UNIT – 6 Database Security

Introduction to Database Security Issues

Types of Security

Database security is a broad area that addresses many issues, including the following:

1. Legal and Ethical Issues

Some data is protected by **laws** or **ethical rules**.
Examples:

- Personal data
- Medical records
- Financial information

Such data cannot be accessed by unauthorized people. Many countries (like the U.S.) have privacy laws to protect this information.

2. Policy Issues

Organizations and governments set **rules** about what information must remain confidential.
Examples:

- Credit ratings
- Personal health records
- Military or strategic data

These policies define what information can or cannot be made public.

3. System-Related Issues

Security can be enforced at different system layers:

- **Hardware level**
- **Operating system level**
- **DBMS level**

A key decision is **where** security checks should be implemented to be most effective

4. Multi-Level Security and Classification

Some organizations use **security clearance levels**, such as:

- Top Secret
- Secret
- Confidential

- Unclassified

Data and users are assigned levels, and access is granted only when the user's level is high enough. The DBMS must enforce this classification system.

Threats to Databases.

Database threats can harm the three key security goals: **integrity**, **availability**, and **confidentiality**.

1. Loss of Integrity

Integrity means the data must be **correct**, **accurate**, and **unchanged** except by authorized users.

- Data loses integrity when it is **modified incorrectly**, **changed by unauthorized users**, or **corrupted by mistakes**.
- Modifications include: inserting, updating, deleting, or altering data.
- If corrupted data is used, it can lead to:
    - wrong decisions
    - fraud
    - inaccurate reports

Example: A hacker changing bank account balances, or an employee accidentally deleting important records.

2. Loss of Availability

Availability means authorized users and programs must be able to **access the database when needed**.

Loss of availability happens when the database is **unreachable** due to:

- system crash
- network failure
- denial-of-service attacks
- power outages
- incorrect system configuration

Example: A hospital cannot access patient records during an emergency because the database server is down.

3. Loss of Confidentiality

Confidentiality means data must be protected from **unauthorized access or disclosure**.

If confidential data is exposed, it can cause:

- violation of privacy laws (e.g., Data Privacy Act)

- loss of public trust
- legal consequences
- national security risks

Example: Leaking customer credit card numbers or exposing classified government documents.

Database Security: Not an Isolated Concern.

A database cannot protect itself alone. It is part of a larger system that includes:

- applications
- web servers
- firewalls
- SSL/security layers
- monitoring tools

If any part of this system is weak, the database can be compromised, even if the DBMS itself is secure.
Security is only as strong as the *weakest link*.

Four Types of Control Measures Used to Protect Databases

To guard against threats (loss of integrity, availability, and confidentiality), systems use:

1. **Access control**
2. **Inference control**
3. **Flow control**
4. **Encryption**

Why Access Control Is Needed

In multiuser environments, different users should only see the data they are allowed to see. For example, in a company, most employees should *not* access salary information.

A DBMS provides a **security and authorization subsystem** to enforce this.

Two Main Types of Database Security Mechanisms

1. Discretionary Security

- Administrators *grant specific privileges* to users.
- Privileges may include: read, insert, update, delete.
- Access can be controlled at file, record, or field level.
- Flexible, commonly used in most DBMSs.

2. Mandatory Security

- Used in environments requiring **multilevel security** (e.g., military, government).

- Data and users are assigned security levels (e.g., Top Secret → Secret → Confidential).
- Users can only access data at or below their clearance level.
- Includes **role-based access control**, where permissions depend on a user's job/role.

Control Measures

To protect databases from unauthorized access, data leaks, and misuse, **four main types** of **security control measures** are used:

1. Access Control

- Prevents **unauthorized people** from entering or using the database system.
- Uses **user accounts and passwords** to control who can log in.
- Ensures only authorized users can access specific data.

2. Inference Control

- Used in **statistical databases** (for example, population surveys).
- Users can see **summary statistics**, but must not be able to infer details about **individuals**.
- Prevents attackers from deducing personal information indirectly.
- Protects privacy when only aggregated data should be revealed.

3. Flow Control

- Ensures information does **not flow** to users who are not authorized to receive it.
- Prevents data leaks through **covert channels**—hidden or unintended communication paths that violate security policy.
- Helps stop sensitive information from being transferred indirectly.

4. Data Encryption

- Protects sensitive data by **converting it into unreadable form**.
- Only users with the decryption key can read the data.
- Used especially for:
    - credit card numbers
    - personal identification numbers
    - medical or personal information
- Required by law in many places for systems storing personal data.
- Includes techniques like:
    - **Public key encryption**
    - **Digital signatures**

Database Security and the DBA

The **Database Administrator (DBA)** is the **main authority** responsible for **managing** and **securing** the database system.

The DBA has a special **superuser account** with powerful commands that normal users do not have.

Key Security Responsibilities of the DBA

The DBA **handles four main types** of **actions**:

1. Account Creation

- Creates new user accounts and passwords.
- Controls who is allowed to access the DBMS at all.

2. Privilege Granting

- Gives users specific **permissions** such as:
  - read
  - insert
  - update
  - delete
  - use certain tables or features
- Ensures users only get the access they need.

3. Privilege Revocation

- **Removes permissions** that were previously granted.
- Important for security when:
  - employees leave
  - roles change
  - privileges were given by mistake

4. Security Level Assignment

- Assigns users to **security clearance levels** (for mandatory access control).
- Ensures users **can only access data** allowed by their security classification.

Access Control, User Accounts, and Database Audits

To ensure database security, **access** must be **controlled**, **users** must be **identifiable**, and **actions** must be **traceable**.

1. User Accounts and Login

- Anyone who needs to use the database must apply for a **user account**.
- The **DBA creates the account** and assigns a **password** if the person has a valid need.
- To access the DBMS, the user must **log in** with:
  - account number
  - password
- Application programs also need accounts and must log in.

The DBMS checks the login information to ensure the user is authorized.

2. Storing Accounts and Passwords

- The system keeps an **encrypted table** with:
  - AccountNumber
  - Password
- When accounts are created or deleted, the table is updated.
- This table ensures only valid users can access the database.

3. Tracking User Activities

During each login session, the DBMS keeps track of:

- which **user account** is logged in
- which **computer/device** they are using
- **all operations** they perform
- especially **update operations**, since these can change or damage data

This makes it possible to trace who did what.

4. Audit Trail Using the System Log

- The system log (used for recovery) can be expanded to include:
  - user account number
  - device/computer ID
  - every operation performed
- This extended log acts as an **audit trail**.

5. Database Audits

- A **database audit** examines **the log to review all actions** during a period.
- If the database is tampered with or something suspicious happens, the DBA can:
  - inspect the log
  - identify which user account performed the unauthorized action

Audits are crucial for sensitive systems (e.g., banks) where many users make frequent updates.

Sensitive Data and Types of Disclosures

1. What Makes Data Sensitive

Sensitive data is data that requires protection due to its importance or confidentiality. It can be sensitive for several reasons:

1. **Inherently sensitive** – The value of the data itself is confidential (e.g., salary, medical conditions).
2. **From a sensitive source** – The source requires secrecy (e.g., a confidential informant).
3. **Declared sensitive** – The data owner explicitly marks it as sensitive.
4. **Sensitive attribute/record** – Specific fields or records are sensitive (e.g., salary attribute).
5. **Sensitive in combination** – Data may become sensitive when combined with other data (e.g., exact location coordinates revealing a secret event).

2. Who Controls Access

- The **DBA and security administrator** enforce policies for who can access which data.
- Before revealing data, three factors are considered:

1. **Data availability** – Ensure users **don't see partially** updated or **inaccurate** data (handled by concurrency control).
2. **Access acceptability** – Only authorized users can access data; some queries may be denied if they indirectly reveal sensitive data.
3. **Authenticity assurance** – Access may depend on user characteristics (e.g., working hours, prior query history), especially for statistical databases.

3. Security vs. Precision

- **Security:** Protect all sensitive data, allowing access only to nonsensitive data.
- **Precision:** Make as much nonsensitive data available as possible while still protecting sensitive data.
- **Tradeoff:** Maximum security often reduces precision; ideal systems balance both.

Relationship Between Security and Privacy

- **Security: Protects** systems and data from **unauthorized access** (authentication, encryption, access control, firewalls). It ensures **safe access** to information.
- **Privacy:** Focuses on **appropriate use of personal information according** to **user consent** and expectations.

    - Users should know what data is collected, how it is used, and have the option to approve or disapprove.
- **Trust:** Security is a foundation for privacy. Users trust a system more when both **security and privacy** are maintained.

Discretionary Access Control Based

Discretionary Access Control is a **database security mechanism** that allows the **DBA or owners of database objects to grant or revoke privileges to users** based on need. The "**discretionary**" part **means** that **access is controlled** at the **discretion of the owner** of the data.

1. Levels of Privileges

Privileges in a database can be assigned at two main levels:

a) Account-Level Privileges

- Apply to the user account itself, not specific tables.
- Examples:
    - **CREATE TABLE / CREATE SCHEMA** – create new relations or schemas.
    - **CREATE VIEW** – create virtual tables.
    - **ALTER** – modify schema (add/remove columns).
    - **DROP** – delete tables/views.
    - **MODIFY** – insert, update, delete data.
    - **SELECT** – retrieve data.
- These privileges govern what actions an account can perform in general.

b) Relation (Table) Level Privileges

- Apply to specific tables (relations) or views.
- Examples:
    - **SELECT** – retrieve tuples from a table.
    - **INSERT / UPDATE / DELETE** – modify tuples in a table.
    - **REFERENCES** – use table attributes in integrity constraints.
- Can also be restricted to specific columns.

Owner accounts of a table have **all privileges** on that table by default.

2. Using Views for Access Control

- **Views** can **restrict** user access to **certain attributes or rows**.
- **Example**: If account B should only see the `name` and `salary` of employees, the owner creates a **view** with just those columns and grants SELECT privilege on the view.

3. Revoking Privileges

- Users may be **granted privileges temporarily** for specific tasks.
- SQL provides the **REVOKE** command to **remove privileges** when no longer needed.

4. Propagation of Privileges (GRANT OPTION)

- Owners can allow a user to **grant privileges to others** by giving them the **GRANT OPTION**.
- Example:
  - A1 grants SELECT to A3 **with GRANT OPTION**, so A3 can grant it to A4.
  - If A1 revokes A3's privilege, all privileges propagated from A3 are also revoked automatically.
- Privileges can come from multiple sources; revocation only removes privileges granted by a specific source.

5. Limiting Privilege Propagation

Two types of limits can be applied (though most DBMSs don't implement this):

a) Horizontal Propagation

- Limits the **number of accounts** a privilege can be granted to.
- Example: Horizontal = 1 → a user can grant to only 1 other account.

b) Vertical Propagation

- Limits the **depth of grant chains**.
- Example: Vertical = 2 → the user can grant the privilege to another user **but reduce vertical propagation by 1** when passing it on.

Mandatory Access Control and Role-Based Access Control for Multilevel Security

Why Mandatory Access Control Is Needed

Traditional relational DBMS security relies mainly on **Discretionary Access Control (DAC)**:

- Users are **granted or revoked privileges** (SELECT, INSERT, UPDATE, DELETE).
- Access is **binary**: either you have a privilege or you do not.

However, **DAC alone is insufficient** in environments where:

- Data has **different sensitivity levels**
- Users have **different clearances**
- Preventing **information leakage** is critical

Such environments include:

- Government, military, intelligence systems
- Corporate systems handling **personally identifiable information (PII)**

This leads to **Mandatory Access Control (MAC)**, where access is determined by **security classifications**, not user discretion.

Security Classifications and Multilevel Security

Typical **security levels:**

```
Top Secret (TS) ≥ Secret (S) ≥ Confidential (C) ≥ Unclassified (U)
```

- **Subjects**: users, programs, accounts
- **Objects**: relations, tuples, attributes, views

Each subject and object is assigned a **security classification**:

- `class(S)` → clearance of subject
- `class(O)` → classification of object

Bell–LaPadula Model (Core of MAC)

The **Bell–LaPadula model** enforces two fundamental rules:

1. Simple Security Property ("No Read Up")

A subject **cannot read** data at a higher classification:

```
class(S) ≥ class(O)
```

Prevents users from seeing more sensitive data than they are cleared for.

2. Star Property (*-Property) ("No Write Down")

A subject **cannot write** data to a lower classification:

```
class(S) ≤ class(O)
```

Prevents sensitive data from leaking to lower levels.

Example
A TS-cleared user must not write TS data into a U-level object—otherwise, confidential information would become widely accessible.

Filtering: Different Views for Different Users

Users with different clearances see **different versions** of the same relation.

Example: `SELECT * FROM EMPLOYEE`

- **Secret (S) user**: sees all data with classification ≤ S
- **Confidential (C) user**:
  - Attributes classified S appear as **NULL**
- **Unclassified (U) user**:
  - Only U-classified attributes appear
  - Everything else is NULL

Filtering prevents unauthorized disclosure while maintaining database consistency.

Polyinstantiation

Polyinstantiation allows **multiple tuples with the same apparent key** but different:

- Attribute values
- Security classifications

This occurs when lower-level data **cannot be derived (filtered)** from higher-level data.

Why It Is Necessary

Consider this update by a **C-level user**:

```
UPDATE EMPLOYEE
SET Job_performance = 'Excellent'
WHERE Name = 'S';
```

- The C-level view shows `Job_performance = NULL`
- Rejecting the update would reveal that a higher-level value exists
- That would create a **covert channel** (unauthorized inference)

Solution:
Create a **new tuple at level C** instead of overwriting the higher-level S tuple.

This preserves:

- Security
- User expectations
- Non-interference between levels

Covert Channels

A **covert channel** occurs when:

- A user infers sensitive information indirectly
- For example, through error messages or rejected updates

High-security systems must:

- Avoid revealing the existence of higher-level data
- Use mechanisms like polyinstantiation instead of rejection

Comparing Discretionary Access Control and Mandatory Access Control

1. Discretionary Access Control (DAC)

What it is

- Access rights are **granted and revoked at the discretion of the data owner**.
- Common in most commercial DBMSs.
- Implemented using privileges such as `SELECT`, `INSERT`, `UPDATE`, `DELETE`.

Strengths

- **Highly flexible**
  - Easy to manage and adapt
  - Suitable for a wide range of applications (business, academic, web systems)
- **Low administrative overhead**
  - No need for strict data classification
  - Objects do not require security labels

Weaknesses

- **Vulnerable to malicious programs**, such as **Trojan horses**
  - Once a user has access, DAC does not control **how data is later used or propagated**
- **No control over information flow**
  - A user can legally read data and then pass it to unauthorized users or programs
- **Weaker protection for highly sensitive data**

Example

A user with read access to confidential data can unintentionally leak it through:

- Malicious software
- Scripts or applications running under their credentials

2. Mandatory Access Control (MAC)

What it is

- Access decisions are based on **system-enforced security policies**
- Users **cannot change permissions**
- Subjects and objects are assigned **security classifications** (e.g., TS, S, C, U)

Strengths

- **Strong security guarantees**
  - Prevents illegal information flow
  - Enforces rules like *no read up* and *no write down*
- **Resistant to Trojan horse attacks**
  - Even malicious programs cannot bypass security rules
- **Ideal for high-security environments**
  - Military, intelligence, government systems

Weaknesses

- **Rigid and inflexible**
  - Requires strict classification of all users and data
- **High administrative burden**
  - Every object must be labeled with a security level
- **Limited applicability**
  - Not suitable for many commercial or dynamic environments

3. Key Difference: Information Flow Control

Feature	DAC	MAC
Who controls access	Data owner	System policy
Flexibility	High	Low
Information flow control	None	Strict
Trojan horse protection	Weak	Strong
Data labeling required	No	Yes
Typical usage	Commercial systems	Military / high-security

4. Why DAC Is Often Preferred in Practice

Although MAC provides **stronger security**, it is often **too restrictive** and costly to deploy.

DAC is commonly preferred because:

- It offers a **better balance between security and usability**
- It works well in environments where:
  - Absolute secrecy is not required

    - Ease of administration and flexibility are important

Summary

- **DAC** is flexible and widely used but vulnerable to misuse and malicious software.
- **MAC** provides strong protection against information leakage but is rigid and costly.
- The choice depends on the **security requirements of the application**:
    - **High-security systems** → MAC
    - **General-purpose systems** → DAC