

```
echo | nc 192.168.1.10 25
```

---

14. Lab exercises (practice safely in a lab)

1. Build a lab with 3 VMs: one Linux, one Windows, one vulnerable web app. Practice full pipeline: discovery → masscan → nmap → web enumeration.

2. Use `arp-scan` to find devices when ICMP is blocked.
3. Use `gobuster` to enumerate directories and interpret results.
4. Configure a mock SNMP device with `public` community and practice `snmpwalk`.
5. Timeboxing: run nmap with `T1` vs `T4` and observe logs on host IDS (practice stealth).

---

15. Summary — key takeaways

- Start with passive recon; escalate to active scanning only with authorization.
- Use **masscan** for breadth, **nmap** for depth.
- Combine scanning results intelligently (masscan → nmap → service tools).
- Always parse, filter, and prioritize findings.
- Understand the output — fingerprints + versions = possible CVEs.
- Use timing and evasion carefully — it increases false negatives and risks.