Cyber Security Internship – Task 1

Understanding Cyber Security Basics & Attack Surface

Name: Charvi

Internship Domain: Cyber Security

Task Number: 1

Date: 15/ 01 / 2026

## 1. Introduction to Cyber Security

Cyber security refers to the protection of computer systems, networks, applications, and digital data from unauthorized access, cyber attacks, and misuse. In today's digital world, people use the internet for banking, social media, shopping, education, and work. Because so much personal and financial information is shared online, cyber security has become very important. It ensures that data remains safe from hackers, malware, phishing attacks, and data theft. Cyber security uses technologies, rules, and best practices to protect information and maintain user trust.

**Example**

Imagine you use a **banking app** on your mobile phone. You enter your **username, password, and OTP** to check your balance or transfer money. If cybersecurity is weak, a hacker can send you a **fake bank message (phishing)** asking you to verify your account. If you click the link and enter details, the hacker steals your credentials and transfers money from your account.
Cybersecurity prevents this by using **secure login systems, encryption, OTP verification, firewalls, and fraud detection systems**.

## 2. CIA Triad

The foundation of cyber security is the CIA Triad, which consists of Confidentiality, Integrity, and Availability.

These three core principles collectively ensure the protection and reliability of information systems.

Confidentiality restricts information access to authorized users only, preventing data breaches and unauthorized disclosure.

Integrity ensures that data remains accurate, complete, and unaltered except by authorized actions.

Availability guarantees that systems and information are accessible to authorized users when required.

Together, the CIA Triad provides a comprehensive framework for designing and implementing effective security controls**.**

## ❖ Confidentiality

Confidentiality means that information should be accessible only to authorized users and should not be disclosed to unauthorized people. It focuses on keeping data private and preventing data leaks. Confidentiality is maintained using passwords, PINs, encryption, OTPs, and access control systems. If confidentiality is compromised, sensitive information can be stolen and misused, leading to financial loss and identity theft.

Real-World Example (Banking & Social Media):
In online banking, user credentials such as account number, password, and OTP must remain confidential so that only the account holder can access the account. Similarly, on social media platforms like WhatsApp or Instagram, private messages and personal photos are protected by login passwords to ensure that only the account owner can view them.

### ❖ Integrity

Integrity ensures that data remains accurate, complete, and unchanged unless modified by an authorized user. It protects data from unauthorized modification, deletion, or manipulation. Integrity is very important because incorrect or altered data can lead to wrong decisions. Techniques such as hashing, checksums, digital signatures, and access controls are used to maintain data integrity.

**Real-World Example (Banking & Social Media):**
In banking systems, transaction details such as the amount transferred and account numbers must remain correct. If a hacker changes the transaction amount, it becomes an integrity issue. On social media, integrity ensures that posts, comments, and profile information are not altered without the user’s permission.

---

### ❖ Availability

Availability ensures that systems, applications, and data are accessible to authorized users whenever they are needed. It focuses on minimizing downtime and ensuring smooth operation of services. Availability is maintained using backups, load balancing, redundancy, and disaster recovery plans. Cyber attacks like Denial of Service (DoS) can affect availability by making systems unavailable.

**Real-World Example (Banking & Social Media):**
Online banking services should be available 24×7 so that users can check balances or make transactions at any time. Similarly, social media platforms must remain accessible so users can send messages or post content. If servers go down, users cannot access these services, affecting availability.

---

## 3. Types of Cyber Attackers

Cyber attackers are individuals or groups who try to gain unauthorized access to computer systems, networks, or data. Based on their skills, motives, and resources, attackers can be classified into different types. Understanding these attacker types helps organizations design better security measures.

❖ **Script Kiddies**

**Define -**

Script kiddies are inexperienced attackers who lack deep technical knowledge of hacking. Instead of developing their own attack techniques, they use pre-built tools, scripts, or software freely available on the internet. Their main motivation is often curiosity, entertainment, or gaining attention rather than financial or political gain. Even though their skills are limited, script kiddies can still cause serious damage by exploiting known vulnerabilities in poorly secured systems. They usually target small websites, personal blogs, or organizations with weak security.

**Example -**

A college student downloads a free Distributed Denial of Service (DDoS) tool from the internet after watching online tutorials. He launches an attack on a small college website just to test the tool. As a result, the website becomes unavailable for several hours, disrupting online notices and student services, even though the attacker had no serious malicious intent.

---

❖ **Insider Attackers**

**Define -**

Insider attackers are individuals who already have authorized access to an organization's systems, such as employees, contractors, interns, or business partners. Because they are trusted users, insiders understand internal systems, data locations, and security controls. Insider attacks may be intentional, such as data theft or sabotage, or unintentional, such as accidental data leakage. These attacks are

particularly dangerous because they are difficult to detect and can bypass many security mechanisms.

**Example -**

An employee working in a customer support department has access to a company's customer database. Before leaving the company, the employee copies sensitive customer information and sells it to a competitor. Since the employee used legitimate access, the breach is discovered only after customers report misuse of their data.

---

❖ **Hacktivists**

**Define -**

Hacktivists are attackers who use hacking techniques to promote political, social, or ideological causes. Their goal is not financial profit but raising awareness, protesting policies, or damaging the reputation of organizations or governments. Hacktivists often target public-facing systems such as websites and social media accounts. Common hacktivist activities include website defacement, data leaks, and denial-of-service attacks to attract public attention.

**Example -**

A hacktivist group opposes a government policy and targets a government website. They deface the homepage with protest messages and temporarily shut down the site. This draws public and media attention to their cause and disrupts normal government online services.

---

❖ **Nation-State Attackers**

**Define -**

Nation-state attackers are highly skilled and well-funded cyber attackers supported by governments. Their activities are part of cyber espionage, intelligence gathering, or cyber warfare. These attackers target critical infrastructure, military systems, financial institutions, and government networks. Nation-state attacks are advanced, stealthy, and long-term, making them very difficult to detect and defend against. They often use custom malware and sophisticated techniques.

**Example -**

A government-backed hacking group infiltrates another country's power grid systems to gather intelligence or prepare for future disruption. Such an attack may remain hidden for months or years, posing serious risks to national security and public safety.

## 4. Common Attack Surfaces in Cyber Security

An attack surface refers to all the possible points where an attacker can try to enter, exploit, or attack a system. The larger the attack surface, the more opportunities attackers have to exploit vulnerabilities. Modern systems use multiple technologies such as web applications, mobile apps, APIs, networks, and cloud infrastructure, which increases the attack surface. Understanding these attack surfaces helps organizations identify risks and apply proper security controls.

### ❖ Web Applications

**Define -**

Web applications are websites or online portals that users access through a browser. Because they are publicly available on the internet, they are one of the most targeted attack surfaces. Web applications accept user inputs such as login details, search queries, and form data. If these inputs are not properly validated, attackers can exploit vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), or broken authentication. Outdated frameworks and weak session management also increase risk.

**Example -**

An online banking website has a login page where users enter their username and password. If the website does not properly protect its database queries, an attacker can enter malicious input instead of a username. This allows the attacker to bypass login checks and access customer account details. Once inside, the attacker can view balances or transaction history, leading to a serious data breach.

---

### ❖ Mobile Applications

**Define -**

Mobile apps are widely used for banking, social media, food delivery, and shopping. These apps store sensitive data such as authentication tokens, user IDs, and sometimes cached passwords. If data is stored insecurely or communication with servers is not encrypted, attackers can steal this information. Fake or modified apps are also a common threat, as users may unknowingly install malicious apps.

**Example -**

A fake mobile banking app is designed to look exactly like the original app. Users download it from an unofficial app store and log in using their real banking credentials. The app sends these credentials to the attacker’s server. The attacker then uses the stolen details to access the real bank account and transfer money.

---

❖ **APIs (Application Programming Interfaces)**

**Define -**

APIs are used to connect mobile apps, web apps, and backend services. They expose important system functions such as fetching user data or processing payments. If APIs are not properly secured, attackers can manipulate API requests to access unauthorized data. Common problems include missing authentication, weak authorization checks, and no rate limiting.

**Example -**

A food delivery app uses an API to fetch order details. The API request contains a user ID. If the API does not verify whether the requester is allowed to access that user ID, an attacker can change the ID value in the request and view order details of other users, including addresses and phone numbers.

---

❖ **Networks**

**Define -**

Networks connect devices, servers, and applications. Weak network security such as open ports, weak Wi-Fi passwords, and unencrypted traffic makes networks vulnerable. Attackers can intercept data, impersonate users, or overload networks with traffic to disrupt services.

**Example -**

A user connects to free public Wi-Fi at an airport. An attacker on the same network captures unencrypted traffic using sniffing tools. The attacker steals the user's email login credentials and later accesses the email account to reset passwords for other services.

---

❖ **Cloud Infrastructure**

**Define -**

Cloud infrastructure includes virtual servers, databases, storage, and services hosted on platforms like AWS, Azure, or Google Cloud. Misconfigurations such as publicly accessible storage, weak access controls, or exposed credentials are common attack vectors. Because cloud services are accessible from anywhere, attackers actively scan for these mistakes.

**Example -**

A company stores customer documents in a cloud storage bucket. Due to a configuration error, the bucket is publicly accessible. An attacker discovers the open bucket through automated scans and downloads thousands of customer records, causing a major data breach and legal consequences

## 5. OWASP Top 10 Web Application Vulnerabilities

Define -

- Vulnerabilities is an authoritative list published by the Open Web Application Security Project (OWASP) that highlights the ten most critical security flaws commonly found in web applications. It serves as a guideline for developers and organizations to identify major risks and implement effective security controls to safeguard applications against cyber threats.

- **Broken Access Control**

  **Define -**
  Broken access control occurs when an application **fails to enforce restrictions** on what users can do or access. Normal users might reach admin pages, view other users' data, or perform restricted actions. It often happens due to missing permission checks, predictable URLs, or weak role-based security. Attackers can exploit it to steal, modify, or delete sensitive information. This can also lead to privilege escalation where attackers act as admins. Both web and mobile apps are affected. Regular testing, role enforcement, and secure coding are essential to prevent it.

  **Example -**
  A social media site had user profiles like /user/101. An attacker changed the URL to /user/102 and accessed someone else's private posts and messages. Millions of profiles were exposed. Attackers could even post content on others' accounts. The company faced privacy complaints and reputation loss. This shows that weak access control can fully compromise user data.

---

- **Cryptographic Failures**

  **Define -**
  Cryptographic failures occur when sensitive data such as passwords, tokens, or financial information is **not encrypted properly**. Weak algorithms, plain text

storage, or missing HTTPS make it easy for attackers to steal or modify data. It affects data confidentiality and integrity. Proper encryption, key management, and secure transmission protocols are necessary. Failing to implement cryptography correctly exposes even otherwise secure systems to theft. Sensitive data can be stolen in transit or at rest if cryptography is weak

**Example -**
A shopping website stored passwords in plain text and didn't use HTTPS. Hackers on public Wi-Fi captured user credentials. Many accounts were hacked, and credit card info was stolen. Users faced financial loss, and the website suffered reputational damage. This shows why encryption and secure protocols are critical.

---

- **Injection**

  **Define -**
  Injection vulnerabilities happen when **untrusted user input is treated as code** by the system. Common types include SQL Injection, Command Injection, and LDAP Injection. Lack of input validation or escaping allows attackers to manipulate databases, execute commands, or bypass authentication. Injection attacks can compromise confidentiality, integrity, and availability of data. They are extremely dangerous because they directly target backend systems. Parameterized queries, input validation, and escaping special characters can prevent them.

  **Example -**
  An online login form did not validate input. An attacker entered '; DROP TABLE users; -- into the password field. The database executed the command and deleted the users table. Customer data, order history, and personal info were lost. The website went offline, disrupting business. This demonstrates how injection can cause massive damage.

---

- **Insecure Design**

  **Define -**
  Insecure design occurs when the **application is built without considering security** from the start. Weak logic, missing checks, or lack of restrictions make the app vulnerable. These flaws are harder to fix later because they exist in the app's core architecture. Attackers can exploit design weaknesses for account takeover, data theft, or privilege escalation. Secure coding standards, threat

modeling, and design reviews are necessary. It affects both frontend and backend systems and is often overlooked in early development.

**Example -**
A banking app allowed unlimited login attempts without account lockout. Attackers guessed PINs and withdrew money. The flaw was in the design, not coding. Users lost funds, and the bank faced complaints. Security had to be redesigned to add limits and MFA. This shows why designing security from the start is crucial.

---

- **Security Misconfiguration**

**Define -**
Security misconfiguration happens when servers, cloud storage, or applications **are not configured securely**. Examples include default passwords, debug mode enabled, or open cloud buckets. Attackers exploit these misconfigurations to gain unauthorized access. Misconfiguration can also expose sensitive data or backend systems. Regular audits, patching, and disabling unnecessary features are essential. This vulnerability is common in cloud, web servers, and APIs. Proper monitoring reduces risk.

**Example -**
A company left a cloud storage bucket public. Hackers downloaded millions of customer records. Names, emails, and phone numbers were exposed. Data was sold online, causing legal issues. The company faced fines and lost customer trust. This shows how misconfiguration can lead to massive breaches.

---

- **Vulnerable and Outdated Components**

**Define -**
Using **old or vulnerable software, libraries, or frameworks** can create serious risks. Attackers often scan for known vulnerabilities in outdated components. This includes old plugins, APIs, or server software. Exploiting these can compromise data, take control of servers, or allow malware installation. Regular updates, patching, and monitoring dependencies are critical. Even if the app code is secure, outdated components can make it vulnerable

**Example -**
A website used an old version of Apache Struts. Hackers exploited a known vulnerability and executed commands on the server. Customer data was stolen,

and the site went offline. The breach caused financial and reputational damage. This shows why outdated components are very dangerous.

---

- **Identification & Authentication Failures**

  **Define -**
  This happens when login and authentication systems are **weak or improperly implemented**. Weak passwords, missing multi-factor authentication, or stolen session tokens can let attackers impersonate users. It can lead to account takeover and data breaches. Secure password policies, MFA, session expiration, and monitoring failed logins are essential. Both web and mobile apps are affected. Weak authentication is one of the easiest ways for attackers to gain access.

  **Example -**
  A website allowed weak passwords like "123456" and didn't support MFA. Attackers used password lists to break in easily. Several user accounts were accessed, and sensitive data was stolen. The company faced user complaints and lost reputation. This shows why strong authentication is vital.

---

- **Software & Data Integrity**

  **Define -**

  This occurs when an application **trusts unverified data or software**. Attackers can tamper with updates, APIs, or external components. It affects data integrity and may allow malware installation. Systems must verify digital signatures, check data integrity, and validate inputs. Failing this can let attackers inject malicious code or corrupt data. Both client-side and server-side data are at risk.

  **Example -**
  An auto-update system downloaded updates from the internet without checking signatures. Attackers replaced the update with malware. Users' devices were infected, stealing sensitive information. The company had to recall the update and patch the system. This shows how integrity failures compromise trust in software.

---

- **Security Logging & Monitoring Failures**

  **Define -**
  This occurs when applications **do not log important security events** or fail to

monitor them. Attacks may go unnoticed, allowing attackers to continue undetected. Lack of alerts, missing audit trails, and unmonitored logs are common causes. Effective logging helps detect breaches, analyze attacks, and meet compliance requirements. Both successful and failed security events should be recorded. Without it, damage can go on for months before discovery.

**Example -**
A company suffered a breach, but their logs didn't record failed login attempts or privilege changes. Attackers had access for months without detection. Millions of records were stolen before discovery. The company faced legal fines and reputational damage. Proper logging and monitoring could have prevented long-term impact.

---

- **Server-Side Request Forgery (SSRF)**

  **Define -**
  SSRF occurs when an application **makes requests to internal servers based on user input** without proper validation. Attackers can use this to access internal resources behind firewalls. It can expose sensitive internal data, cloud metadata, or admin interfaces. Proper input validation, allowlists, and network segmentation help prevent SSRF. It is particularly dangerous in cloud environments. Both web apps and APIs are commonly affected.

  **Example -**
  A web app fetched preview data for a user-submitted URL. An attacker submitted an internal URL (169.254.169.254) to access cloud metadata. The server returned secret cloud credentials. Attackers used these credentials to access other services. The company faced a serious cloud security breach. This shows SSRF can give attackers internal access from outside the network.

## 6. Mapping Daily-Used Applications to Possible Attack Surfaces

➢ This section explains how commonly used applications such as email, messaging, and banking apps can be exposed to different attack surfaces at various levels.

- **Email Applications (Gmail, Outlook)**

  **Define -**
  Email apps handle sensitive information like messages, contacts, and

attachments. They are vulnerable through **web, mobile, network, and APIs** attack surfaces. Phishing, malware, or unsecured networks can allow attackers to access accounts. Strong passwords, multi-factor authentication, and encrypted communication are essential to protect users.

**Example -**
On public Wi-Fi, hackers can sniff traffic and capture login credentials. Phishing emails may trick users into entering passwords on fake login pages. If compromised, private messages and sensitive information can be stolen.

---

- Messaging Apps (WhatsApp, Telegram)

  **Define -**
  Messaging apps are mainly vulnerable through **mobile and network attack surfaces**. Cloud backups and APIs also expose data to risk. Malware or insecure apps on the device can access sensitive messages. End-to-end encryption protects messages, but cloud backups may still be insecure if not properly encrypted.

  **Example -**
  If a phone is infected with malware, attackers can read messages. Misconfigured cloud backups may leak chat history. Public Wi-Fi can allow attackers to intercept messages in transit.

---

- **Banking Applications**

  **Define -**
  Banking apps are exposed via **mobile, web, API, network, and cloud** attack surfaces. They handle financial and personal data, making them high-value targets. Weak encryption, insecure storage, or phishing attacks put accounts and money at risk. Strong authentication, secure coding, and encryption are essential.

  **Example -**
  Attackers can intercept transactions on public Wi-Fi. Insecure app storage can leak passwords. Fake login pages can trick users and allow attackers to steal funds.

---

- **Social Media Apps (Instagram, Facebook)**

  **Define -**
  Social media apps are vulnerable through **web, mobile, APIs, and cloud** attack surfaces. Profile data, posts, and private messages are sensitive. Weak authentication, XSS, or IDOR attacks can expose user information. Using strong passwords and two-factor authentication helps reduce risk.

  **Example -**
  Attackers can manipulate URLs to view other users' profiles (IDOR). Misconfigured cloud storage can leak images. Fake login pages can steal user credentials.

---

- **Online Shopping Apps (Amazon, Flipkart)**

  **Define -**
  Shopping apps expose **web, mobile, API, network, and cloud** attack surfaces. Payment information, addresses, and order history are sensitive. Vulnerabilities like SQL Injection, XSS, and network sniffing are common risks. Encryption and secure development practices are critical to protect users.

  **Example -**
  Attackers on public Wi-Fi can intercept payment data. Exploiting API vulnerabilities can reveal customer orders. Misconfigured cloud storage can leak invoices or personal data.

## 7. End-to-End Application Data Flow

Data Flow: User → Application → Server → Database

- Data Flow

  Define -

  Data Flow refers to the systematic movement of information through a system, from the user to the application, then to the server, and finally to the database. Understanding data flow allows organizations to identify potential vulnerabilities and implement effective security controls to protect sensitive information.

## 1️⃣ User

**Define -**
The user is the person interacting with the system through a device like a computer, smartphone, or tablet. Users provide inputs such as login credentials, search queries, orders, or messages. They initiate requests that the system must process. Users rely on applications to communicate with servers and databases. They are the **starting point** of the data flow and can also be the source of potential errors or security risks if they enter malicious input.

**Example -**

- A user logging into a banking app with username and password.
- A student filling an online admission form.
- A customer placing an order on Amazon or Flipkart.
- A person uploading a photo on Instagram.
- A user sending a message on WhatsApp.

---

## 2️⃣ Application

**Define -**
The application is the software interface through which users interact with the system. It can be a web app, mobile app, or desktop application. The application collects user input, validates it, formats it, and optionally encrypts sensitive data before sending it to the server. It acts as a bridge between the user and the backend server. The application also displays the server's response to the user in a readable format.

**Example -**

- Mobile banking app validating user login before sending it to the server.
- WhatsApp app encrypting messages before sending to the server.
- Instagram app compressing and resizing a photo before upload.
- Online shopping app checking cart items before placing an order.
- College portal web app verifying required fields in a form.

---

## 3️⃣ Server

**Define -**
The server receives requests from the application and processes them. It applies business logic, checks authentication and authorization, and prepares queries for the database. The server ensures data is processed securely, efficiently, and according to rules defined by the application. It acts as the central processing unit, coordinating between the application and the database. Secure communication between server and application is essential to prevent data leaks or tampering.

**Example -**

- Bank server verifying user credentials against the database.
- E-commerce server calculating total order price and checking inventory.
- Email server storing messages in the database and forwarding them to recipients.
- Social media server updating user feeds after a post.
- Cloud storage server saving files and updating metadata.

---

## 4️⃣ Database

**Define -**
The database stores and manages all persistent data for the application. It handles queries from the server to retrieve, insert, update, or delete information. Databases enforce **data integrity, consistency, and access control**. They are critical for secure storage and reliable retrieval of data. Both structured (SQL) and unstructured (NoSQL) databases are used depending on the application's needs.

**Example -**

- Bank database storing account balances and transaction history.
- E-commerce database storing customer orders and product inventory.
- Email database storing user inbox and sent messages.
- Social media database storing posts, images, and comments.
- Cloud storage database storing uploaded documents and file metadata.

## 8. Attacks in Data Flow: User → Application → Server → Database

- In any online system, data continuously flows from the user to the application, then to the server, and finally to the database where it is stored or processed. This flow happens when we log in, send messages, shop online, or use banking apps. Each step of this data flow can be attacked if proper security is not applied. Hackers target weak points to steal data, change information, or disrupt services.

### 1. Attacks at the User Level (User → Application)

**Explanation**

The user layer is the starting point of the data flow. Users interact with the system through mobile phones, laptops, or desktops. They enter sensitive information such as usernames, passwords, personal details, and payment information. Since this layer involves human interaction, it is highly vulnerable to manipulation.

**Where attacks happen**

Attacks happen when users trust fake messages, use weak passwords, or operate on infected devices. Hackers take advantage of user mistakes rather than breaking technical security.

**Common attacks with real-life examples**

**Phishing attack:** A user receives a fake email claiming to be from a bank and clicks a link. The fake website looks real, and when the user enters login details, the attacker steals them.

**Malware/keylogger:** A user downloads pirated software that installs a keylogger. Every password typed is recorded and sent to the attacker.

**Fake mobile apps:** A fake payment app steals card details before forwarding data.

**Why this stage is dangerous**

If user credentials are stolen, attackers can log in as genuine users. Even a highly secure system cannot protect itself if the user willingly gives access.

---

**2. Attacks at the Application Layer (Frontend / Client Side)**

**Explanation**

The application layer acts as a bridge between the user and the server. It collects input from users, performs basic validation, and sends requests to the backend server. This includes websites and mobile apps that are exposed to the internet.

**Where attacks happen**

Attacks occur when the application fails to properly validate input or relies too much on client-side checks. Since attackers can inspect and modify application code, this layer is a common target.

**Common attacks with real-life examples**

**Cross-Site Scripting (XSS):** An attacker enters a malicious script in a comment section. When other users view the comment, their session cookies are stolen.

**Cross-Site Request Forgery (CSRF):** A user logged into a banking website clicks a malicious link, which secretly triggers a money transfer.

**Client-side manipulation:** A user changes the price of a product in the browser before purchasing.

**Why this stage is dangerous**

If malicious data passes through the application layer, it reaches the server and database. One vulnerable application can affect thousands of users at once.

---

**3. Attacks During Data Transmission (Network Layer)**

**Explanation**

At this stage, data travels from the user to the application, then to the server and database through networks like the internet or Wi-Fi. This communication must be protected to prevent interception.

**Where attacks happen**

Attacks happen when data is transmitted without encryption or over public and unsecured networks.

**Common attacks with real-life examples**

**Man-in-the-Middle (MITM):** A user accesses a banking app on public Wi-Fi. The attacker intercepts login credentials.

**Packet sniffing:** Attackers capture network traffic to read usernames and passwords.

**Session hijacking:** Stolen session tokens allow attackers to access accounts without passwords.

**Why this stage is dangerous**

Attackers can silently read or modify data in transit without accessing the system directly. Sensitive data can be stolen without detection.

---

**4. Attacks at the Server Layer (Backend / Business Logic)**

**Explanation**

The server processes user requests, enforces authentication and authorization, applies business rules, and communicates with the database. It is the core of the system.

**Where attacks happen**

Attacks occur when server-side validation is weak, APIs are insecure, or access controls are improperly implemented.

**Common attacks with real-life examples**

**SQL Injection:** An attacker enters malicious SQL code in a login form and gains access to the database.

**Privilege escalation:** A normal user changes a URL to access admin pages.

**Denial of Service (DoS/DDoS):** Attackers send大量 fake requests to crash the server.

**Why this stage is dangerous**

A compromised server gives attackers control over the application and access to sensitive data. Server attacks often lead to database breaches.

---

**5. Attacks at the Database Layer (Data Storage)**

**Explanation-**

The database stores critical information such as user details, passwords, financial records, and logs. It is the final destination of the data flow.

**Where attacks happen**

Attacks happen when databases are misconfigured, exposed to the internet, or lack proper encryption and access controls.

**Common attacks with real-life examples**

**Data breaches:** Attackers steal millions of user records from insecure databases.

**Insider attacks:** An employee misuses access to copy customer data.

**Backup leaks:** Stolen database backups expose sensitive data.

**Why this stage is dangerous**

A database breach causes massive financial loss, legal penalties, and loss of user trust. The damage is often long-term and difficult to recover.