Reconnaissance

Reconnaissance is the first phase of an attack (ethical hacking or malicious hacking). It involves collecting information about the target system, network, or organization to find weaknesses.

Reconnaissance is the systematic process of gathering information about a target to inform subsequent decisions or actions. It appears across many fields: militaries scout terrain and enemy forces to plan manoeuvres businesses research markets and competitors before launching products; and cybersecurity professionals perform reconnaissance to map networks, discover assets, and identify weaknesses. In cyber contexts recon is divided into passive and active techniques. Passive reconnaissance collects publicly available data—such as domain registration records, social media profiles, leaked credentials, and search engine results—without interacting directly with target systems. Active reconnaissance involves direct probes like port scans, banner grabbing, and vulnerability scanning, which provide detailed technical insight but can be detected. Effective reconnaissance balances thoroughness with stealth and ethics: defenders use it to harden systems and anticipate threats, while attackers rely on it to craft exploit strategies. Common tools include Nmap for scanning, Shodan for discovering exposed devices, and OSINT frameworks for aggregating public intelligence. Ultimately reconnaissance reduces uncertainty, guiding planners toward informed choices, exposing risks early, and increasing the chance of successful, low-risk operations in both defensive and offensive contexts. Regularly updating reconnaissance practices and sharing findings with stakeholders improves resilience and keeps security measures aligned with evolving threats and technology.

Types of reconnaissance.

1. Passive Reconnaissance.
2. Active reconnaissance.

-

- Passive Reconnaissance.

Passive reconnaissance is the process of **gathering information about a target without directly interacting** with its systems. The goal is to stay undetected while collecting as much intelligence as possible. Since there is no direct contact, the target usually has no idea someone is gathering data.

Passive Reconnaissance are.

Whois: WHOIS is the a protocol/tool is used to look up domain registration details. Which provides info such as Domain or register, Contact ,emails & phone number, name server , Registration & expiration dates.

Using WHOIS in Kali Linux

Kali Linux has the whois command pre-installed.

Steps:

1. Open a terminal in Kali.
2. Run:
3. whois example.com

Replace example.com with the target domain.

4. You'll see details like:
    - Registrar (e.g., GoDaddy, Namecheap)
    - Registrant info (sometimes hidden with privacy protection)
    - Administrative & technical contacts
    - Domain status & expiration

Nslookup : nslookup is a network command-line tool used to query **Domain Name System (DNS)** servers and obtain information about domain names, IP addresses, and other DNS records. It's very useful for troubleshooting network issues and understanding how domain names resolve to IP addresses.

nslookup [options] [domain_name or IP_address]

Google Dorking: combining Google search operators (like site:, filetype:, intitle: etc.) to locate specific pages or file types. It's simply smarter searching — but it can reveal unintentionally exposed files, outdated pages, or misconfigured systems.

Useful operators (safe, general)

- site:example.com — restrict results to a domain.
- Filetype :pdf — find PDFs (also docx, xls, zip, etc.).

- intitle :"your words" — pages with those words in the title.
- Inurl : admin — pages with “admin” in the URL (useful to locate admin pages you own).
- cache: URL — view Google’s cached copy of a page.
- related:example.com — sites related to a domain.
- Intext :"phrase" — finds pages containing the phrase in the body.
- Quoting: "exact phrase" — match the exact phrase.
- Boolean: site:example.com (security OR privacy) — combine terms.

Shodan : Shodan is a **search engine for the Internet of Things (IoT)** and devices connected to the internet. Unlike Google, which indexes websites, Shodan indexes **servers, routers, webcams, industrial control systems, and other connected devices**.

Key Features:

1. **Device Discovery:**
    - You can find devices by type, software version, location, and open ports.
    - Example: Search for “webcam” in a city to see all public cameras.
2. **Vulnerability Analysis:**
    - Shodan can show devices with **known vulnerabilities**, outdated software, or default passwords.
3. **Network Research:**
    - Security researchers use it to **map the internet** and see exposed systems.
4. **Filters:**
    - country: → Filter devices by country
    - port: → Filter by open port
    - org: → Filter by organization
    - os: → Filter by operating system

Active Recon — Ping Sweep & Banner Grabbing

Summary, risks, defensive detection & mitigation, and safe testing notes.

Overview / Purpose

Active reconnaissance (active recon) is the phase where an attacker probes a network or host to discover live systems, open services, and software versions. Two common active recon techniques are **ping sweeps** (to find which hosts are up) and **banner grabbing** (to collect basic service/version information). Both give an attacker a map of targets and possible vulnerabilities — and both are detectable if defenders are monitoring.

---

1) Ping Sweep — What it is

- **Goal:** Identify which IP addresses in a range respond (are alive) so the attacker can focus further probing on live hosts.
- **How it works (conceptually):** Send lightweight network probes to many IPs and note which addresses reply. Probes may use ICMP echo, TCP SYN to a common port, or other "is host up?" packets.
- **Why it matters:** Reduces attack surface by eliminating unreachable IPs and points to active machines or devices (servers, workstations, IoT).

Defensive indicators & detection

- Unusually high ICMP or TCP SYN volume across many destination IPs.
- Bursts of short-timed probes from a single source targeting a broad IP range.
- Alerts from IDS/IPS on network scanning signatures or threshold-based scan rules.
- Netflow/flow logs showing many small flows with short duration to many hosts.

Mitigations and hardening

- Block/limit ICMP echo replies at network edge (but allow necessary ICMP types for diagnostics).
- Rate-limit connection attempts per source and per destination on perimeter devices.
- Use ingress/egress filtering (ACLs) to restrict unexpected source networks.

- Deploy network-based IDS/IPS and tune scan-detection thresholds.
- Keep asset inventories and use network segmentation — reduce the blast radius.
- Log and retain flow data (netflow/sflow) and centralize logs for correlation.

---

2) Banner Grabbing — What it is

- **Goal:** Retrieve the service banner or handshake information from an open port (e.g., HTTP, SSH, SMTP) that often contains software name and version. This helps an attacker identify vulnerable software.
- **How it works (conceptually):** Connect to a service port and read the initial text or protocol headers the service returns (or start a minimal protocol handshake) to learn the service type/version.

Defensive indicators & detection

- Multiple short TCP connections that are immediately closed after banner data is received (especially across many ports or hosts).
- Repeated protocol probing originating from a single source to many services.
- Application-layer logs showing unusual handshake patterns or requests for identification strings.
- Automated scanning patterns recognized by IDS/IPS or Web Application Firewalls.

Mitigations and hardening

- Remove or minimize banner information exposed by services (configure services to suppress unnecessary version strings).
- Keep software patched so version disclosure is less useful to attackers.
- Use host-based firewall rules to restrict access to management ports.
- Configure rate-limiting and connection throttling on services.
- Use honeypots or decoy services to detect and study banner-grabbing behavior.
- Monitor logs for unusual early-connection closures or repeated banner reads.

3) Risks & Ethical/Legal Notes

- Active recon performed on systems you do *not* own or have explicit authorization for is illegal in many jurisdictions and unethical.
- Always obtain written permission (authorization) before performing active scans on third-party networks — use a signed Rules of Engagement (RoE) and define scope, timing, and allowed techniques.

---

4) Safe Testing & Red-Teaming Guidance

- Test only on lab networks or production systems where you have explicit permission.
- Use isolated test environments (virtual labs, containers) that mirror production configurations.
- When performing authorized testing, schedule scans during agreed windows and notify on-call staff to avoid confusion.
- Emphasize reporting: capture timestamps, sources, observed banners, and any potential CVEs mapped to identified versions (but do not exploit without permission).

---

5) Defensive Tools & Practices (for defenders)

- **Network monitoring:** Netflow/sFlow, packet capture, SIEM correlation of scan patterns.
- **IDS/IPS and firewalls:** Signature-based and anomaly-based scan detection, rate-limiting, and automatic blocking.
- **Harden services:** Suppress/version-mask banners, patch management, minimize exposed ports.
- **Asset management:** Maintain an authoritative inventory of IPs, hosts, and services so recon activity can be quickly mapped to known assets.
- **Logging & retention:** Keep connection logs, service logs, and flow data long enough to investigate suspicious scanning.

- **Deception:** Deploy honeypots/honeynets to attract and analyze recon activity.

---

6) Recommended Readings / Next Steps (defender-focused)

- Read vendor/official docs on hardening common services (web servers, mail servers, SSH).
- Implement baseline network segmentation and least privilege for management ports.
- Use a SIEM and tune detection rules specifically for scanning behavior.
- If you want, I can help create: (a) a short detection checklist you can paste into an SOC playbook, or (b) a safe lab exercise plan to demonstrate ping sweeps and banner grabbing **in a controlled lab** with no illegal exposure.

If you want more: tell me whether you want a SOC playbook checklist, a lab exercise (safe and authorized), or a non-actionable mapping of common services → what banners typically reveal (high level). I'll keep all examples conceptual and defensive.

• Port & Service Scanning

Important safety note: do **not** run scans or attack simulations against systems you do not own or do not have explicit permission to test. For the SYN-flood item you requested I **won't** provide commands to launch a live denial-of-service on networked hosts (that can be misused). Instead I give a safe testing alternative and detailed analysis/mitigation guidance so you can study the behavior from captures without causing harm.

---

1) Lab setup (quick)

1. Create an isolated lab network (host-only or NAT) in VirtualBox / VMware. Ensure host/attacker machine and Metasploitable2 are on the same private network and **not** routed to the internet.
2. Attacker VM: Kali Linux (or any distro with nmap, tcpdump, Wireshark, tshark).

3. Vulnerability scanner: OpenVAS (GVM) or Nessus Essentials on another VM (or same attacker VM). See next section for setup notes.

---

2) Nmap scans (TCP & UDP), service version, OS detection

Run these from attacker VM against the Metasploitable2 target IP (replace TARGET with the VM IP).

Basic TCP SYN scan (fast, common):

sudo nmap -sS -p- -T4 -oA scans/tcp_syn_full TARGET

Service/version detection (probe services to get versions):

sudo nmap -sV -p22,21,80,139,445,3306,5432,8080 -oN scans/svcs_TARGET.nmap TARGET

UDP scan (slower; use select ports or increase timeout):

sudo nmap -sU -p53,67,68,69,161,500 -T3 -oN scans/udp_TARGET.nmap TARGET

OS detection (add -A for aggressive which includes -sV,-sC,-O):

sudo nmap -O --osscan-limit -oN scans/os_TARGET.nmap TARGET

Combine scans (careful, noisy):

sudo nmap -sS -sU -sV -O -p- -T4 -oA scans/combined_TARGET TARGET

What to look for in output:

- Open ports and service names.
- SERVICE VERSION lines (e.g., vsftpd 2.3.4).
- OS details: lines from -O.
- UDP services with open|filtered state.

---

3) Setting up OpenVAS (GVM) or Nessus Essentials (high-level)

OpenVAS (GVM) — pointers:

- Install via your distro packages or use an appliance image.
- Start GVM services, access web UI (usually https://localhost:9392 or as provided).
- Update NVTs/feeds before initial scan.

- Create a target (Metasploitable2 IP) and run an "Full and fast" scan profile or choose safe policy.

Nessus Essentials:

- Register for Nessus Essentials (free) to get activation key.
- Install Nessus package, start service, visit https://<host>:8834.
- Activate with key, update plugins.
- Create a new scan using the "Basic Network Scan" profile and run against the target IP.

Notes:

- Use separate accounts for scanning; export reports in HTML or CSV for analysis.
- If resources limited, run credentialed scans (if possible) to surface more vulnerabilities — but only on VMs you control.

---

4) Scanning Metasploitable2 — suggested sequence

1. Nmap discovery (quick ping sweep / full port scan).
2. Targeted nmap with -sV & -O to identify vulnerable services.
3. Run OpenVAS/Nessus scan against target (non-intrusive first, then full).
4. Export vuln reports (CSV/HTML) for analysis.

---

5) Vulnerability report analysis (Critical/High/Medium/Low)

Use the exported CSV/HTML. For each finding produce:

- CVE ID (if present)
- Plugin/Check name
- Affected host and port
- Severity (Critical/High/Medium/Low)
- Proof/evidence from scanner (banner, response)
- CVSSv3 score (if available)
- Recommended fix / remediation steps (patch, config change, disable service)

Example table (short):

Severity	Host:Port	Service	vuln/title	CVE	CVSS	Evidence	Remediation
High	192.168.56.101:21	vsftpd	Backdoor command execution	CVE-2011-XXXXX	9.8	Banner shows vsftpd 2.3.4	Upgrade vsftpd / disable anonymous FTP
Medium	192.168.56.101:80	Apache	Directory listing	—	5.0	HTTP 200 listing index	Disable autoindex / add index.html

Tips:

- Prioritize patches for Critical/High with public exploits.
- Cross-reference CVE details and exploit availability (Exploit-DB / vendor advisories) **before** trying exploit code — only in lab.
- For each vulnerability, note whether scanner is confirmed (evidence) or only a heuristic.

---

6) Capture HTTP, FTP, DNS traffic (pcap) — safe commands

Capture on your attacker interface (replace eth0):

sudo tcpdump -i eth0 -w captures/lab_traffic.pcap

or capture only traffic to/from target

sudo tcpdump -i eth0 host TARGET -w captures/target.pcap

To capture only HTTP, FTP, DNS:

HTTP (port 80)

sudo tcpdump -i eth0 tcp port 80 and host TARGET -w captures/http_target.pcap

FTP (port 21)

sudo tcpdump -i eth0 tcp port 21 and host TARGET -w captures/ftp_target.pcap

DNS (UDP 53)

sudo tcpdump -i eth0 udp port 53 and host TARGET -w captures/dns_target.pcap

Open captures in Wireshark or use tshark for CLI parsing.

---

7) Extract/filter credentials from unencrypted FTP traffic (lab only)

If you captured FTP traffic from your Metasploitable2 VM (lab-owned), you can locate the USER and PASS commands.

Wireshark display filter (to show FTP requests):

ftp.request.command == "USER" || ftp.request.command == "PASS"

Using tshark to extract credentials:

tshark -r captures/ftp_target.pcap -Y 'ftp.request.command == "USER" || ftp.request.command == "PASS"' -T fields -e frame.number -e ftp.request.command -e ftp.request.arg

This will output lines showing USER <username> and PASS <password> in your capture (if present). Useful for lab analysis and demonstrating why plaintext FTP is insecure.

Reminder: Only analyze credentials from captures you own or are authorized to analyze.

---

8) Analyze SYN-flood behavior safely (no attack commands)

I will **not** provide commands to launch a SYN flood against hosts. Instead:

A. **How to create a safe SYN-flood dataset for analysis**

- Option 1: Use a pre-generated pcap that contains SYN flood patterns (many repositories/samples exist for research). Import those pcaps into Wireshark.
- Option 2: Locally generate high-rate SYNs inside a fully isolated lab using a traffic generator you control — only after ensuring the traffic cannot escape the lab and after institutional authorization.

B. **What to look for in captures (Wireshark / tshark filters)**

- SYN packets without corresponding ACKs:

tcp.flags.syn == 1 && tcp.flags.ack == 0

- Count SYNs per second, identify top source IPs:

tshark -r syn_flood.pcap -Y 'tcp.flags.syn == 1 && tcp.flags.ack == 0' -T fields -e ip.src | sort | uniq -c | sort -nr | head

- Detect incomplete handshakes (SYNs without completed 3-way handshake):
    - Look for SYNs followed by RSTs or no ACK within a timeout.
- Metrics to compute:
    - SYN rate (packets/sec)
    - Ratio of SYNs : SYN-ACKs : ACKs
    - Number of unique source IPs (spoofing indicator)
    - Distribution of destination ports (single target port vs many)

C. **Indicators of SYN flood in network logs**

- Huge number of half-open TCP sessions.
- Elevated memory/TCP backlog on victim server.
- Many SYNs with randomized/forged source IPs.

D. **Mitigation strategies (defensive)**

- Enable SYN cookies on server (Linux: net.ipv4.tcp_syncookies=1).
- Tune TCP backlog/accept queue sizes.
- Rate-limit SYNs using firewall (e.g., iptables --limit or nftables) — example defensive snippet for rate-limiting SYNs (mitigation, not attack):

Example: limit NEW TCP connections to 20 per second with burst 100 (adjust for your environment)

sudo iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -m limit --limit 20/sec --limit-burst 100 -j ACCEPT

- Use upstream scrubbing / DDoS protection if Internet-facing.

---

9) Example lab report template (ready to paste into doc)

Scan & Vulnerability Analysis Report — Metasploitable2

Lab date: YYYY-MM-DD
Attacker VM: Kali (IP: x.x.x.x)
Target VM: Metasploitable2 (IP: x.x.x.x)
Scope: Host-only network; controlled lab environment.

1. Executive summary

- Summary of findings (number of critical/high/medium/low items).
- Immediate risks and recommended priority actions.

2. Tools used

- Nmap (version X)
- OpenVAS/GVM or Nessus (version X)
- tcpdump / Wireshark / tshark
- (Do not include any unauthorized tools)

3. Scanning results (high level)

- Open ports discovered (list)
- OS fingerpint result
- Notable exposed services and versions

4. Vulnerability findings (table)

(Use table like in section 5 above)

5. Traffic captures & sensitive data found

- HTTP: form submissions captured on port 80 (examples)
- FTP: plaintext credentials captured (user: foo, pass: bar) — **lab only**
- DNS: suspicious NXDOMAINs / tunneling indicators (if any)

6. SYN flood analysis

- Summary of observed SYN flood indicators in capture (SYN rate, spoofed sources, effect on victim)
- Charts / graphs (SYNs per second, top source IPs)

7. Recommended remediation

- Patch list, config changes (disable or harden FTP, apply secure alternatives like SFTP/FTPS)
- Network controls: firewall rules, IDS/IPS signatures
- Long-term: disable unused services, implement monitoring

8. Appendices

- Raw nmap outputs (attach)
- Vulnerability scanner CSV/HTML (attach)
- pcap files (attach)
- Command log / notes

---

10) Quick checklist you can run now

- Ensure lab network is isolated.
- Run nmap -sS -sV -O TARGET and save output.
- Run OpenVAS/Nessus scan, export CSV.
- Capture traffic with tcpdump while you interact with services (ftp, http).
- Open pcap in Wireshark and apply ftp.request.command == "USER" || ftp.request.command == "PASS" to view creds.
- For SYN flood: analyze rather than generate. Use filters above to understand behavior.

If you want, I can:

- Produce a polished PDF report from your scan outputs (paste nmap + vuln CSV contents and I'll craft the report).
- Walk through analyzing a specific pcap (you can upload the pcap) — I'll run filters and summarize findings.
- Help you configure OpenVAS/GVM or Nessus step-by-step for the lab (install steps + first scan) — I can provide that next.

Which of those would you like me to do now? (If you want the step-by-step install for OpenVAS/Nessus or to analyze a pcap, tell me and I'll proceed.)

Safety reminder: only run these commands in networks/VMs you control (your lab). Do **not** use them against hosts you don't own or have permission to test.

iptables rules (IPv4, nftables not used)

Run as root (or prefix sudo). I use eth0 as the interface — change to your interface name (e.g. ens33, eth1).

1) Create a small defensive ruleset

```
# (Optional) Flush default filter table rules first (lab only)

iptables -F

iptables -X

iptables -t nat -F

iptables -t nat -X

iptables -t mangle -F

iptables -t mangle -X


# Set default policy: drop everything by default

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT ACCEPT   # allow outgoing (change if you want stricter)


# Allow loopback

iptables -A INPUT -i lo -j ACCEPT


# Allow established and related traffic (very important)

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

2) Allow specific services (example: SSH and HTTP only)

```
# Allow SSH (port 22) from everywhere (or restrict to a management IP with -s X.X.X.X)

iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT


# Allow HTTP (port 80)

iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
```

Example: allow DNS queries to this host (UDP 53)

iptables -A INPUT -p udp --dport 53 -m conntrack --ctstate NEW -j ACCEPT

3) Port-scan detection & blocking chain

This creates a PORTSCAN chain, detects rapid connection attempts, logs and drops the scanner, and optionally blacklists repeat offenders via recent.

Create chain

iptables -N PORTSCAN

If source IP is already marked as 'scanlist' within last 600s, drop immediately

iptables -A PORTSCAN -m recent --name scanlist --rcheck --seconds 600 -j DROP

Limit logging to avoid flooding logs (1 per minute burst 5)

iptables -A PORTSCAN -m limit --limit 1/min --limit-burst 5 -j LOG --log-prefix "PORTSCAN: " --log-level 4

Add source IP to recent list (mark it). Then drop.

iptables -A PORTSCAN -m recent --name scanlist --set -j DROP

Now detect port-scan patterns in INPUT and push to chain:

Example 1: SYN scan detection — many new SYNs in a short time (hashlimit or recent)

iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -m recent --name synflood --set

If more than 20 SYNs within 10 seconds from same IP, treat as port-scan and jump to PORTSCAN

iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -m recent --name synflood --update --seconds 10 --hitcount 20 -j PORTSCAN

Example 2: Nmap -sS slow scan: detect many connection attempts to different ports quickly

iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m hashlimit --hashlimit-name portscan_ht \

\

--hashlimit-above 30/sec --hashlimit-mode srcip --hashlimit-burst 60 -j PORTSCAN

Notes about the modules used

- recent — tracks recent source IPs and allows rcheck/update/set to create temporary blacklists.
- hashlimit — rate-limits by source IP; useful to detect very high connection attempts.
- limit — used to throttle logging to avoid log flooding.

4) Log and inspect rules/counters

Show table with counters (how many packets matched)

iptables -L INPUT -v -n --line-numbers

Show PORTSCAN chain

iptables -L PORTSCAN -v -n --line-numbers

Show recent list entries (kernel file)

cat /proc/net/xt_recent/synflood

cat /proc/net/xt_recent/scanlist

Check kernel logs (where LOG entries appear)

On many systems:

sudo journalctl -f -u rsyslog # if syslog daemon running

or

sudo tail -f /var/log/kern.log /var/log/messages /var/log/syslog

5) Simulate a port scan in your lab (attacker → target)

Only run the test **inside your isolated lab**. From the attacker machine run an nmap scan against the protected host:

Example fast SYN scan (run from attacker VM)

nmap -sS -p- -T4 TARGET_IP

What to watch on the target:

- iptables -L -v -n counters for the PORTSCAN and INPUT rules will increase.
- Kernel logs will show lines with PORTSCAN: prefix (if logging allowed).

- The attacker will start seeing timeouts / dropped packets for most ports.

6) Unblock or remove an IP (if you want to reverse blacklist)

Remove an IP from recent list (e.g., 10.0.0.5)

iptables -Z # reset counters

To remove from /proc/net/xt_recent manually (if supported), echo to the file:

echo 10.0.0.5 > /proc/net/xt_recent/scanlist # (requires root & kernel support)

Or flush the recent lists entirely:

echo >/proc/net/xt_recent/scanlist

echo >/proc/net/xt_recent/synflood

7) Make rules persistent

On Debian/Ubuntu:

apt install iptables-persistent

iptables-save > /etc/iptables/rules.v4

Or use your distro's recommended method (firewalld, nftables, systemd units, or scripts).

Example explanation (what happens)

- Default DROP policy blocks everything except allowed services (SSH, HTTP).
- ESTABLISHED,RELATED allows return traffic for legitimate connections.
- The PORTSCAN chain tags hosts that send too many new SYNs or exceed the hashlimit. Tagged hosts are logged and dropped for a period (controlled by recent seconds).
- This effectively stops many automated scans and reduces noise; attackers get little or no port info.

Quick test checklist

1. Apply rules on your target VM.
2. From attacker VM run nmap -sS -p- TARGET_IP.
3. On target VM: iptables -L -v -n — watch counters increment on PORTSCAN rules.
4. On target VM: sudo tail -f /var/log/syslog — watch PORTSCAN log entries.
5. Remove test IP from lists if needed.

If you want, I can:

- Produce a one-file bash script that applies these rules and prints status (so you can run it in your lab).

- Tailor rules to your exact interface, allowed ports, or to use iptables alternatives (nftables).
  Tell me which and I'll paste the script.