CYBER SECURITY BASICS

The CIA Triad is a fundamental model for cybersecurity, representing the three core goals of information security: Confidentiality, Integrity, and Availability. These principles guide the development of security policies and controls to protect data and systems.

The CIA Triad

- **Confidentiality:** This principle ensures that information is accessible only to those authorized to have access. Think of it as **secrecy**. Encryption, access control lists, and multi-factor authentication are all tools used to maintain confidentiality. For example, a bank statement should only be viewable by the account holder.
- **Integrity:** This principle guarantees that data remains **accurate and trustworthy** throughout its lifecycle. It ensures that information hasn't been altered or destroyed in an unauthorized manner. Digital signatures, hashing, and checksums are methods used to verify data integrity. For instance, ensuring a file downloaded from the internet is the same as the original and hasn't been tampered with.
- **Availability:** This principle ensures that systems, services, and data are **accessible and usable** by authorized users when they are needed. It focuses on the timely and uninterrupted access to information. Redundancy, backups, and disaster recovery plans are crucial for maintaining availability. A website crashing or a server being down would be a violation of availability.

Threat Types

Threats are potential dangers that could exploit vulnerabilities to compromise an asset.

- **Phishing:** A type of social engineering where attackers impersonate a trustworthy entity (like a bank or a well-known company) to trick individuals into revealing sensitive information, such as passwords or credit card numbers, usually via email.
- **Malware:** Malicious software designed to disrupt computer operations, gather sensitive information, or gain unauthorized access to computer systems. Types include viruses, worms, and trojans.
- **DDoS (Distributed Denial of Service):** An attack where multiple compromised computer systems attack a single target, causing a denial of service for users of the targeted system. This essentially floods a server with traffic to overwhelm it and make it unavailable.
- **SQL Injection:** A web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can enable the attacker to view, modify, or delete sensitive data from the database.
- **Brute Force:** A trial-and-error method used to guess information, such as passwords or encryption keys. The attacker systematically tries every possible combination until the correct one is found.
- **Ransomware:** A type of malware that encrypts a victim's files, and the attacker demands a ransom payment to restore access to the data.

Attack Vectors 🚀

Attack vectors are the paths or methods an attacker uses to gain unauthorized access to a system or network.

- **Social Engineering:** Manipulating people into divulging confidential information or performing actions that compromise security. This can be done through various techniques like phishing, pretexting (creating a fabricated scenario), or baiting.
- **Wireless Attacks:** Exploiting vulnerabilities in wireless networks (Wi-Fi, Bluetooth) to gain access. Examples include cracking Wi-Fi passwords, setting up fake access points (evil twins), or sniffing unencrypted traffic.
- **Insider Threats:** Security risks that originate from within the targeted organization. This could be a current or former employee, contractor, or partner who has access to an organization's systems and intentionally or unintentionally uses that access to cause harm.

LINUX FUNDAMENTALS

File System Navigation

These commands help you move around and explore the Linux file system.

- **cd (change directory):** This is used to change your current location. For example, cd /home/user/documents moves you to the "documents" directory. Typing cd by itself returns you to your home directory.
- **ls (list):** This command lists the files and directories in your current location. You can add options like -l to get a long list with more details, or -a to show all files, including hidden ones.
- **pwd (print working directory):** This command simply tells you where you are right now by printing the full path of your current directory.

File & Directory Permissions

Linux uses permissions to control who can **read**, **write**, or **execute** a file or directory. Permissions are set for three categories: the **user** (owner), the **group** the file belongs to, and **others** (everyone else).

- **chmod (change mode):** This command is used to change a file's permissions. You can use symbolic notation (e.g., chmod u+x file.sh to add execute permission for the user) or numeric (octal) notation. The numeric values are: **r**ead (4), **w**rite (2), and **e**xecute (1). For example, chmod 755 file.sh gives the owner all permissions (4+2+1=7), while the group and others can only read and execute (4+1=5).
- **chown (change owner):** This command changes the ownership of a file or directory. For example, chown newuser file.txt changes the owner of file.txt to "newuser." You can also change the group by using the format chown newuser:newgroup file.txt.

Package Management

Package managers handle the installation, upgrading, configuration, and removal of software. **apt** and **dpkg** are two common tools on Debian-based systems like Ubuntu.

- **apt (Advanced Packaging Tool):** This is a high-level command-line tool. It's user-friendly and handles dependency resolution automatically.
    - **apt update**: Fetches the latest package information from the repositories.
    - **apt install <package_name>**: Installs a new package and its dependencies.
    - **apt upgrade**: Upgrades all installed packages to their newest versions.
    - **apt remove <package_name>**: Removes a package but leaves its configuration files behind.
- **dpkg (Debian Package):** This is a lower-level tool for managing individual .deb package files. Unlike apt, it does not resolve dependencies.
    - **dpkg -i <file.deb>**: Installs a package from a .deb file.
    - **dpkg -r <package_name>**: Removes an installed package.

Networking Commands

These commands are essential for configuring and troubleshooting network connections.

- **ifconfig (interface configuration):** This command displays information about network interfaces, such as IP addresses, subnet masks, and MAC addresses. While still widely used, it has been largely superseded by the **ip** command in many modern Linux distributions.
- **ping:** This command sends a small packet of data to a target host to test connectivity. It's a fundamental tool for checking if a machine is online and reachable.
- **netstat (network statistics):** This command shows active network connections, routing tables, and interface statistics. For example, netstat -a lists all active connections and listening ports.
- **traceroute:** This command traces the route that data packets take to reach a destination. It shows you all the "hops" (routers) along the path, which is useful for diagnosing where a connection might be failing or experiencing delays.

NETWORK BASICS

OSI Model Layers & Functions 🌐

The **OSI (Open Systems Interconnection) model** is a conceptual framework that standardizes the functions of a telecommunication or computing system into seven distinct layers. It helps us visualize and understand how data travels from one device to another across a network.

- **Layer 7: Application Layer** 💻
    - **Function:** This is the layer that interacts directly with the user and their software. It's where protocols like HTTP, FTP, and SMTP live.
- **Layer 6: Presentation Layer** 🖼️

    - **Function:** Responsible for data formatting, encryption, and compression. It ensures that data is in a format that the receiving application can understand.

- **Layer 5: Session Layer** 🗣️
    - **Function:** Manages and controls communication sessions between applications. It establishes, maintains, and terminates connections between devices.
- **Layer 4: Transport Layer** 📦
    - **Function:** Provides reliable (TCP) or unreliable (UDP) data transfer between hosts. It segments data from the upper layers and reassembles it at the destination. This is where **end-to-end communication** is managed.
- **Layer 3: Network Layer** 🗺️
    - **Function:** Responsible for **routing** data packets between different networks. It uses **IP addresses** to find the best path for data to travel.
- **Layer 2: Data Link Layer** ⛓️
    - **Function:** Handles communication between two devices on the **same network** segment. It uses **MAC addresses** and organizes data into frames.
- **Layer 1: Physical Layer** 🔌
    - **Function:** The physical transmission of raw bits over a medium. This includes cables, wireless signals, and network adapters.

TCP/IP Protocol Suite 🚦

The **TCP/IP model** is a more practical, four-layered model that's the foundation of the internet. It combines several of the OSI layers.

1. **Application Layer:** Combines OSI layers 5, 6, and 7. It includes protocols like HTTP, FTP, SMTP, and DNS.
2. **Transport Layer:** Similar to the OSI Transport Layer, it uses **TCP** (Transmission Control Protocol) for reliable, connection-oriented communication, and **UDP** (User Datagram Protocol) for fast, connectionless communication.
3. **Internet Layer:** Similar to the OSI Network Layer, it's responsible for addressing and routing using the **Internet Protocol (IP)**.
4. **Network Access Layer:** Combines OSI layers 1 and 2. It handles the physical transmission of data on the network hardware.

DNS & HTTP/HTTPS Deep Dive 🔍

- **DNS (Domain Name System):** The internet's phonebook. It translates human-readable **domain names** (e.g., www.google.com) into machine-readable **IP addresses** (e.g., 172.217.168.14). When you type a website address, your computer performs a DNS lookup to find the server's IP address.
- **HTTP (Hypertext Transfer Protocol):** The core protocol for the web. It's a client-server protocol used to request and transfer resources, like HTML documents and images, over the internet. HTTP is a **stateless** protocol, meaning each request is independent of previous ones.

- **HTTPS (Hypertext Transfer Protocol Secure):** This is the **secure version of HTTP**. It uses **SSL/TLS encryption** to secure the communication between a web browser and a server. This protects sensitive data, such as passwords and credit card information, from being intercepted. You can identify it by the lock icon in your browser's address bar.

IP Addressing, Subnetting, and NAT 📍

- **IP Addressing:** An **IP address** is a unique numerical label assigned to each device on a computer network that uses the Internet Protocol for communication.
    - **IPv4** is the most common version, using 32-bit addresses (e.g., 192.168.1.1).
    - **IPv6** is the newer standard, using 128-bit addresses to address the exhaustion of IPv4 addresses.
- **Subnetting:** The process of dividing a single large network into smaller, more manageable subnetworks (or subnets). This is done by using a **subnet mask** to separate the network portion of an IP address from the host portion. Subnetting improves network efficiency and security by controlling traffic flow.
- **NAT (Network Address Translation):** A method used by routers to allow multiple devices on a private network to share a single public IP address. When a device on a private network wants to access the internet, the router uses NAT to translate the device's private IP address into the router's public IP address. This conserves public IP addresses and adds a layer of security by hiding the internal network's structure.

CRYPTOGRAPHIC BASICS

Symmetric vs. Asymmetric Encryption 🔑

Encryption is the process of converting data into a scrambled, unreadable form to protect its confidentiality. There are two primary types of encryption:

- **Symmetric Encryption**
    - **How it works:** This method uses a **single, shared key** for both encrypting and decrypting data. It's like a single key that both locks and unlocks a box. Both the sender and receiver must have this same key.
    - **Pros:** It's **fast** and efficient, making it ideal for encrypting large amounts of data.
    - **Cons:** Key distribution is a major challenge. How do you securely share the secret key with the other party without it being intercepted?
    - **Examples:** AES (Advanced Encryption Standard), DES (Data Encryption Standard).
- **Asymmetric Encryption**
    - **How it works:** This method uses a **pair of keys**: a public key and a private key. Data encrypted with the public key can only be decrypted with the corresponding private key, and vice versa. You can freely share your public key with anyone, but your private key must remain a secret.
    - **Pros:** It solves the key distribution problem. You don't need to share a secret key beforehand.
    - **Cons:** It's much **slower** and more computationally intensive than symmetric encryption, making it less suitable for bulk data encryption.

- **Examples:** RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography).

In practice, a hybrid approach is often used: asymmetric encryption is used to securely exchange a symmetric key, which is then used for the main data transfer.

Hashing (MD5, SHA256) 📝

A **hash function** is a mathematical algorithm that takes an input (or 'message') and produces a fixed-size string of bytes. This output is called a **hash value** or **digest**. Hashing is a **one-way process**; you cannot reverse the hash to get the original data.

- **Key Properties of a Hash Function:**
  - **Determinism:** The same input will always produce the same hash.
  - **Uniqueness:** It's computationally infeasible to find two different inputs that produce the same hash (a "collision").
- **MD5 (Message Digest 5):** An older hashing algorithm that produces a 128-bit hash. It's now considered **cryptographically broken** because collisions have been found, meaning it's not secure for verifying data integrity.
- **SHA256 (Secure Hash Algorithm 256):** A part of the SHA-2 family, it produces a 256-bit hash. It is widely used and considered secure for applications like verifying file integrity, digital signatures, and in cryptocurrencies like Bitcoin.

Digital Certificates & SSL/TLS 🔐

- **Digital Certificates:** An electronic document used to prove the **ownership of a public key**. Issued by a **Certificate Authority (CA)**, it links a public key to an entity's identity (like a website, organization, or individual) and verifies that the key belongs to that entity. This provides **authentication** and ensures you're communicating with the right person or website.
- **SSL/TLS: SSL (Secure Sockets Layer)** and its successor, **TLS (Transport Layer Security)**, are protocols that provide **encryption and authentication** for communication over a network. When you visit a website with HTTPS, your browser and the web server perform a "TLS handshake" to:
  1. **Verify the server's identity** using its digital certificate.
  2. **Establish a secure, encrypted connection** using a shared symmetric key. This protects data in transit from being read or tampered with.

Hands-on: Encrypt and Decrypt with OpenSSL 💻

OpenSSL is a powerful command-line tool and library for various cryptographic tasks. Here's a basic example of using it for symmetric encryption with AES.

1. Encrypt a message:

First, create a simple text file, message.txt, with your message.

Bash

```
echo "Hello, this is a secret message." > message.txt
```

Now, encrypt the file using AES-256. The -aes-256-cbc flag specifies the encryption algorithm, -e is for encrypt, -a creates a base64-encoded output, -in is the input file, and -out is the output file. You will be prompted to create a password (the symmetric key).

Bash

openssl aes-256-cbc -e -a -in message.txt -out encrypted.txt

2. Decrypt the message:

To decrypt, use the same command but change the flag from -e to -d (for decrypt) and swap the input and output files. You must enter the same password.

Bash

openssl aes-256-cbc -d -a -in encrypted.txt -out decrypted.txt

3. Verify the decryption:

You can now check the decrypted file to ensure the message is restored.

Bash

cat decrypted.txt

TOOL FAMILIARIZATION

Wireshark (Packet Capture) 🕵️‍♀️

Wireshark is a free and open-source **network protocol analyzer**. It's used to capture and interactively browse the traffic running on a computer network. Think of it as a microscope for your network. It allows you to see the individual data packets flowing in and out of your system, providing deep insights into network communication.

- **Key Functions:**
    - **Packet Sniffing:** Captures live data packets from a network interface.
    - **Protocol Analysis:** Decodes and displays the contents of various protocols (TCP, UDP, IP, HTTP, etc.) in a human-readable format.
    - **Filtering:** Allows you to filter captured traffic based on specific criteria, such as source/destination IP addresses, ports, or protocol types, which is essential for troubleshooting and security analysis.
- **Use Cases:**
    - Troubleshooting network problems.
    - Analyzing application performance.
    - Examining network security issues.

Nmap (Network Scanning) 📡

Nmap (Network Mapper) is a powerful, open-source utility for **network discovery and security auditing**. It's used by network administrators and security professionals to discover hosts and services on a computer network.

- **Key Functions:**
    - **Host Discovery:** Identifies active hosts on a network.
    - **Port Scanning:** Scans for open ports on a host to determine what services are running.
    - **Service & Version Detection:** Can identify the application and version number of services running on open ports.
    - **OS Detection:** Can guess the operating system of a host.
- **Use Cases:**
    - Auditing network security.
    - Managing service upgrade schedules.
    - Monitoring host or service uptime.

Burp Suite (Web Proxy) 🕸️

Burp Suite is a comprehensive platform for performing **web application security testing**. Its most commonly used feature is its **web proxy**, which sits between your browser and a web server. This allows you to intercept, inspect, and modify all traffic passing between them.

- **Key Functions:**
    - **Interception Proxy:** Intercepts HTTP/HTTPS requests and responses.
    - **Scanner:** Automatically finds vulnerabilities in web applications.
    - **Repeater:** Manually resends a request, which is useful for testing how an application handles repeated submissions or modified data.
    - **Intruder:** Automates customized attacks, such as brute force or fuzzing.
- **Use Cases:**
    - Finding vulnerabilities in web applications.
    - Analyzing web traffic and application logic.
    - Testing security controls like authentication and session management.

Netcat (Network Debugging) 🛠️

Netcat (often abbreviated as nc) is a versatile networking utility known as a "network swiss army knife." It's used for reading from and writing to network connections using TCP or UDP.

- **Key Functions:**
    - **Port Scanning:** Can be used for basic port scanning (e.g., nc -zv <host> <port-range>).

    - **File Transfer:** Can transfer files over a network connection (e.g., cat file | nc -l -p 1234 on the server and nc <server_ip> 1234 > file on the client).
    - **Chat Server:** Can be used to create a simple chat or communication channel between two machines.
    - **Basic Web Server:** Can be used to serve a single file or a simple response.
- **Use Cases:**
    - Debugging network scripts and programs.
    - Opening a listening port to test a service.
    - Performing quick data transfers between systems.