Flash Cards

202 / 202

- Below are my flash cards which i use to revise from

- Week 2 Network Security Basics
  - ## CIA Triad & Security Basics
  - **CIA Triad** ↔ Confidentiality, Integrity, Availability1.
  - **Confidentiality** ↔ Preserving authorised restrictions on information access and disclosure (protecting privacy)2.
  - **Integrity** ↔ Guarding against improper information modification or destruction (ensuring non-repudiation)3.
  - **Availability** ↔ Ensuring timely and reliable access to and use of information4.
  - **Symmetric Key Cryptography** ↔ Encryption system where the same secret key is used for both encryption and decryption5.
  - **Public-Key (Asymmetric) Cryptography** ↔ Encryption system using a pair of keys: a public key (for encryption/verifying) and a private key (for decryption/signing)6.
  - ## Network Stack & Protocols
  - **4 Layers of TCP/IP Model** ↔ Application, Transport, Internet, Network (Link)7.
  - **Application Layer** ↔ Encodes/decodes messages (e.g., HTTPS, DNS)8.
  - **Transport Layer** ↔ Breaks messages into packets (TCP/UDP) and handles end-to-end delivery9.
  - **Network Layer** ↔ Handles routing and logical addressing (IP addresses)10.
  - **Link/MAC Layer** ↔ Transfers packets between nodes on the same

network using physical addresses11.

- **DHCP (Dynamic Host Configuration Protocol)** ↔ Protocol that automatically assigns IP addresses and network settings to devices12.
- **DHCP Sequence** ↔ DORA: Discover, Offer, Request, Acknowledge 13.
- **ARP (Address Resolution Protocol)** ↔ Maps an IP address to a MAC address so frames can be sent to the local router/gateway14.
- **DNS (Domain Name System)** ↔ Resolves domain names (like https://www.google.com/search?q=google.com) to IP addresses15.
- **TCP 3-Way Handshake** ↔ SYN → SYN-ACK → ACK16161616.

PGP (Pretty Good Privacy)

- **PGP** ↔ A hybrid cryptosystem for email security that combines symmetric and public-key cryptography17.
- **PGP Confidentiality Mechanism** ↔ Encrypts the message with a random session key, then encrypts the session key with the recipient's public key18.
- **PGP Authentication Mechanism** ↔ Creates a hash of the message and signs (encrypts) it with the sender's private key19191919.
- **Why PGP uses compression** ↔ To save space and reduce patterns in plaintext, which strengthens resistance against cryptanalysis20202020.
- **Web of Trust** ↔ A decentralized trust model where users sign each other's public keys to validate identity (instead of a central CA)21.
- **Radix-64 (Base64) Encoding** ↔ Used by PGP to convert binary encrypted data into ASCII strings for email compatibility22.

IPsec (Network Layer Security)

- **IPsec** ↔ A suite of protocols providing encryption and authentication at the IP (Network) layer, covering all applications23.
- **AH (Authentication Header)** ↔ IPsec protocol providing authentication and integrity, but no confidentiality (no encryption)24.
- **ESP (Encapsulating Security Payload)** ↔ IPsec protocol providing confidentiality (encryption) and authentication25.
- **IPsec Transport Mode** ↔ Encrypts only the payload (data); used for end-to-end communication between hosts26262626.
- **IPsec Tunnel Mode** ↔ Encrypts the entire IP packet (including header) and wraps it in a new header; used for VPNs27272727.
- **Security Association (SA)** ↔ A one-way logical connection defining

security parameters (keys, algorithms) for a specific flow28.

- **IKE (Internet Key Exchange)** ↔ Protocol used to establish Security Associations and exchange keys for IPsec29.

SSL/TLS (Transport Layer Security)

- **SSL/TLS** ↔ Protocol providing security at the Transport layer (above TCP), widely used for HTTPS30.
- **SSL Record Protocol** ↔ Lower layer protocol responsible for fragmenting, compressing, and encrypting data31.
- **SSL Handshake Protocol** ↔ Upper layer protocol responsible for authenticating peers and negotiating keys/algorithms32.
- **Heartbeat Extension** ↔ Keeps a TLS session alive during idle periods to avoid the overhead of renegotiation33.
- **Heartbleed Bug** ↔ A vulnerability in the Heartbeat extension where missing length checks allowed attackers to read sensitive server memory34.

Week 3 Authentication and Access Control

Authentication Fundamentals

- **Authentication** ↔ The process of confirming that a user, service, or application is who they claim to be.
- 3 Core Operations of Authentication → Registration, Authentication, and Recovery .
- Security-Convenience Paradox → The concept that the most secure authentication approaches are often the least convenient, and convenient ones are usually the least secure. Why is Recovery a risk? → If recovery (e.g., "forgot password") is too simple, it bypasses authentication; if too strict, it locks out legitimate users.

Threats to Passwords

- **Brute Force Attack** ↔ Attempting to guess a password by systematically trying every possible combination of characters.
- Dictionary Attack / Educated Guesswork → Trying common words, personal info (like birthdays), or default passwords rather than random combinations.
- Phishing → Tricking a user into revealing their credentials via deceptive emails or fake websites.
- Credential Stuffing (Password Reuse) → Using credentials stolen from one breach to attempt logins on other services where the user reused the

same password.

- Keylogger → Software or hardware that records user keystrokes to steal passwords as they are typed.
- Offline Cracking → Stealing the password file (hashes) and attempting to crack them on the attacker's own hardware (faster and stealthier than online attempts).
- Hardware Keylogger → A physical device inserted between the keyboard and computer to intercept keystrokes; hard to detect via software.

Secure Password Storage

- **Why hash passwords instead of encrypting?** ↔ Hashing is **one-way** (irreversible), while encryption is two-way (can be decrypted if the key is stolen) .
- Salt → A unique, random value added to **each** password before it is hashed.
- Purpose of Salting → Prevents the use of Rainbow Tables (pre-computed hashes) and ensures identical passwords result in different hashes.
- Pepper → A secret key added to the password (pre-hash) or hash (post-hash) that is stored **separately** from the database (e.g., in a Hardware Security Module).
- Work Factor (Iterations) → Increasing the computational cost (time/memory) of hashing to make brute-force attacks slower and more expensive .

Multi-Factor Authentication (2FA)

- **2FA (Two-Factor Authentication)** ↔ A method confirming identity using two distinct forms of evidence from different categories.
- 3 Factors of Authentication → Something you **Know** (password), Something you **Have** (token/phone), Something you **Are** (biometric).
- TOTP (Time-based One-Time Password) → Algorithm that generates a temporary 6-digit code based on the current time and a shared secret key (e.g., Google Authenticator). SMS 2FA
- Vulnerability → Relies on the insecure SS7 network and is vulnerable to SIM Swapping and interception.
- SIM Swapping → An attack where a criminal convinces a mobile carrier to transfer a victim's phone number to a new SIM card to intercept SMS codes.
- FIDO2 / WebAuthn → Modern standards allowing passwordless authentication using public-key cryptography and hardware tokens.

- # Week 4 Malicious Software

Malware Definitions & Types

- **Virus** ↔ Parasitic software that attaches to a host program and requires user execution to
- replicate. Worm → A standalone program that actively replicates and propagates across a network without needing a host file.
- Trojan Horse → Malicious software disguised as legitimate or useful software to trick the user into installing it.
- Logic Bomb → Dormant malicious code triggered by a specific event or condition (e.g., a specific date or file deletion).
- Rootkit → A set of tools used to maintain privileged (admin/root) access to a system while hiding its presence.
- Backdoor (Trapdoor) → A secret entry point into a program or system that allows bypassing normal security procedures.
- Exploit → Code specific to a single vulnerability or bug in software/hardware.
- Zero-day Exploit → An attack that exploits a vulnerability that is unknown to the software developer (no patch exists yet).

Virus Mechanics

- **3 Components of a Virus** ↔ **Infection Mechanism** (Vector), **Trigger** (Logic Bomb), **Payload** (Action) .
- 4 Phases of a Virus → **Dormant**, **Propagation** (Replication), **Triggering**, **Execution** . Boot Sector Virus → Infects the master boot record and spreads when the system is booted from the infected disk.
- Macro Virus → Platform-independent virus that infects documents (like Word/Excel) using scripting languages.
- Polymorphic Virus → Mutates its "signature" (appearance) with every infection to evade signature-based detection.
- Metamorphic Virus → Completely rewrites its own code/behavior at each iteration to be even harder to detect than polymorphic viruses.

Worms & Mobile Code

- **Morris Worm** ↔ The first major internet worm (1988) that exploited UNIX vulnerabilities (finger, debug, rsh).
- Drive-by-download → Malware that downloads and installs itself simply by visiting a malicious webpage, often exploiting browser vulnerabilities.

- Clickjacking → A UI redress attack using transparent layers to trick users into clicking hidden buttons.
- Social Engineering → Attacks that manipulate people into performing actions or divulging confidential information (e.g., Phishing).

Payloads & Botnets

- **Botnet** ↔ A network of compromised computers ("zombies" or "bots") controlled by a central attacker.
- C&C (Command and Control) → The infrastructure used by an attacker to send instructions to a botnet.
- Ransomware → Malware that encrypts a user's data and demands payment for the decryption key.
- Keylogger → Software that captures keystrokes to steal passwords and sensitive info.
- Spyware → Software that monitors user activity and transmits data to a third party without consent.
- Spear-phishing → A targeted phishing email crafted specifically for a chosen individual or organization.

Countermeasures & Detection

- **4 Generations of Antivirus** ↔ 1. Simple Scanners (Signatures), 2. Heuristic Scanners (Static rules), 3. Activity Traps (Memory resident/Behavior), 4. Full-feature Protection .
- Heuristic Scanner → Antivirus that looks for code patterns or probable malware characteristics rather than exact signatures.
- Ingress Monitor → Monitoring traffic **entering** the network (at the border/firewall).
- Egress Monitor → Monitoring traffic **leaving** the network (useful for detecting botnets calling home).
- Honeypot → A decoy system designed to lure attackers to detect, deflect, or study attempts to gain unauthorized access.

Week 5

Web Security Fundamentals

- **Web Security Definition** ↔ The practice of protecting websites and applications to ensure **Confidentiality, Integrity, and Availability** (CIA) .
- HTTP Protocol Nature → **Stateless** (the server does not remember

previous interactions by default).

- How is state maintained in Web Apps? → Using **Cookies** to store a session identifier ("ticket") . Same-Origin Policy (SOP) → A browser security rule that prevents scripts from one origin (domain/protocol/port) from accessing resources on another origin.
- OWASP → **Open Web Application Security Project**; a non-profit providing tools and documenting critical web risks (e.g., OWASP Top 10) .
- Zero Trust Input → The principle that **all** user data (forms, cookies, headers) must be treated as untrusted and potentially malicious.

Server-Side Vulnerabilities

- **SQL Injection (SQLi)** ↔ A vulnerability where an attacker sends malicious SQL code in an input field to manipulate the database query .
- Primary Defense against SQL Injection → **Parameterized Queries** (Prepared Statements) which compile the code before inserting user data .
- Broken Access Control → Failure to verify **authorization** (permissions) after authentication (login) .
- IDOR (Insecure Direct Object Reference) → An attack where a user accesses unauthorized data by simply changing an identifier (e.g., `user_id=501`) in the URL .
- Primary Defense against IDOR → **Server-side permission checks** on every single request.

Client-Side Vulnerabilities

- **XSS (Cross-Site Scripting)** ↔ Injecting malicious client-side scripts (usually JavaScript) into a trusted website to be executed by a victim's browser .
- Stored XSS → Malicious script is **saved on the server** (e.g., in a comment) and attacks anyone who views the page .
- Reflected XSS → Malicious script is part of a **link/URL** and is "reflected" back to the user immediately (requires tricking the user to click) .
- Primary Defense against XSS → **Context-Aware Output Encoding** (converting special characters like `<` to `<` before displaying them) .
- CSRF (Cross-Site Request Forgery) → Tricking an authenticated user's browser into sending an unintended request to a trusted site (e.g., "Change Password") .
- Primary Defense against CSRF → **Anti-CSRF Tokens** (unique, secret tokens embedded in forms that the attacker cannot guess) .

- ## Week 6
  - ## Malware Analysis Fundamentals
    - **What is the core goal of Malware Analysis?** ↔ To investigate suspicious files/behaviors and assess their **capability, intent, and impact**.
    - 3 Main Analysis Techniques → **Static** Analysis, **Dynamic** Analysis, and **Code** Analysis (Reverse Engineering).
    - IOCs (Indicators of Compromise) → Artifacts used to identify malware, such as file **hashes**, C2 **domains/IPs**, mutexes, and registry paths.
    - YARA vs. Sigma → **YARA** rules detect malware in files/memory; **Sigma** rules detect malware behavior in SIEM logs.
  - ## The Analysis Workflow
    - **Static Triage** ↔ Analyzing a file *without* running it (e.g., checking headers, imports, strings, and packer identification).
    - Dynamic Analysis → Running malware in an **isolated environment** (VM) to monitor filesystem, registry, and network changes (using tools like ProcMon/Wireshark).
    - Purpose of Unpacking/Patching → To defeat code obfuscation (packing) or bypass **anti-debug/anti-VM** checks to analyze the core payload.
    - Code Reverse Engineering (RE) → Analyzing the assembly/source code to recover control flow, find encryption keys, and understand triggers.
  - ## Tools & Competencies
    - **Lab Hygiene** ↔ Using **VM isolation** and taking **snapshots** to safely run and analyze malware.
    - Common RE Tools (Disassemblers) → **IDA Pro** and **Ghidra** (used for x86/x64 and .
    - NET analysis). Common Dynamic Tools (Debuggers/Monitors) → **x64dbg**, **WinDbg**, **ProcMon**, **Sysmon** .
    - Memory Forensics Tools → **Volatility** or **Rekall** (used to dump processes and extract keys/configs from RAM).
    - Network Analysis Tool → **Wireshark** (used to analyze PCAP files for C2 beaconing and protocol fingerprinting).
- ## Week 7

Penetration Testing Fundamentals

- **Penetration Testing Definition (NCSC)** ↔ A method for gaining assurance in the security of an IT system by attempting to breach its security using the same tools and techniques as an adversary.
- Core Purpose of Pen Testing → To validate and provide assurance in your existing vulnerability assessment and management processes (Quality Assurance), NOT to be the primary method for finding bugs.
- Financial Audit Analogy → Pen testing is like an **external audit** verifying that internal security processes (daily finance tracking) are working correctly .
- What Pen Testing is NOT → It is not a substitute for regular security controls, functional testing, or continuous monitoring .
- CHECK Scheme → A UK government-approved scheme ensuring testers have verified qualifications and use standardized methodologies (recommended for HMG organizations) .

Types of Testing (Box Colors)

- **Black Box (Opaque) Testing** ↔ Testers have **no prior knowledge** or access; simulates an external real-world attacker .
- White Box (Transparent) Testing → Testers have **full system knowledge** (code, architecture, creds); allows for thorough, comprehensive coverage .
- Grey Box Testing → Testers have **partial knowledge** or limited access (e.g., user credentials); balances realism with efficiency .

The Penetration Testing Methodology

- **6 Stages of Pen Testing** ↔ 1. Reconnaissance, 2. Scanning & Enumeration, 3. Vulnerability Assessment, 4. Exploitation, 5. Post-Exploitation, 6. Reporting .
- Reconnaissance (Stage 1) → Gathering information about the target (Passive vs Active).
- Exploitation (Stage 4) → Attempting to compromise identified vulnerabilities to gain access.
- Post-Exploitation (Stage 5) → Maintaining access, escalating privileges, and expanding control after the initial breach.

Key Tools by Category

- **Nmap** ↔ The industry-standard tool for **network scanning**, host discovery, and service enumeration.

- Metasploit → A comprehensive framework for developing and executing **exploit code** against a remote target.

- Burp Suite → The leading tool for **web application security testing**, acting as an intercepting proxy. Wireshark → Industry-standard tool for **network traffic analysis** and packet capturing.
- John the Ripper / Hashcat → Tools used for **password cracking** (offline hash attacks) .
- Nessus / OpenVAS → Automated **vulnerability scanners** that check for known security flaws .
- Aircrack-ng → A suite of tools for assessing **WiFi network security** (monitoring, attacking, cracking).
- Shodan → A search engine for finding internet-connected devices (passive reconnaissance).

Week 8

Internet of Things (IoT) Fundamentals

- **IoT Definition** ↔ A network of physical devices embedded with sensors, software, and connectivity to collect and exchange data.
- Difference between Sensors and Actuators → **Sensors** gather data (detect), while **Actuators** provide real-world interactions (do something, like turn on a switch). 4 Layers of IoT
- Architecture → 1. **Perception** (Device), 2. **Network** (Connectivity), 3. **Processing** (Middleware), 4. **Application** .
- What is **Edge Computing**? → Processing data as close to the source (device) as possible to reduce latency and bandwidth use, rather than sending everything to the cloud.

IoT Connectivity & Spectrum

- **Wired vs. Wireless Security** ↔ **Wired** signals are constrained to the physical cable (harder to intercept without physical access), while **Wireless** signals are broadcast and easier to intercept .
  Licensed vs. Unlicensed Spectrum ↔
  - **Licensed**: Exclusive use (less interference), predictable performance, but expensive .
  - **Unlicensed: Free to use (e.g., Wi-Fi), faster deployment, but subject to high interference . Range/Power Trade-off protocols** ↔ Back of card

- **Wi-Fi**: High bandwidth, high power, medium range.
- **Bluetooth/Zigbee**: Low bandwidth, low power, short range.
  - **LoRaWAN**: Low bandwidth, low power, **long range**.

• IoT Security Challenges & Best Practices

- **Device-Level Vulnerabilities** ↔ Constrained resources (CPU/Battery) make encryption hard; **insecure default configurations** (e.g., default passwords); lack of update mechanisms . Network-Level Vulnerabilities ↔ Unencrypted communication (Man-in-the-Middle); lack of **network segmentation** allowing lateral movement to critical systems . **Mirai Botnet** Lesson ↔ Exploited **default credentials** on IoT devices to create a massive botnet for DDoS attacks. IoT Security Best Practices ↔
  - **Network Segmentation**: Isolate IoT devices on separate VLANs.
  - **Secure Boot/TPM**: Use hardware roots of trust.
  - **Patch Management**: Automate firmware updates.

• Cloud Security Fundamentals

- **Cloud Computing Service Models** ↔ Back of card
  - **IaaS (Infrastructure as a Service)**: Vendor manages hardware; you manage OS, apps, data (e.g., AWS EC2) .
  - **PaaS (Platform as a Service)**: Vendor manages OS/Runtime; you manage apps and data .
  - **SaaS (Software as a Service)**: **Vendor manages everything; you just use the software (e.g., Gmail) . Shared Responsibility Model** ↔ The Cloud Provider is responsible for security **of** the cloud (hardware, facilities), while the Customer is responsible for security **in** the cloud (data, access, configs) . Top Cloud Security Threat ↔ **Misconfiguration** (e.g., public S3 buckets) is responsible for over 60% of incidents.

• Week 9

• MITRE ATT&CK Framework

- **What is MITRE ATT&CK?** ↔ A global knowledge base of adversary tactics and techniques based on real-world observations.
- TTPs stands for → **Tactics**, **Techniques**, and **Procedures**.
- Tactics (The "Why") → The attacker's tactical goal or objective (e.g., Initial

Access, Persistence).

- Techniques (The "How") → The specific method used to achieve a tactic (e.g., Phishing, T1566).
- Procedures → The specific implementation or steps used to execute a technique. Attack
- Lifecycle (Kill Chain) → A model describing the phases of an attack, from Reconnaissance to Impact