5. Scanning Networks

1. Introduction

1. Transition from Reconnaissance to Scanning

- After gathering information during reconnaissance (e.g., identifying IP ranges, domain names, etc.), the next step is to **interact directly with the systems**.
- If working closely with the client, you may already have access to some of this data. Still, you must always **stay within the agreed-upon scope** of the test.

2. Ethics and Permission

- Even if you're "just scanning" and not hacking or exploiting systems, you **must get clear permission** from the target.
- Some systems (especially old or fragile ones) can **crash from something as simple as a port scan**.
- Inform the client about **possible risks**, and prepare for unexpected system behaviors.

3. Purpose of a Port Scan

- A **port scan** finds open ports on systems—this tells you which services or applications might be running.
- It's not just about finding open ports, but about discovering **what's behind them** (like an HTTP server or FTP service) so you can identify potential vulnerabilities.

4. Using Vulnerability Scanners

- Once you know what services are running, you can use a **vulnerability scanner** to find known weaknesses in those services.
- However, scanners are **not always reliable**—they may:
  - Miss real issues (false negatives)
  - Report fake ones (false positives)
- So, **manual verification by a skilled ethical hacker is still crucial**.

5. Challenges from Security Devices

- **Firewalls** and **Intrusion Detection/Prevention Systems (IDS/IPS)** can block scans or detect and alert security teams.

- For stealth operations (like red team exercises), **evasion techniques** may be needed to avoid detection.

6. Packet Crafting

- One stealthy technique is **packet crafting**—manually building network packets in a way that **bypasses normal OS rules**.
- This can confuse or evade firewalls and security systems because **the packets don't follow standard structures**, helping them sneak through.

7. MITRE's Classification

- The **MITRE ATT&CK framework** defines **active scanning** as a common attacker technique, with two subtypes:
  - Scanning IP blocks
  - Vulnerability scanning
- The passage notes that **this classification is useful**, but **it simplifies the complexity** of real-world scanning tactics and tools.

🔑 Summary:

- **Scanning** is the first hands-on step in ethical hacking.
- It must be done **ethically and with permission**, as it can cause disruptions.
- Tools like **port scanners**, **vulnerability scanners**, and **packet crafting tools** are used to explore systems.
- The goal is always to **identify and report risks**, not exploit them.
- Good hackers don't just rely on tools—they **analyze results critically and ethically**.

2 . Ping Sweep

🔍 What is a Ping Sweep?

A **ping sweep** involves sending **ICMP echo requests** (commonly known as "pings") to a range of IP addresses to determine which hosts are "alive" or responsive.

- **Responsive hosts** reply to the ICMP echo.
- **Unresponsive hosts** may be offline or may have firewalls that block ICMP.

🛠️ Tools for Ping Sweeps

Here's a clear explanation and comparison of **fping** and **MegaPing**:

🛠️ 1. fping

🔹 Overview:

- **Command-line tool** (Linux/Unix).
- Sends **ICMP echo requests** to multiple hosts to check if they're alive.
- Designed for speed and efficiency in network scanning.

🔹 Key Features:

- Supports **pinging a range or subnet** of IPs.
- Displays **only alive hosts** (`-a` flag).
- Shows **round-trip time** (`-e` flag).
- Can generate IP list from subnet (`-g` flag).

🔹 Example Command:

```
fping -aeg 192.168.1.0/24
```

- `-a`: Show only alive hosts.
- `-e`: Show elapsed (response) time.
- `-g`: Generate IPs from subnet.

🔹 Pros:

- Lightweight and fast.
- Scriptable for automation.
- Ideal for bulk ping sweeps in terminal environments.

🔹 Cons:

- No GUI.
- Only does ICMP ping (no additional tools like port scanning).

🖥️ 2. MegaPing

🔹 Overview:

- **GUI-based tool** for Windows.
- Offers a wide range of **network utilities**, including ping sweep, port scanning, NetBIOS enumeration, and more.

🔹 Key Features:

- **IP Scanner** performs ping sweeps.
- Displays:
  - Hostname
  - MAC address
  - Response time
- Additional tools for:
  - Port scanning
  - Shares & users detection
  - SNMP info gathering

🔹 How to Use:

- Open MegaPing.
- Select **IP Scanner** from the left panel.
- Input IP range and start scan.
- Use checkboxes to show extra info like MAC addresses.

🔹 Pros:

- Easy to use with GUI.
- Multifunctional – combines several tools in one.
- Visual output with detailed host information.

🔹 Cons:

- Windows-only.
- Heavier and less scriptable than fping.

📊 Comparison Table:

Feature	fping	MegaPing
Interface	Command-line	Graphical (GUI)
Platform	Linux/Unix	Windows
ICMP Ping Sweep	✅ Yes	✅ Yes
Shows Response Time	✅ Yes (`-e` flag)	✅ Yes
Shows MAC Address	❌ No	✅ Yes (optional checkbox)
Port Scanning	❌ No	✅ Yes
Lightweight/Fast	✅ Very fast	❌ Heavier
Automation	✅ Easy with scripts	❌ Manual
Good for Beginners	❌ CLI-based	✅ GUI is beginner-friendly

✅ Summary:

- Use **fping** if you want **speed, scripting, and CLI control**.
- Use **MegaPing** if you prefer a **GUI with multiple built-in tools** for broader scanning and enumeration.

⚠️ Limitations of Ping Sweeps

- **Not foolproof:** Firewalls or network security settings may block ICMP traffic.
- **False negatives:** No response ≠ host is down.
- **Basic information:** Only tells you if host is alive, not what services it's running.

💡 Takeaway

Ping sweeps are a useful **first step** in network reconnaissance. They help estimate the number of potential target systems, but to gather deeper insights (like open ports or running services), **port scanning** and **enumeration** tools are needed.

3. Port Scanning

Port Scanning is the process of identifying open ports on a networked device (like a computer, server, or router). Ports are points of entry or exit for network communication on a device, and each port corresponds to a specific application or service. There are several types of port scanning, especially using tools like **Nmap**, to discover open ports, running services, and even gather information about the operating system and applications.

Key Concepts:

- **Port**: A port is a communication endpoint used by applications over the network. The operating system binds applications to specific ports, allowing the system to handle messages addressed to those ports.
- **Open Port**: If an application is listening on a port, that port is considered open.
- **Closed Port**: If no application is listening on a port, it is closed.
- **TCP** and **UDP**: These are the two main transport protocols used for network communication. TCP ports use a three-way handshake to establish connections, while UDP doesn't have a handshake and simply sends packets.

1. nmap

This section describes `nmap` (Network Mapper), a widely-used port scanning tool that helps administrators and cybersecurity professionals assess networks and devices. Here's a summary of the key points:

Key Features of `nmap`:

1. **Port Scanning**:
   - `nmap` can scan both TCP and UDP ports.
   - By default, it scans 1,000 commonly used ports, but users can specify custom port ranges.
   - `nmap` can scan a single IP or a range of IPs (using network blocks or CIDR notation).
2. **TCP Scanning**:
   - **SYN Scan**: A half-open scan that sends a SYN request to a port. If the port is open, the target sends a SYN/ACK response, which `nmap` acknowledges with a RST, indicating it doesn't want to complete the connection. Closed ports send an RST directly.
   - **Full Connect Scan**: A more thorough scan that completes the connection to the port before tearing it down.
   - **Xmas Scan**: Uses the FIN, PSH, and URG flags to probe for open or filtered ports. Open ports typically don't respond, and closed ports send a RST.

    - **FIN Scan**: Similar to Xmas, this scan uses the FIN flag to determine open or filtered ports.
3. **UDP Scanning**:
    - Unlike TCP scanning, UDP scanning doesn't involve complex flags and instead checks for the absence of response or an ICMP port unreachable message to determine whether a port is closed.
    - The throttle rate can be adjusted with the `-T` option for faster or slower scans.
4. **Service Version Detection**:
    - `nmap` can detect the version of services running on open ports with the `-sV` option by retrieving banner information from applications like OpenSSH, Apache, etc.
    - This helps identify the software and version running on a particular port.
5. **Operating System Detection**:
    - `nmap` uses fingerprints to identify the operating system running on a target system. The operating system is identified based on network stack behavior, but multiple OS versions can match the same fingerprint.
6. **Scripting Engine**:
    - `nmap` has a powerful scripting engine (written in Lua) that can extend its functionality. There are hundreds of predefined scripts in various categories like discovery, brute force, malware, and vulnerability scanning.
    - The `--script` option allows users to specify which script or category of scripts they want to run during the scan.

Practical Use:

- **Basic Scanning**: Scanning a target IP with basic TCP or UDP scanning.
- **Full Connect Scan**: Scanning an entire subnet and specifying which ports to scan (e.g., ports 80 and 443).
- **Operating System and Service Detection**: Running version and OS detection scans to understand the services and systems in use.
- **Running Scripts**: Using the scripting engine to run discovery or vulnerability scans.

`nmap` is a versatile tool, ideal for discovering open ports, determining running services, and identifying system configurations. It is often used for network discovery, penetration testing, and security auditing.

2. Zenmap

Zenmap is the official graphical user interface (GUI) for Nmap, a widely-used network scanning tool. While Nmap is command-line-based, Zenmap simplifies the scanning process with a user-friendly interface that makes it more accessible, especially for those who prefer working with GUIs.

Key Features of Zenmap:

1. **Scan Types:** Zenmap provides a list of predefined scan types (e.g., Intense Scan, Quick Scan, Regular Scan), which allow users to easily select the scan they wish to perform. Each scan type has different parameters, such as throttle speed, to suit the user's needs.
2. **Command Line Integration:** Even though Zenmap provides a GUI, it runs Nmap in the background. Users can also customize the command line to fit specific needs, giving the flexibility of Nmap while benefiting from the GUI.
3. **Visualizing Results:** One of the significant advantages of Zenmap is its ability to visualize scan results. Instead of parsing through raw text output from the command line, Zenmap organizes and presents the results clearly. The visualization includes:
   - Hosts identified in the scan.
   - Icons indicating the operating system types.
   - A list of services and applications detected during the scan.
   - Option to view detailed information about services and applications running on each host.
4. **Service List and Version Info:** Zenmap categorizes the detected services and allows users to view them in detail. Users can also see application names and versions, which helps in identifying potential vulnerabilities.
5. **Saving and Comparing Scans:** Zenmap allows users to save their scans in XML format, making it easier to analyze and compare different scans over time. By saving multiple scans, users can track network changes and identify new vulnerabilities or configuration changes.
6. **XML-based Comparison:** The ability to compare two saved scans is a valuable feature. Zenmap compares XML files node by node to identify any differences between them, providing insights into changes that may have occurred on the network.

Overall, while experienced users may prefer using Nmap's command-line interface, Zenmap's visualization and organizational features make it a powerful tool for those who want a more intuitive way of analyzing network scan results.

⚡ Masscan

Overview:

- A lightning-fast port scanner designed to scan large IP ranges quickly.
- Created by Robert Graham for high-speed scanning (e.g., Internet-wide scans).

Key Features:

- **Blazing Speed:** Scans 100,000+ packets per second with the `--rate` flag.
- **Randomization:** Uses `--randomize-hosts` to avoid detection and throttling.
- **Simple Syntax:** Similar to Nmap CLI; supports `-p` for port list.
- **Banner Grabbing:** Use `--banners` to get limited service info.
- **Limited Scan Types:** Only supports SYN (TCP) scans; no UDP, no advanced scans (e.g., Xmas, FIN).

Use Case:

- Perfect for high-speed reconnaissance or scanning large networks (e.g., full Class B).
- Not suited for deep service enumeration or OS detection.

4 .MegaPing

Here's a summary of the key features and limitations of **MegaPing** based on your input:

✅ Key Features of MegaPing

1. **Multiple Scanning Capabilities**:
    - Not limited to ping sweeps—also supports port scanning.
    - GUI-based tool, making it beginner-friendly.
2. **Predefined Port Sets**:
    - Offers **preselected port groups** like:
        - **Hostile Ports**: Ports often misused by malware (e.g., Trojan ports).
        - **Authorized Ports**: Ports typically used by standard applications.
    - Helps focus scans on more security-relevant areas.
3. **Detailed Port Information**:
    - Identifies **elevated ports** (e.g., port 80), which require admin rights and are prime targets for attackers.
4. **Visual and Interactive Interface**:
    - You can add IPs manually, and scan results are shown in a user-friendly way.

❌ Limitations of MegaPing

1. **No CIDR or IP Range Support**:

    - Cannot scan subnets (like `192.168.1.0/24`) or IP ranges (e.g., `192.168.1.1-192.168.1.100`).
    - All IPs must be added **manually**, which is inefficient for large networks.
2. **Less Advanced than Nmap**:
    - Lacks advanced scan types (e.g., FIN, Xmas, ACK).
    - Doesn't support command-line scripting or automation.
    - Not ideal for large-scale or stealthy scans.

🛠 Use Case Summary

- **Ideal for**:
    - Beginners and Windows users needing a simple UI.
    - Quick scans of a few IPs.
    - Focused port scanning with a security lens (e.g., hostile ports).
- **Not suitable for**:
    - Full-scale or automated network scanning.
    - Users needing flexible input (CIDR, range).
    - Deep or custom scan types.

5. Metasploit

Sure, Kartik — here's a clear and structured explanation of **Metasploit**, especially focused on what it is, what it can do, and how it's used in cybersecurity.

💣 What is Metasploit?

Metasploit is an open-source **penetration testing framework** used by:

- Ethical hackers
- Security researchers
- System admins

Its main job is to **identify**, **validate**, and **exploit vulnerabilities** in computer systems.

🔧 What Can You Do with Metasploit?

Function	Description
🔍 **Scanning**	Find open ports, services, and known vulnerabilities.
🛠 **Exploitation**	Use pre-written **exploits** to take control of systems with known flaws.
📡 **Payload Delivery**	Send malicious code (**payloads**) to a target machine to open backdoors or gain shell access.
🔄 **Post-Exploitation**	Once inside, gather passwords, escalate privileges, or pivot to other machines.
📊 **Reporting**	Log and store results for audits or reports.
🧪 **Testing Custom Exploits**	Write your own exploits using Ruby (Metasploit's scripting language).

🧱 Key Components of Metasploit

Component	Role
msfconsole	The main command-line interface for using Metasploit.
Modules	Reusable components like exploits, scanners, and payloads.
Payloads	Code that runs after exploiting a system (e.g., a reverse shell).
Listeners	Wait for a connection back from the exploited target (e.g., Meterpreter).
Auxiliary Modules	For tasks like scanning, fuzzing, or sniffing—not exploitation.
Database	Stores scan results and session data across multiple runs.

🔐 Ethical Use Only

> Metasploit is a **powerful hacking tool**.
> It's **legal only when used with permission** — like in:

- Penetration testing (with client's consent)
- Hacking your own lab or network
- Cybersecurity training

🚀 Fun Fact: Meterpreter

Meterpreter is Metasploit's advanced payload:

- Runs **in memory** (stealthy)
- Allows **file transfer**, **webcam access**, **keylogging**, etc.
- Helps **maintain control** of the target after exploitation

4 . Vulnerability Scanning

🔍 Vulnerability Scanning: Overview

1. **Initial Steps**:
   - Discover **open ports** and **listening applications**.
   - Research associated **vulnerabilities**.
   - Avoid blindly throwing exploits—this can cause **unexpected failures**.
2. **Risks of Blind Testing**:
   - Can unintentionally **harm systems**.
   - May **crash services** or **cause outages**.
   - Unethical and unprofessional without **client consent** and **impact awareness**.
   - In **red team testing**, random noisy attacks increase chances of **detection** and **test failure**.
3. **Smart Approach**:
   - Use a **vulnerability scanner**:
     - Identifies open ports and apps.
     - Maps known vulnerabilities to them.
     - Runs **specific, defined tests** (non-exploitative).
4. **Important Note**:
   - Scanner results must be **verified manually**.
   - It may:
     - Misidentify (False Positive).
     - Miss real issues (False Negative).
   - Scanners are a **starting point**, not the end.

📊 Types of Scanner Results:

Type	Meaning
False Positive	Reported vulnerability that isn't real after verification.
False Negative	Missed vulnerability that actually exists.
True Positive	Real vulnerability correctly identified.
True Negative	No vulnerability reported, and none exists.

🕰 History:

- **1990s**: First scanner - **SATAN** (by Dan Farmer & Wietse Venema).
- Followed by tools like:
  - **SARA**
  - **SAINT**
- Modern tools like **Nessus** (very popular), and **OpenVAS** (open-source, related to Nessus).

1. OpenVAS

OpenVAS (Open Vulnerability Assessment System) is a **free, open-source vulnerability scanner** used to detect security issues in computer systems and networks. It helps identify weaknesses like outdated software, misconfigurations, or unpatched services that could be exploited by attackers.

🔍 Key Features of OpenVAS

1. History

- It began as a **fork of Nessus v2**, which was open source until 2005.
- When Nessus became proprietary, OpenVAS continued as a community-driven project.
- It's now maintained by **Greenbone Networks**, and the user interface is called **Greenbone Security Assistant (GSA)**.

🧱 Core Components

✓ GSA (Greenbone Security Assistant)

- A **web-based GUI** to manage scans and view results.

✓ Targets

- You specify **IP addresses or hostnames** to scan.
- You can **exclude** certain IPs (e.g., fragile systems).
- You can define which **ports** to scan and even provide **credentials** for deeper, authenticated scans.

✓ Credentials

- OpenVAS can use **username/passwords or SSH keys** to log into systems and detect **local vulnerabilities**.
- Useful for authenticated scans (more accurate results).

✓ Scan Configs

- Predefined or custom configurations that determine **which vulnerability tests (NVTs)** to run.
- NVTs are organized into **families** (e.g., Cisco, Windows).
- You can build your own config from scratch or customize existing ones like **Full and Fast**.

✓ Scan Tasks

- Combines the **target**, **scan config**, and **schedule**.
- Can be run once or on a recurring schedule.
- Results are stored and visualized within the GSA interface.

🛠️ How It Works (Workflow)

1. **Create Target** → Define IPs and scan ports.
2. **Create Credentials** (optional) → For authenticated scanning.
3. **Choose or Create Scan Config** → Select what tests to run.
4. **Create Task** → Combine target and config.
5. **Run Scan** → View detailed vulnerability reports.

✅ Advantages of OpenVAS

- Completely **free and open-source**.
- **Regularly updated** NVTs (Network Vulnerability Tests).
- **Web interface** is beginner-friendly.
- Can perform both **unauthenticated and authenticated scans**.
- Allows **role-based access** for multiple users.

⚠️ Important Note

After running scans, it's critical to:

- **Review the results**
- **Prioritize vulnerabilities**
- **Plan remediation**

Ignoring results can pose a **legal and security risk**, since known vulnerabilities left unpatched could lead to breaches.

2. Nesus

💡 What is Nessus?

Nessus is a **vulnerability scanner** used to find **security weaknesses** in systems, such as outdated software, weak passwords, or misconfigurations. Think of it like an antivirus, but for finding all kinds of **network and system vulnerabilities** — not just viruses.

- 🧠 Made by **Tenable**.
- 🔐 Used by **security professionals**, **network admins**, and **ethical hackers**.
- 🧑‍💻 Originally open-source, but later became a **commercial (paid)** product.
- 🆓 It has a **free version** for personal use called **"Nessus Essentials"**, but it has limits.

⚙️ How Does Nessus Work?

Nessus **scans devices on a network** to detect:

- Open ports (e.g., SSH, HTTP)
- Running services and their versions

- Known vulnerabilities (CVEs)
- Misconfigurations
- Weak credentials

🔧 How to Use Nessus (Basics)

1. **Create a Scan**
   - Click “New Scan” → choose a type like **Basic Network Scan**.
   - Name it and set the target IPs/domains.
2. **Add Credentials (Optional)**
   - You can give Nessus login credentials (SSH, Windows) to check deeper.
   - Example: For Linux, you can provide **SSH username/password** or **SSH keys**.
3. **Set Plugins**
   - Nessus uses **small scripts (plugins)** to test for different vulnerabilities.
   - Example: One plugin checks if a system is vulnerable to Heartbleed.
4. **Run the Scan**
   - Click **Launch**.
   - Nessus scans the targets and checks for thousands of known vulnerabilities.
5. **Review the Results**
   - You’ll see vulnerabilities sorted by severity:
     - **Critical**, **High**, **Medium**, **Low**, and **Info**.
   - Each result shows:
     - What the problem is
     - How to fix it (Remediation)
     - References (e.g., CVE ID, links)

🛠️ Tabs in Scan Configuration

Tab	What It Does
Discovery	How Nessus finds devices (e.g., by pinging or scanning ports).
Assessment	What kinds of issues to check (e.g., web, malware, config errors).
Plugins	Enable/disable specific vulnerability tests.
Report	Customize how results are displayed.

Tab	What It Does
Advanced	Deep control (timeouts, scan performance, etc.).

📁 Extra Info

- Nessus plugins use **NASL** (a scripting language) to define what to test and how.
- Plugin files are stored in:
  - Linux: `/opt/nessus/lib/plugins`
  - Windows: Usually inside Program Files under Nessus
- You can read plugin scripts to understand how vulnerabilities are detected.

✅ Key Benefits

- Scans **quickly** and with **high accuracy**
- Suggests **remediation** (how to fix things)
- Updated frequently with new vulnerabilities
- Offers **deep scans** if login credentials are provided

5. Packet Crafting and Manipulation

🧾 What is Packet Crafting?

Packet crafting means **manually creating or modifying network packets** instead of letting your operating system do it automatically. Normally, when you visit a website, your system builds packets in a specific, predictable way. But in **packet crafting**, you override this and design the packet *exactly* how you want — including unusual or invalid header values.

🧱 How Packets Are Normally Built (Quick Recap)

1. **You enter a URL** in your browser.
2. The browser creates an **HTTP request**.
3. The **operating system** takes over:

- Adds **TCP** header (Layer 4)
- Adds **IP** header (Layer 3)
- Sends it to the **network interface** to transmit over the internet.

> These headers (TCP/IP) contain fields like source IP, destination IP, port numbers, flags, and more.

🎯 Why Craft or Manipulate Packets?

Sometimes you want to:

- Test how a system behaves when it receives **unexpected or malformed packets**
- **Bypass security rules** (e.g., firewalls)
- **Fuzz** systems (send bad inputs to test their robustness)
- **Perform network attacks**, like:
  - SYN floods
  - IP spoofing
  - Port scanning with custom flags
  - IDS evasion

📦 Headers and Fields

- Headers at **Layer 3 (IP)** and **Layer 4 (TCP/UDP)** have fixed **binary** fields (e.g., numbers, flags).
- You can't insert random **ASCII text** or characters into these — values must be binary numbers that match field sizes (like 16-bit, 32-bit, etc.).
- Misusing these fields can crash systems, expose bugs, or bypass controls.

🔍 Use Case Examples

1. **Firewalls/IDS Testing**: Send TCP packets with weird flag combinations and see if firewalls block or allow them.
2. **Operating System Fingerprinting**: Analyze how different OSes respond to odd packets (e.g., by hping or nmap).

3. **Spoofing**: Craft packets to appear as if they're coming from a different IP address.
4. **DoS Attack Testing**: Send malformed packets to see if a system crashes (e.g., using fuzzed payloads).

🧪 Summary

- Normally, packets are auto-constructed by the OS.
- **Packet crafting** gives you full control over packet content and structure.
- It is useful for **security testing, penetration testing, research**, and **learning protocols deeply**.
- Tools like **packETH** (GUI) and **hping** (CLI) help craft packets without writing code.

1. hyping

hping – Swiss Army Knife of TCP/IP Packet Crafting

What is `hping`?

`hping` is a command-line network tool used for:

- Crafting and sending custom TCP/IP packets
- Network scanning
- Firewall testing
- Advanced pinging
- Security auditing

Key Features & Concepts

1. Protocol Modes

- `-1` → ICMP (ping-like)
- `-2` → UDP
- `Default` → TCP

2. Superuser Required

- Must be run with `sudo` or as root because it uses **raw sockets** (bypasses OS-level networking stack).

3. TCP Port Probing

- Example: `sudo hping3 -S -p 80 192.168.1.1`
    - Sends TCP **SYN** packets to port 80.
    - If the port is open, response flags: **SA (SYN+ACK)**
    - If closed: **RA (RST+ACK)**

4. Flag Manipulation

- Send unusual/illegal flag combos: `-S -F -P -A`
    - Can test firewall behavior or crash target stacks.

5. Packet Crafting

- Customize:
    - Header fields
    - Data payload: `-d <size>` or `--file <filename>`
    - TTL: `-t <value>`
    - Offset: `-O <value>`
    - Source port: `-s <port>`
    - Spoofed IP: `-a <fake IP>`

6. UDP Scanning

- Example:
  `hping3 --scan 1-1023 -2 192.168.1.1`
    - Scans UDP ports 1–1023 on the target.

Use Cases

- Replace `ping` with `hping -1`
- Check service availability on a port (like HTTP on 80)
- Stealth scanning by customizing flags
- Firewall and IDS/IPS testing
- Spoofing source IPs (testing, not for malicious use)

2. packETH

🔹 PackETH – Packet Generator and Editor

PackETH is a GUI-based Ethernet packet generator and editor primarily used for **network testing, debugging, and protocol analysis**. It allows users to **craft custom packets** at various protocol layers and send them over a selected network interface.

✅ Key Features of PackETH:

- **Graphical Interface**: Easy to use with fields for each protocol layer.
- **Layer Support**: Ethernet, ARP, IP, UDP, TCP, ICMP, VLAN, MPLS, etc.
- **Custom Packet Creation**: You can set fields like IP addresses, ports, flags, checksums manually.
- **Replay Mode**: Send packets repeatedly at specified intervals.
- **Send Mode Options**:
  - Send once
  - Send continuously
  - Send a defined number of times
- **PCAP Support**: Can import/export `.pcap` files.

📚 Use Cases:

- Simulate various network traffic for testing firewalls or IDS/IPS.
- Study how routers/switches behave with malformed or unusual packets.
- Educational purposes: teaching packet structures and protocols.

⚠️ Requirements & Permissions:

- Needs **root or admin privileges** to send raw packets.
- Must select the correct **network interface** to send packets.

3. fragroute

🔹 Fragroute – Packet Fragmentation Tool

Fragroute is a **network security testing tool** that **intercepts, modifies, and fragments outbound packets** from a host. It is primarily used to **evade intrusion detection systems (IDS)** by simulating evasion techniques attackers might use.

✅ Key Features of Fragroute:

- **Fragments outbound packets** in various ways.
- Simulates different **evasion techniques**, like:
  - Overlapping fragments
  - Tiny fragments
  - Fragment reordering
- Can **modify** packet contents, TTL, and timing.
- Works transparently, sitting between your application and the network.

📚 Use Cases:

- Test **IDS/IPS robustness** against evasion.
- Understand how **fragmentation affects packet reassembly**.
- Security research and education.

🛠️ Common Usage:

```
sudo fragroute -f rules.txt <target>
```

- `rules.txt` contains a set of rules specifying how to fragment or modify packets.

📝 Example Rules:

```
fragment 8 old
order random
tcp_seg 16
```

- `fragment 8`: breaks packets into 8-byte IP fragments.
- `order random`: sends fragments in random order.
- `tcp_seg 16`: splits TCP segments into 16-byte chunks.

⚠️ Limitations & Requirements:

- Works only on **Linux/Unix**.
- Needs **root privileges**.
- Only modifies **outgoing packets** (not incoming).
- Best used in a **lab/testing environment**.

6. Evasion Technique

Common Evasion Techniques for Bypassing Network Security

When testing an organization's defenses, such as firewalls, intrusion detection systems (IDS), or intrusion prevention systems (IPS), attackers use various **evasion techniques** to avoid detection and bypass these mechanisms. Here's a breakdown of the key techniques:

---

1. Hide/Obscure the Data

Encryption & Obfuscation:

- **Encryption**: Ensures that the data is unreadable without the proper decryption key. This helps avoid detection because the traffic appears as random data to firewalls and IDS/IPS.
- **Encoding**: Techniques like **URL encoding** (which replaces characters with their ASCII hexadecimal equivalents) can be used to hide malicious payloads in seemingly innocuous data.

---

2. Alterations (Polymorphism)

- **Polymorphism**: Changing the form of data to avoid signature-based detection systems (e.g., cryptographic hash checks). For example, if a malicious file has a unique hash, changing even a single character will generate a new hash, making detection difficult.

- This helps to evade **signature-based detection** by altering the appearance of the payload without changing its functionality.

---

3. Fragmentation

- **Fragmentation Attacks**: Attackers can fragment their packets in such a way that network security devices (like IDS) may struggle to reassemble them for inspection. The fragmentation may result in the security device being unable to detect malicious activities.
- **Tools**: **Fragroute** can be used to fragment packets to evade network security defenses.

---

4. Overlaps

- **TCP Sequence Number Overlap**: By sending overlapping TCP segments, an attacker can cause ambiguity in how the segments are reassembled. This may confuse both the target OS and the IDS.
- **Outcome**: This might lead to the IDS misinterpreting the data stream and allowing the malicious packets to pass through undetected.

---

5. Malformed Data

- **Protocol Violations**: Networks rely on specific protocols (like TCP/IP), and violating those protocols or exploiting loopholes in them can cause unexpected behavior.
- For instance, using **Xmas scans**, **FIN scans**, or **NUL scans** (as seen with **nmap**) takes advantage of protocol quirks to avoid detection while gathering useful information.

---

6. Low and Slow Attacks

- **Slow Scans**: Sending network scans at a very slow rate, such as one packet per hour, reduces the chance of detection since most IDS/IPS systems look for quick bursts of scanning activity.
- **Goal**: By spreading the attack over a long time period, an attacker reduces the likelihood of detection, making it less obvious to security systems.

7. Resource Consumption

- **Exploiting Resource Exhaustion**: Attackers can try to overwhelm the security device's resources (like **CPU** or **memory**), causing it to fail or to enter a state where it cannot process further packets.
- This can result in security devices **failing open**, where they let all packets pass through without inspection.

8. Screen Blindness

- **Overwhelming Alerts**: By generating a massive number of low-priority alerts (from benign activities), attackers can overwhelm the security team monitoring the alerts, leading them to miss more important threats.
- **Goal**: Cause **alert fatigue**, where the security team ignores or overlooks the real attack due to the overwhelming number of irrelevant alerts.

9. Tunneling

- **Traffic Encapsulation**: Tunneling involves encapsulating malicious traffic inside legitimate traffic. For example, protocols like **GRE**, **SSH**, **HTTP**, **ICMP**, or **DNS** can be used to encapsulate malicious data.
- **Goal**: The malicious traffic is hidden inside normal-looking packets, making it harder to detect by firewalls and IDS/IPS systems.

Summary of Evasion Techniques

- **Encryption & Obfuscation**: Hides data via encryption or encoding, making it unreadable to security devices.
- **Polymorphism**: Alters the data to change its signature and avoid detection by signature-based systems.
- **Fragmentation**: Splits the data into small packets to evade detection during packet reassembly.

- **TCP Sequence Overlap**: Sends overlapping packets to confuse packet reassembly on the target device.
- **Protocol Violations**: Exploits errors or loopholes in the protocol to avoid detection.
- **Low and Slow**: Spreads out attacks over time to avoid detection.
- **Resource Exhaustion**: Overwhelms the security device's resources to force it into an open state.
- **Screen Blindness**: Overloads alert systems, leading to missed threats.
- **Tunneling**: Encapsulates malicious traffic inside legitimate protocols.

```
These techniques highlight how attackers attempt to bypass modern security
measures. While many organizations are aware of these tactics, the
effectiveness of these methods can still vary depending on the specific
defenses in place. Understanding these evasion methods is crucial for both
attackers and defenders in enhancing the security posture of a network.
```

Evasion with nmap

This section describes various methods to evade detection when performing network scans with Nmap. The techniques mentioned are designed to make it harder for intrusion detection systems (IDS) and firewalls to identify or block the scans. Below are the key evasion methods discussed:

1. **Malformed Requests**: Nmap has built-in techniques to create malformed packets that should not exist in the real world. These packets may be ignored by some firewalls and IDS systems because they do not conform to standard network protocols. However, these methods are widely known, and most security systems can detect them.
2. **Idle Scan**: The idle scan is a technique where the attacker uses a third-party system (the "idle" system) to send packets to the target, effectively hiding the attacker's identity. The attacker spoofs packets to appear as if they come from the idle system. This scan works by exploiting the target's IP identification numbers, making it harder to trace the scan back to the real attacker. While some detection systems may notice the scan, the source of the scan (the attacker) remains hidden.
3. **Decoy Scans**: Nmap can create decoy traffic to hide the true source of a scan. This is done using the `-D` option, which allows the attacker to specify decoy IP addresses or have Nmap randomly generate them. The decoys make it more difficult for network defenders to determine which system is performing the actual scan.

4. **Fragmentation**: Nmap can fragment packets using the `-f` option, which may help bypass certain firewalls or intrusion detection systems. When packets are fragmented, firewalls or IDS systems may struggle to reassemble them and inspect the full packet. The scan can still be detected by certain systems, but fragmentation can reduce the likelihood of detection.
5. **Manipulating Maximum Transmission Unit (MTU)**: Setting the MTU to a smaller value forces packets to be fragmented. Some intermediate devices, like routers or firewalls, may choose to forward fragmented packets without inspecting them fully. Nmap can adjust the MTU value with the `-f` option, helping to evade detection by such devices.
6. **Spoofing the MAC Address**: Nmap allows you to spoof the MAC address of the scanning system using the `--spoof-mac` option. This can be useful if the firewall or IDS on the local network has a trust relationship based on specific MAC addresses. This method is only effective on local networks, as MAC addresses are removed by routers or firewalls in remote networks.
7. **Spoofing the Source Port**: Some firewalls or IDS systems may allow or block traffic based on the source port. For example, DNS traffic typically uses port 53. Nmap allows you to spoof the source port using the `-g` option, which can help bypass firewalls that rely on source port-based rules.

These techniques are all aimed at evading detection and filtering mechanisms used by security systems. However, it's important to note that these evasion methods are not foolproof, and modern firewalls and IDS systems may still detect and block malicious scans.

7. Protection and Detecting

In the context of network security, protecting against and detecting malicious activities such as port scanning, vulnerability scanning, packet crafting, and spoofing attacks are critical tasks. Here are some methods and strategies to detect and protect against such activities:

1. Detecting Port Scans

Port scans, including those from tools like Nmap, can be detected by monitoring for patterns of unusual or unexpected traffic:

- **Abnormal Port Requests**: Port scans tend to target multiple ports in a short period. If there are requests for many ports that aren't typically used in your environment, this is a sign of a scan.
- **Network Intrusion Detection Systems (NIDS)**: NIDS can be configured to detect port scans by watching for unusual traffic patterns (e.g., a high number of connection attempts to ports that are not typically in use).

- **Firewall Logs**: Firewalls can be configured to detect and block port scan traffic, as they may observe many requests over non-standard ports. Regular monitoring of firewall logs can help spot this kind of activity.

2. Protecting Against Port Scans

To protect against port scans, you can take the following measures:

- **Firewalls**: Configuring firewalls to block traffic on ports that aren't in use by any services on your network helps prevent unnecessary exposure.
- **Intrusion Prevention Systems (IPS)**: An IPS can block traffic that matches known attack patterns, including common scanning techniques. It can also take actions like rate-limiting or blocking IP addresses after a certain number of requests.
- **Rate Limiting**: This technique involves limiting the number of requests a host can make to specific ports in a given period, making it harder for a scanner to complete its job.
- **Port Knocking**: This technique involves requiring a specific sequence of port accesses to authenticate access to a system. This "secret handshake" can block unauthorized scans from reaching certain ports.

3. Detecting Vulnerability Scanners

Vulnerability scanners send out requests to determine which services or vulnerabilities are present on a system, and these actions often generate recognizable traffic:

- **Signature-Based Detection**: NIDS or network monitoring tools can identify the traffic patterns associated with well-known vulnerability scanners. Many scanners send specific probe packets to check for particular vulnerabilities, and these can be detected by known signatures.
- **Anomaly-Based Detection**: In addition to signature-based detection, monitoring systems can identify behaviors that deviate from the norm. For instance, a high volume of requests to multiple services or common vulnerability checks may be flagged as suspicious activity.

4. Protecting Against Vulnerability Scanners

- **Blocking Known Vulnerability Scanners**: Firewalls or IPS systems can be configured to block traffic from known scanner IP ranges. Many commercial vulnerability scanners use predictable patterns and addresses, making them detectable and blockable.
- **Patch Management**: Ensuring that systems are up to date with security patches can prevent scanners from detecting unpatched vulnerabilities.
- **Limit Service Exposure**: Reduce the number of services and ports exposed to the network. Services that are not needed should be disabled or moved behind additional layers

of defense (e.g., VPN or proxy).

5. Detecting Packet Crafting and Spoofing

Packet crafting, like in IP spoofing or crafting malicious packets, can often appear as if traffic is coming from a trusted source or a source that it's not. Detection methods include:

- **Spoofed Source Address Detection**: If packets arrive from addresses that shouldn't be coming from a certain network (e.g., private IP addresses from the public internet), the system should flag these as suspicious.
- **Perimeter Firewalls**: A firewall should block packets with source addresses that fall within ranges that shouldn't be routable from the outside (e.g., private IP address ranges as defined by RFC 1918). This ensures that packets from these addresses cannot come from the internet.

6. Protecting Against Spoofing Attacks

- **Source Address Filtering**: Configure firewalls to drop packets from the public internet that claim to come from private IP ranges. This can prevent many spoofing attacks, especially in the case of IP spoofing.
- **Ingress and Egress Filtering**: This is a method used to detect and block spoofed packets based on their source IP. For instance, a network may apply filtering rules to prevent packets with internal IP addresses from being sent from external interfaces.
- **Use of Anti-Spoofing Techniques**: Implementing ingress filtering (ensuring that packets entering the network have valid source addresses) and egress filtering (ensuring packets leaving the network have valid source addresses) can help mitigate spoofing attacks.

7. Additional Techniques to Detect and Protect

- **Behavioral Analysis**: Monitoring systems that track typical user and device behavior can help detect anomalies such as sudden, unusual traffic spikes or unexpected requests to odd ports.
- **Deep Packet Inspection (DPI)**: DPI tools can inspect the contents of network packets for signs of suspicious behavior, such as malformed packets or known attack patterns, helping to detect packet crafting attacks.
- **Honeypots**: Deploying decoy systems (honeypots) that seem like real targets can help detect and analyze scanning or attack attempts in a controlled environment. Honeypots attract attackers, allowing you to monitor their activities and gather intelligence on attack methods.

By combining these detection and protection methods, organizations can increase their chances of detecting and blocking network attacks like port scans, vulnerability scans, spoofing, and packet crafting. Regular monitoring and auditing of network traffic, alongside a layered security approach, are key to maintaining a secure environment.