# Task-1

## Understanding Cyber Security Basics & Attack Surface

### 1.CIA Triad:-

The CIA triad is a foundational model in cybersecurity consisting of three key principles: Confidentiality, Integrity, and Availability. These principles guide organizations in developing security policies to protect data and systems effectively.

### Confidentiality:

Confidentiality ensures that sensitive information remains accessible only to authorized users, preventing unauthorized disclosure through measures like encryption, access controls, and multi-factor authentication.

### Integrity :

Integrity protects data from unauthorized modification, alteration, or destruction, maintaining its accuracy and trustworthiness via techniques such as checksums, digital signatures, and version control.

### Availability :

Availability guarantees timely and reliable access to information and resources for authorized users, defended against disruptions like DDoS attacks or hardware failures using redundancies and backups.

The CIA triad applies directly to real-world sectors like banking and social media, where breaches often target one or more principles, leading to significant consequences. Examples illustrate how failures in confidentiality, integrity, or availability disrupt operations and trust.

## Banking Example:

Online banking systems use encryption and multi-factor authentication (MFA) for confidentiality to protect account details from unauthorized access. The 2018 Barclays IT glitch violated availability by preventing customer access to accounts during payday, while also compromising integrity through duplicated or inaccurate transactions. ATM machines enforce confidentiality via debit cards and PINs, integrity through transaction logging, and availability with redundant systems.

## Social Media Example:

Fake accounts on platforms like Facebook breach confidentiality by stealing personal data, integrity by spreading misinformation, and availability by disrupting legitimate content access. In the 2018 Facebook breach, attackers exploited user data, primarily violating confidentiality as details were exposed without authorization.

## 2. Identify different types of attackers :

Different types of cyber attackers vary in skill, motivation, and impact, with script kiddies, insiders, hacktivists, and nation-state actors representing key categories identified in security analyses.

### Script Kiddies

Script kiddies are unskilled individuals who use pre-written scripts or tools from others to launch attacks, lacking the expertise to create their own exploits. They often target easy vulnerabilities for fun, bragging rights, or minor disruption, relying on dark web resources or free tools.

### Insiders

Insider threats come from employees, contractors, or partners with legitimate access who misuse it intentionally for gain, revenge, or negligence, leading to data leaks or sabotage. These actors pose unique risks due to their trusted position, bypassing external defenses.

### Hacktivists

Hacktivists launch attacks to promote political, social, or ideological causes, such as defacing websites or leaking data to draw attention to issues. Groups like Anonymous exemplify this, using disruptive tactics like DDoS to protest rather than steal.

### Nation-State Actors

Nation-state actors, often government-backed, conduct sophisticated espionage, sabotage, or disruption with advanced resources for geopolitical goals. Examples include APT29 (Cozy Be linked to Russia or China's Unit 61398, employing long-term persistent threats.

## 3. Explore common attack surfaces:

Attack surfaces represent all potential entry points where cyber attackers can exploit vulnerabilities to gain unauthorized access, disrupt operations, or steal data. Common surfaces like web applications, mobile apps, APIs, networks, and cloud infrastructure each present unique risks that require targeted defenses.

### Web Applications

Web applications serve as prime targets due to frequent exposure to the public internet, vulnerable to attacks like SQL injection, cross-site scripting (XSS), or insecure authentication. Misconfigurations or unpatched flaws allow attackers to manipulate data or hijack sessions, as seen in breaches exploiting outdated frameworks.

### Mobile Apps

Mobile apps expose sensitive user data through insecure storage, weak encryption, or flawed permission models, enabling reverse engineering or interception of network traffic. Common risks include unencrypted local data and API calls that leak personal information when apps run on compromised devices.

### APIs

APIs connecting services often suffer from broken authentication, excessive data exposure, or lack of rate limiting, permitting attackers to scrape data or execute unauthorized commands. Publicly exposed endpoints without proper authorization checks amplify risks in microservices architectures.

### Networks

Networks face threats via open ports, unpatched routers, or misconfigured firewalls, allowing scanning, man-in-the-middle attacks, or lateral movement post-breach. Internal segments and remote access points like VPNs become vulnerable without segmentation or monitoring.

### Cloud Infrastructure

Cloud setups invite issues from misconfigured storage buckets, over-permissive IAM policies, or exposed metadata services, leading to data leaks or resource hijacking. Shadow IT and hybrid environments expand this surface, complicating visibility and control.

## 4. The OWASP Top 10 lists the most critical web application:

The OWASP Top 10 lists the most critical web application security risks, updated periodically to reflect evolving threats, with the 2021 edition (still widely referenced into 2026) emphasizing issues like access control and injections. Each vulnerability is dangerous because it exploits common development flaws, enabling attackers to steal data, disrupt services, or gain full system control, often affecting millions as seen in major breaches.

### Broken Access Control

Attackers bypass authorization to access restricted resources, such as user accounts or admin functions, by manipulating URLs or parameters. This tops the list due to its prevalence in 94% of apps, leading to unauthorized data exposure or privilege escalation.

### Cryptographic Failures

Weak encryption or improper key management exposes sensitive data in transit or at rest, like passwords or PII, making it trivial for interception via man-in-the-middle attacks. Previously "Sensitive Data Exposure," it risks massive compliance violations and identity theft.

### Injection Flaws

Unsanitized inputs allow SQL, command, or XSS injections, letting attackers execute arbitrary code, extract databases, or hijack sessions.Harms include data dumps (e.g., millions of records stolen) and remote code execution on servers.

## Insecure Design

Flaws in application architecture, like missing threat modeling, enable logic bypasses or business rule abuses from the outset. This systemic issue amplifies others, causing persistent exploits despite patches.

## Security Misconfiguration

Default setups, incomplete hardening, or misconfigured permissions (e.g., open S3 buckets) create easy entry points for automated scanners. Affects nearly 90% of apps, often leading to full environment compromise.

## Vulnerable Components

Unpatched libraries or outdated software harbor known exploits, like Log4Shell, allowing zero-effort attacks with public exploits. Supply chain risks propagate to downstream users undetected.

## Identification Failures

Weak auth mechanisms permit credential stuffing, brute force, or session hijacking, granting persistent unauthorized access. Impacts user trust and enables further lateral movement.

## Integrity Failures

Unverified updates or CI/CD pipelines introduce malicious code, as in SolarWinds, undermining software trustworthiness and enabling backdoors.

## Logging Failures

Inadequate monitoring hides breaches, delaying detection from days to months, complicating forensics and response.

## SSRF

Apps fetch attacker-controlled URLs, accessing internal systems or cloud metadata, bypassing firewalls for pivoting or data exfiltration.

## 5. Map daily-used applications:

Daily-used applications like email, WhatsApp, and banking apps expose multiple attack surfaces through their web, mobile, API, network, and cloud components, making them prime targets for exploitation. Mapping these reveals how routine interactions can lead to breaches via phishing, injections, or misconfigurations.

### Email Applications

Email apps (e.g., Gmail, Outlook) primarily expose network and web application surfaces vulnerable to phishing, spoofing, and malware attachments. APIs for IMAP/SMTP connections risk injection flaws or weak authentication, while cloud backends suffer from misconfigured storage leaking inboxes.

### WhatsApp (Messaging App)

WhatsApp leverages mobile app surfaces with insecure local storage or backup encryption, alongside APIs for end-to-end chats prone to man-in-the-middle if keys are compromised. Network traffic during calls or file shares forms another vector, with cloud-synced data (via Google Drive/iCloud) risking exposure from over-permissive policies.

### Banking Apps

Banking apps combine mobile interfaces susceptible to reverse engineering or overlay attacks, APIs for transactions vulnerable to broken access control, and cloud infrastructure with IAM misconfigurations. Networks via public Wi-Fi enable interception, while web portals face OWASP risks like XSS or insecure direct object references.

## 6. Document how data flows:

Data flows through a typical web application follow a structured path from user input to persistent storage, involving client-side rendering, server processing, and database operations. This architecture highlights key attack surfaces at each stage, such as injections or session hijacking, tying back to OWASP risks and daily apps like email or banking.

### User to Application

Users initiate the flow via browsers or mobile apps (e.g., entering login credentials in Gmail or a banking app), where client-side JavaScript validates inputs before packaging them into HTTP requests like POST or GET. Data travels over networks, exposing surfaces to interception if unencrypted, as in WhatsApp calls without E2EE verification.

### Application to Server

The client application sends requests to the web server (e.g., Apache/Nginx), which handles routing, authentication, and business logic in backend code (Node.js, Python). Servers parse data, apply access controls, and generate responses, vulnerable to OWASP flaws like broken authentication during API calls in email syncs.

### Server to Database

Servers query databases (e.g., MySQL, MongoDB) using connectors like SQL drivers, executing reads/writes after sanitization to prevent injections. Results flow back through the server to the user, with cloud-hosted DBs (as in banking backends) risking misconfigurations that leak data in transit.

### Return Flow

Responses retrace the path: database → server (serialization) → application (rendering) → user, completing the cycle. Monitoring this bidirectional flow mitigates risks like SSRF when servers fetch external resources.

## 7. Identify where attacks can happen during this flow:

Attacks can occur at every stage of the data flow from user to application, server, and database, exploiting vulnerabilities tied to OWASP Top 10 risks, attack surfaces, and common apps like email or banking. These entry points align with the previously mapped flows, where poor validation, misconfigurations, or weak controls enable exploitation by attackers like script kiddies or nation-states.

### User to Application

Client-side attacks target input fields or browsers via phishing (e.g., fake Gmail login), XSS injecting malicious scripts into WhatsApp web, or overlay malware on banking apps that steals keystrokes before data even leaves the device. Network interception on public Wi-Fi compromises unencrypted traffic, violating CIA confidentiality.

### Application to Server

HTTP requests face injection flaws (SQLi, command injection) if unsanitized, API abuse via broken access control, or session hijacking through insecure cookies during email syncs. DDoS overwhelms app servers, hitting availability as in Barclays glitches.

### Server to Database

Backend queries risk SSRF if servers fetch attacker URLs, cryptographic failures exposing data in transit, or vulnerable components like unpatched DB connectors allowing RCE. Misconfigured cloud IAM in banking backends leaks entire datasets.

### Return Flow

Responses carry risks like insecure deserialization injecting code or logging failures hiding ongoing breaches, enabling lateral movement post-initial access in email attachments or WhatsApp media previews.

## 8. Summary :

Cybersecurity boils down to protecting the CIA triad—keeping data confidential (only for authorized eyes), integral (untampered), and available (always accessible)—while understanding how attackers exploit everyday app flows.

Key attackers include script kiddies (amateurs running ready-made tools for kicks), insiders (trusted folks gone rogue), hacktivists (protest-driven disruptions like Anonymous DDoS), and nation-states (sophisticated spies like APT groups chasing geopolitics).

Attack surfaces span web apps (SQLi/XSS), mobile (insecure storage), APIs (auth flaws), networks (open ports), and cloud (misconfigs), as in Gmail phishing, WhatsApp backups, or banking overlays.

OWASP Top 10 flags killers like broken access control (sneaking into admin), injections (code execution), and insecure design (flawed from the start), turning routine inputs into breaches.

Data flows user (input via browser/app) → app (HTTP request) → server (logic/auth) → DB (query/store), with attacks everywhere: phishing at input, API abuse mid-flight, SSRF at DB handoff.

Real-world: Email hit by spoofed links (confidentiality loss), banking glitched for unavailability, social media fakes eroding integrity—defend with validation, encryption, monitoring.