Database Security Issues

- Legal and ethical issues
  - Right of access
  - Privacy laws
- Policy issues
  - Government, institutional or corporate policies
- System-related issues
  - Where should security be handled: HW, OS or DBMS
- Multiple security levels
  - Categorization of data and users
  - Example: top secret, secret, confidential, unclassified
Threats to Databases

- **Loss of integrity**
  - Data should not be corrupted, through intentional or accidental acts
- **Loss of availability**
  - Data should remain accessible to those who have legitimate access rights
- **Loss of confidentiality**
  - Data should not be accessible to those who do not have legitimate access rights
Database Countermeasures

- **Access control**
  - User accounts and passwords identify database users

- **Inference control**
  - Statistical or summary data may allow users to infer or deduce information. Such inference must not allow inference of data that user is not authorized to access

- **Flow control**
  - Covert channels, which allow data to flow in manners violating security must be blocked

- **Encryption**
  - Encryption protects sensitive data during storage and transmission
  - Passwords, SSNs, credit card information ...
Database Security Mechanisms

- **Discretionary** security mechanisms
  - Privilege grants allow specific users to perform specific operations on specific data
  - Initial grants start with DBA
  - Grants may be passed on between users

- **Mandatory** security mechanisms
  - Enforce multi-level security
  - Data and users are classified into security classes
  - Typically, user can only see data which has a lower (or same) classification as themselves
  - Role-based security is similar
The **DBA** is the central authority for managing a database system ... thus responsible for overall security.

- **Security responsibilities**
  - granting privileges to users who need to use the system
  - classifying users and data in accordance with the policy of the organization

- **System / root / superuser** account allows:
  - Account creation – access control
  - Privilege granting – discretionary
  - Privilege revocation – discretionary
  - Security level assignment – mandatory
Access Protection and Audits

- Login Session: user logs in with account/password

- DBMS tracks all operations applied by a user throughout each login session.
  - Can be tracked in system log, which records all operations for recovery from a transaction failure or system crash.
  - A log used primarily for security purposes is an audit trail

- A database audit is performed when tampering is suspected
  - Logs are reviewed to try to identify what happened and who did it
Discretionary Access Control
Privileges

• The typical method of enforcing discretionary access control is based on the granting and revoking privileges.

• Account level privileges:
  ▪ DBA specifies the particular privileges that each account holds independently of the relations in the database.

• Relation level (table level) privileges:
  ▪ DBA controls privilege to access each individual relation or view in the database.
Account Level Privileges

- CREATE SCHEMA or CREATE TABLE privilege
- CREATE VIEW privilege
- ALTER privilege
- DROP privilege
- MODIFY privilege execute insert, delete, or update
- SELECT privilege

- Privilege names are based on corresponding SQL commands
- Account level privileges are not specified by SQL standard, left to DBMS to define
Relation Level Privileges

- These privileges are specified by SQL standard

- **SELECT** privilege on R
  - privilege to use the **SELECT** statement to retrieve tuples from R

- **UPDATE, DELETE** and **INSERT** privileges on R:
  - Capability to modify tuples of R
  - Both the **INSERT** and **UPDATE** privileges can specify that only certain attributes can be modified

- **REFERENCES** privilege on R:
  - Capability to **reference** relation R when specifying integrity constraints
  - Can also be restricted to specific attributes of R
The Access Matrix Model is a way of keeping track of discretionary privileges.

- **Rows** represents **subjects**
  - (users, accounts, programs)

- **Columns** represent **objects**
  - (relations, records, columns, views, operations)

- Each position $M(i,j)$ represents the types of privileges (read, write, update) that **subject i** holds on **object j**
Privilege Control

- Each relation R in a database is assigned an **owner account**
  - typically, the account used when the relation initially created

- Owner of a relation is given **all privileges** on that relation.
  - In SQL2, the DBA can assign an owner to a whole schema by creating the schema, using the **CREATE SCHEMA** command

- Account owner can **pass privileges** on to other users by **granting** privileges to their accounts
Specifying Privileges Using Views

- **Views** are often used for discretionary authorization
  - Example: owner A of a relation R wants to give account B read access to *some fields* of R
    A can create view V of R that includes *only those attributes* and then grant SELECT on V to B
  - Example: owner A of a relation R wants to give account B read access to *some rows* of R
    A can create view V’ by means of a query that selects only those tuples from R that A wants to allow B to access and then grant SELECT on V’ to B

- Creating a view requires SELECT privilege on all relations involved in the view definition.
Revoking Privileges

- Revoking privileges takes them away
  - Sometime it is desirable to grant a privilege to a user temporarily, then revoke it
  - Example: The owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed.
Propagating Privileges

- When A grants a privilege to B, that privilege can be given with or without the **GRANT OPTION**.
- If the **GRANT OPTION** is given, B can also grant that privilege to other accounts.

- If B then grants the privilege to C, also with **GRANT OPTION**, privileges may **propagate** to other accounts without the knowledge of the original owner of the relation.

- If A later revokes the privilege granted to B, all the privileges that propagated through B, based should be automatically **revoked** by the system.
Examples

- DBA creates four accounts: A1, A2, A3, A4
- A1 should be able to create base relations.
  
  DBA must issue the following GRANT command:

  ```
  GRANT CREATETAB TO A1;
  ```

- Same effect can be accomplished by:

  ```
  CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
  ```
Examples

• A1 can now create tables under the schema called EXAMPLE
• A1 creates the two base relations: EMPLOYEE and DEPARTMENT
  ▪ A1 is then owner of these two relations and has all relation privileges on each of them

• A1 grants A2 the privilege to insert and delete tuples in both of these relations, but A2 cannot propagate these privileges to others:

  GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
Examples

- A1 allows A3 to retrieve information from either table and also allows A3 to propagate the privilege to other accounts:
  
  ```
  GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
  ```
  
- A3 can grant the SELECT privilege on the EMPLOYEE relation to A4:
  
  ```
  GRANT SELECT ON EMPLOYEE TO A4;
  ```

A4 can’t propagate the SELECT privilege since GRANT OPTION was not given to A4
Examples

- A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3:
  
  ```sql
  REVOKE SELECT ON EMPLOYEE FROM A3;
  ```

- DBMS must now automatically revoke the SELECT privilege on EMPLOYEE from A4
Examples

- A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation with ability to propagate the privilege
  - limited to retrieve only the NAME, BDATE, and ADDRESS attributes and only for the tuples with DNO=5

```
CREATE VIEW A3EMPLOYEE AS
    SELECT NAME, BDATE, ADDRESS
    FROM EMPLOYEE
    WHERE DNO = 5;

GRANT SELECT ON A3EMPLOYEE TO A3
    WITH GRANT OPTION;
```
Examples

• A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE:

  GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;

• UPDATE or INSERT privilege can specify particular attributes that may be updated or inserted in a relation.
• Other privileges (SELECT, DELETE) are not attribute specific.
example

tuna owns:
  Cities(name, state, population)
  States(name, abbreviation, capital, area, population)

tuna:  GRANT SELECT, UPDATE ON Cities
        TO shark WITH GRANT OPTION;

tuna:  GRANT SELECT ON Cities TO minnow;

tuna:  GRANT SELECT ON States
        TO shark, minnow WITH GRANT OPTION;

shark: GRANT SELECT ON Cities
       TO starfish WITH GRANT OPTION;

shark: GRANT UPDATE (area, population) ON Cities
       TO starfish;

shark: GRANT UPDATE ON States TO starfish;
       (not allowed)

starfish: GRANT SELECT ON Cities TO squid;
A grant graph can help the system keep track of privilege propagation.

Suppose next command is:
shark: `REVOKE SELECT ON Cities FROM starfish`
Limits on Privilege Propagation

- Techniques to limit the propagation of privileges have been developed
  - not implemented in most DBMSs and not a part of SQL
  - Limiting **horizontal propagation** to an integer number \( i \) means that an account B given the GRANT OPTION can grant the privilege to at most \( i \) other accounts.
  - Limiting **vertical propagation** is more complicated, it limits the depth of the granting of privileges
Mandatory Access Control
Mandatory Access Control

- Discretionary access control techniques (grant/revoke privileges) has been the main security mechanism for relational database systems
  - This is an all-or-nothing method:
  - a user either has or does not have a certain privilege

- Many applications require an additional security policy that classifies data and users based on security classes.
  - This approach of mandatory access control, would typically be combined with the discretionary access control mechanisms
Typical security classes:
top secret (TS), secret (S),
confidential (C), unclassified (U)
TS ≥ S ≥ C ≥ U

Bell-LaPadula model classifies
each subject (user, account, program) and
object (relation, tuple, column, view, operation)
into one of the security classifications, T, S, C, or U:
- class(S) → clearance (classification) of a subject S
class(O) → classification of an object O
Multilevel Security

- Two restrictions are enforced on data access based on the subject/object classifications:
  - **Simple security property:** A subject $S$ is not allowed read access to an object $O$ unless $\text{class}(S) \geq \text{class}(O)$
  - Keeps subjects from accessing data above their clearance
  
  - **Star property:** A subject $S$ is not allowed to write an object $O$ unless $\text{class}(S) \leq \text{class}(O)$
    - Keeps subjects from moving data from a high clearance to a lower clearance
Comparing DAC and MAC

- **Discretionary Access Control (DAC) policies:**
  - + high degree of flexibility
  - + suitable for a large variety of application domains
  - - vulnerable to malicious attacks, such as Trojan horses embedded in application programs.

- **Mandatory Access Control (MAC) policies:**
  - + ensure a high degree of protection
  - + prevent illegal flow of information
  - - too rigid - applicable in limited environments

- In many practical situations, DAC is preferred
  - better trade-off between security and applicability
Role Based Access Control
Role-Based Access Control

- Role-based access control (RBAC)
  - emerged rapidly in the 1990s
  - suitable for managing and enforcing security in large-scale enterprise-wide systems
- Permissions are associated with roles, and users are assigned to appropriate roles
  - avoid overhead of managing each individual’s privileges
- Roles are created using CREATE ROLE and DESTROY ROLE commands
  - GRANT and REVOKE commands can then be used to assign and revoke privileges from roles
Role-Based Access Control

- **RBAC** ensures that only authorized users are given access to certain data or resources
- Many DBMSs support roles
- A role hierarchy is a natural way of organizing roles to reflect the organization’s lines of authority and responsibility
- **RBAC** systems may allow temporal constraints on roles
  - time and duration of role activations
  - timed triggering of a role by an activation of another role
CREATE ROLE Bigfish;

GRANT SELECT ANY TABLE
TO Bigfish;

GRANT Bigfish
TO Tuna, Flounder;

Tuna and Flounder now have all privileges available to the BigFish role
E-Commerce Access Control
E-Commerce Access Control

- **E-Commerce environments** (and similar web environments) require elaborate policies
  - beyond traditional DBMS access control
  - e-commerce environment resources include not only data, but also knowledge and experience.
  - Access control mechanism should be flexible enough to support a wide spectrum of heterogeneous objects
E-Commerce Access Control

- Role-based models have promise for addressing the key security requirements of Web-based applications
- In contrast, DAC and MAC models lack capabilities needed to support security requirements of emerging enterprise and Web-based applications.
E-Commerce Access Control

- Heterogeneity of subjects requires access control policies based on user characteristics and qualifications.
  - A possible solution is the notion of credentials
  - A **credential** is a set of properties concerning a user that are relevant for security purposes
    - For example, age, position within an organization
  - XML may play a key role in access control for e-commerce applications
Statistical Database Security
Statistical Database Security

- **Statistical databases** are used mainly to produce statistics on various populations.
- Database may contain **confidential data** on individuals, which should be protected from unauthorized access.
- General users are only permitted to retrieve **statistical information** on the populations, such as **averages, sums, counts, maximums, minimums, and standard deviations**.
- Statistical database security techniques must prohibit the retrieval of individual data.
Statistical Database Security

- **Allowed:**
  - retrieve the *number* of individuals in a population
  - retrieve the *average income* of the population
- **Not Allowed:**
  - retrieve individual data, such as the income of a specific person

- This can be achieved by prohibiting queries that retrieve attribute values and allowing only queries using statistical aggregate functions
In some cases it is possible to **infer** the values of individual tuples from a sequence statistical queries
- particularly true when the conditions result in a population consisting of a small number of objects

**Example:**
- Following are allowable queries:
  
  ```sql
  SELECT COUNT(*) FROM PERSON WHERE <condition>;
  SELECT AVG(INCOME) FROM PERSON WHERE <condition>;
  ```
- Suppose condition on both queries is
  
  ```sql
  Last_degree='Ph.D.' AND Sex='F' AND City='Bellaire' AND State='TX'
  ```
- If first query returns 1, then we have an individual’s income.
- If we can match the condition to that one actual person, we have gained prohibited information about that person
Flow Control
Flow Control

- **Flow control** regulates the distribution or flow of information among accessible objects.
- A **flow** between object X and object Y occurs when a program reads values from X and writes values into Y.
  - Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects.
- A **flow policy** specifies the channels along which information is allowed to move.
  - simplest flow policy specifies just two classes of information: confidential (C) and nonconfidential (N).
  - all flows allowed except those from class C to class N.
Covert Channels

- A **covert channel** allows a transfer of information that violates the security or the policy
  - **allows** information to pass from a higher classification level to a lower classification level through **improper means**
- Two broad categories:
  - **Storage channels** information is conveyed by accessing system information or information otherwise inaccessible to the user
  - **Timing channel** allow the information to be conveyed by the timing of events or processes
- One way to avoid covert channels:
  - programmers to not actually gain access to sensitive data that a program is supposed to process after the program has been put into operation
Encryption
Encryption

- **Encryption** is a means of maintaining secure data in an insecure environment.
- **Encryption** consists of applying an **encryption algorithm** to data using some pre-specified **encryption key**.
- The resulting data has to be **decrypted** using a **decryption key** to recover the original data.