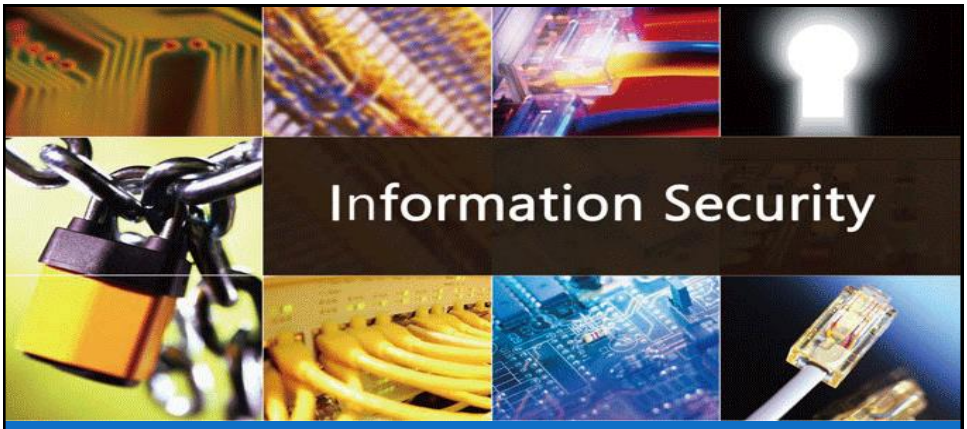



# Database security

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

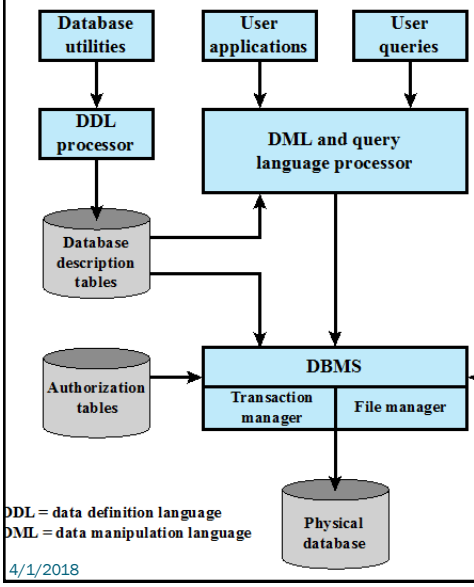
## Objective

- Understand the importance of securing data stored in databases
- Learn how the structured nature of data in databases impacts security mechanisms
- Understand attacks and defenses that specifically target databases

# Importance of Database Security

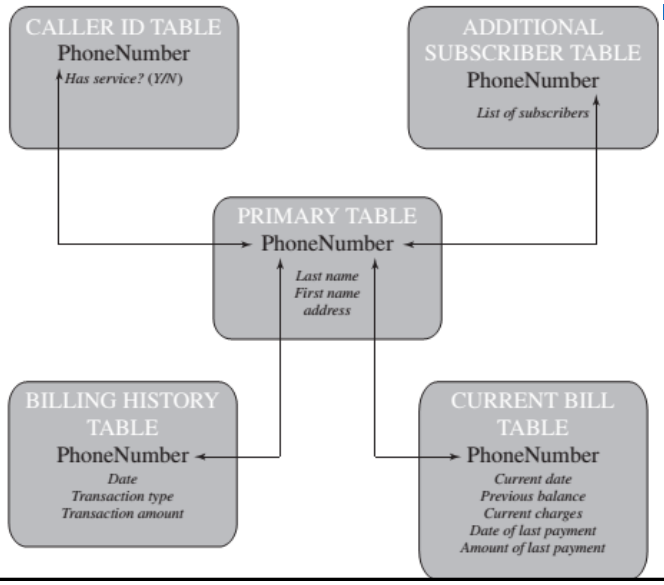
- Why securing data stored in databases so important and different?
  - Databases store massive amounts of sensitive data
  - Data has structure that influences how it is accessed
  - Accessed via queries or programs written in languages like SQL (Structured Query Language)
  - Transactional nature of queries (updates or reads)
  - Derived data or database views

# DBMS Architecture




- DBS security requirements:
  - OS-based security mechanisms: control r/w
  - security packages
- => security services and mechanisms are designed and integrated with DBSs

# Relational Database Systems (RDBS)




# RDBS - Components

- Relational model based database systems are widely used in real-world environments
- A relational database consists of relations or tables
- A table is defined by a schema and consists of tuples
- Tuples store attribute values as defined by schema
- Keys used to access data in tuples

| Formal Name | Common Name | Also Known As |
|---|---|---|
| Relation | Table | File |
| Tuple | Row | Record |
| Attribute | Column | Field |

# RDBS – ex

**Department Table**

| Did | Dname | Dacctno |
|---|---|---|
| 4 | human resources | 528221 |
| 8 | education | 202035 |
| 9 | accounts | 709257 |
| 13 | public relations | 755827 |
| 15 | services | 223945 |

Primary key

**Employee Table**

| Ename | Did | Salarycode | Eid | Ephone |
|---|---|---|---|---|
| Robin | 15 | 23 | 2345 | 6127092485 |
| Neil | 13 | 12 | 5088 | 6127092246 |
| Jasmine | 4 | 26 | 7712 | 6127099348 |
| Cody | 15 | 22 | 9664 | 6127093148 |
| Holly | 8 | 23 | 3054 | 6127092729 |
| Robin | 8 | 24 | 2976 | 6127091945 |
| Smith | 9 | 21 | 4490 | 6127099380 |

Foreign key

Primary key

**Operations on relations (using SQL):**

- Create, select, insert, update, join and delete
- Example: SELECT * FROM EMPLOYEE WHERE DID = '15'
- It returns tuples for Robin and Cody

**Queries written in a query language (e.g., SQL)** use such basic operations to access data in a database as needed.

# Database Administrator (DBA)

- The database administrator (**DBA**) is the central authority for managing a database system.
  - The DBA's responsibilities include
    - granting privileges to users who need to use the system
    - classifying users and data in accordance with the policy of the organization
- The DBA is responsible for the overall security of the database system.

# Database Administrator (DBA)

- The DBA has a DBA account in the DBMS
  - Sometimes these are called a system or superuser account
  - These accounts provide powerful capabilities such as:
    - 1. Account creation
    - 2. Privilege granting
    - 3. Privilege revocation
    - 4. Security level assignment
  - Action 1 is access control, whereas 2 and 3 are discretionarym and 4 is used to control mandatory authorization

10

# Access Protection, User Accounts, and Database Audits(2)

- The database system must also keep **track of all operations** on the database that are applied by a certain user throughout **each login session**.
  - To keep a record of all updates applied to the database and of the particular user who applied each update, we can modify **system log**, which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.

11

## Access Protection, User Accounts, and Database Audits(3)

- If any tampering with the database is suspected, a **database audit** is performed
  - A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.
- A database log that is used mainly for security purposes is sometimes called an **audit trail**.

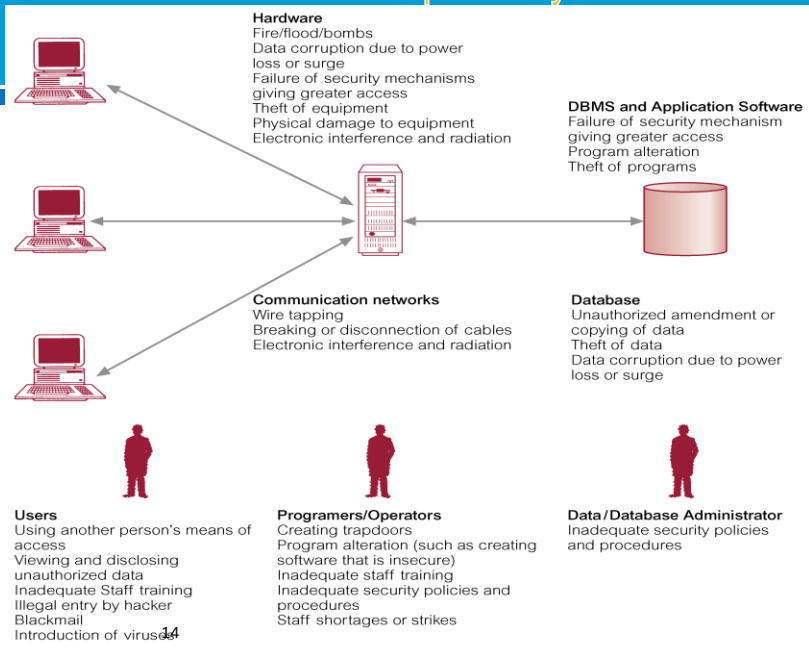
12

## Database Security Issue

- Threats to databases
  - Loss of **integrity**
  - Loss of **availability**
  - Loss of **confidentiality**

13

# Threats to Computer Systems




14

# Database Threats Quiz

Choose the best answer.

- Oracle, a major database vendor, sponsored a database security study which identified key security threats. In your view, which of the following is the biggest threat...

☐ External hackers

☐ Insiders and unauthorized users

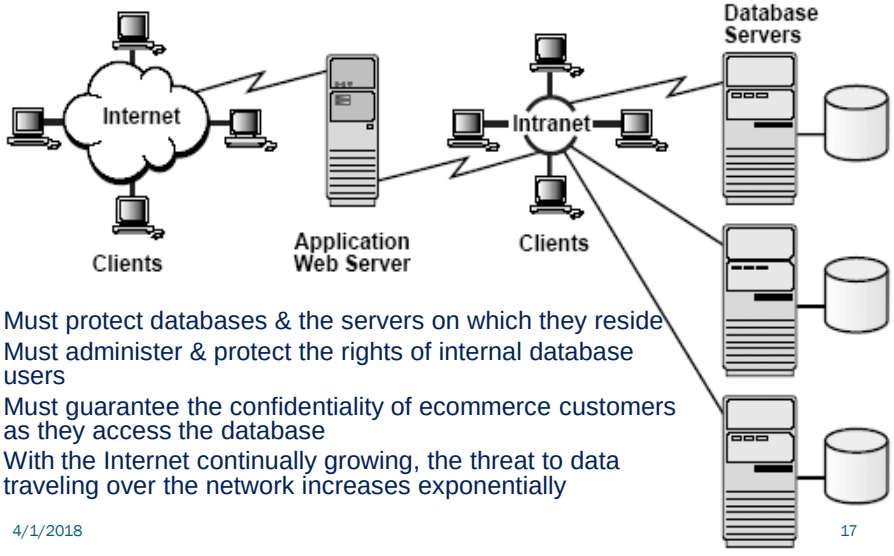
# Database Hacking Quiz

Mark all applicable answers.

Databases are attractive targets for hackers because...

- ☐ They store information such as SS#, DOB etc. that can be easily monetized
- ☐ They store information about lots of users
- ☐ Queries languages used to access data can be abused to gain unauthorized access.

# Scope of Data Security Needs




- Must protect databases & the servers on which they reside
- Must administer & protect the rights of internal database users
- Must guarantee the confidentiality of ecommerce customers as they access the database
- With the Internet continually growing, the threat to data traveling over the network increases exponentially

# Security Requirements

- Physical database integrity
- Logical database integrity
- Element integrity
- Auditability
- Access control
- User authentication
- Availability

# Security Requirements

- Physical database integrity
  - immunity to physical catastrophe, such as power failures, media failure
    - physical securing hardware, UPS
    - regular backups
- Logical database integrity
  - reconstruction Ability
    - maintain a log of transactions
    - replay log to restore the systems to a stable point

# Security Requirements

- Element integrity
  - integrity of specific database elements is their correctness or accuracy
    - field checks
      - allow only acceptable values
    - access controls
      - allow only authorized users to update elements
    - change log
      - used to undo changes made in error
    - referential Integrity (key integrity concerns)
    - two phase locking process
- Auditability
  - log read/write to database

# Security Requirements

- User Authentication
  - may be separate from OS
  - can be rigorous (hard)
- Availability
  - concurrent users
    - granularity of locking
  - reliability

# Security Requirements

- Access Control (similar to OS)
  - logical separation by user access privileges
  - more complicated than OS due to complexity of DB (granularity/inference/aggregation)
  - DAC, MAC, RBAC,
- Quiz: Database access control can be managed centrally by a few privileged users. This is an example of...

☐ DAC

☐ MAC

# Database Access Control System (DACS)

- a DACS: provides a specific capability that controls access to portions of the database (DAC or BRAC)
- A DBMS can support a range of administrative policies:
  - Centralized administration: A small number of privileged users may grant and revoke access rights.
  - Ownership-based administration: The owner (creator) of a table may grant and revoke access rights to the table.
  - Decentralized administration: In addition to granting and revoking access rights to a table, the owner of the table may grant and revoke authorization rights to other users, allowing them to grant and revoke access rights to the table.

# Database Access Control System (DACS)

- DACS: distinguishes different access right
- Access rights (create, insert, delete, update, read and write) to:
  - the entire database,
  - individual tables,
  - selected rows or columns within a table.
  - be determined based on the contents of a table entry.
- SQL provides 2 commands: GRANT and REVOKE

# Database Access Control Quiz

Choose the best answer.

Alice has SELECT access to a table and she can propagate this access to Bob when...

- ☐ Alice was granted this access with GRANT option
- ☐ She can always propagate an access she has

Cascading authorizations occur when an access is propagated multiple times and possibly by several users. Assume that Alice grants access to Bob who grants it further to Charlie. When Alice revokes access to Bob, should Charlie's access be also revoked?

☐ Yes ☐ No

# SQL Security Model

- SQL security model implements DAC based on
  - *users*: users of database - *user identity* checked during login process;
  - *actions*: including **`SELECT`**, **`UPDATE`**, **`DELETE`** and **`INSERT`**;
  - *objects*: tables (*base relations*), views, and columns (*attributes*) of tables and views
- Users can protect objects they own
  - when object created, a user is designated as 'owner' of object
  - owner may grant access to others
  - users other than owner have to be granted *privileges* to access object

# SQL Security Model

- Components of privilege are
  - `grantor, grantee, object, action, grantable`
  - privileges managed using **`GRANT`** and **`REVOKE`** operations
  - the right to grant privileges can be granted
- Issues with privilege management
  - each grant of privileges is to an individual or to "Public"
  - makes security administration in large organizations difficult
  - individual with multiple roles may have too many privileges for one of the roles
  - SQL3 is moving more to role based privileges

# SQL Security Model

- Inherent weakness of DAC
  - DAC allows subject to be written to any other object which can be written by that subject
  - trojan horses to copy information from one object to another
- Mandatory access controls (MAC)
  - no read up, no write down
  - traditional MAC implementations in RDBMS have focused solely on MLS
  - there have been three commercial MLS RDBMS offerings
    - trusted Oracle ,Informix OnLine/Secure, Sybase Secure SQL Server

# SQL Security Model

- Enforce MAC using security labels
  - assign security levels to all data
    - label associated with a row
  - assign a security clearance to each users
    - label associated with the user
  - DBMS enforces MAC
    - access to a row based upon
      - the label associated with that row and the label associated with the user accessing that row.

# Role-Based Access Control

- RBAC:
  - is a natural fit for database access control
  - use of roles in database security
  - provides a means of easing the administrative burden and improving security.
- A database RBAC facility needs to provide the capabilities:
  - Create and delete roles.
  - Define permissions for a role.
  - Assign and cancel assignment of users to roles.
- SQL supports 3 types of roles: server, database, user-defined.
  - The first two types of roles are referred to as fixed roles, are preconfigured for a system with specific access rights.
  - The administrator or user cannot add, delete, or modify fixed roles; it is only possible to add and remove users as members of a fixed role.

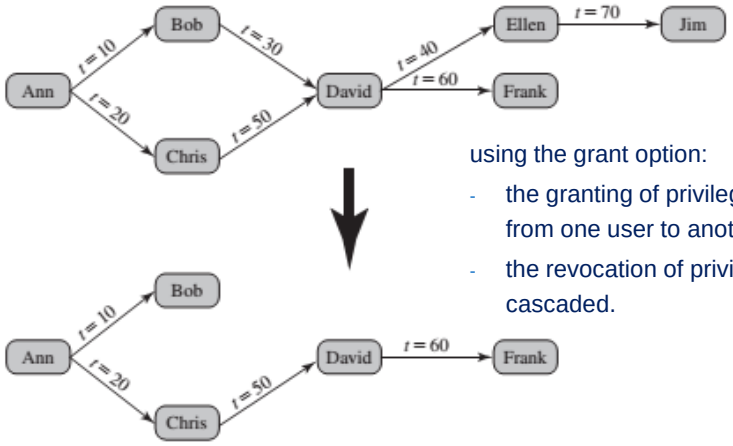


# SQL Security Model

- Authentication & identification mechanisms
  - CONNECT <user> USING<password>
  - DBMS may chose OS authentication
  - or its own authentication mechanism
    - Kerberose
    - PAM

# Cascading Authorizations

- The **grant option** enables an access right to cascade through a number of users




using the grant option:

- the granting of privileges cascades from one user to another,
- the revocation of privileges also cascaded.



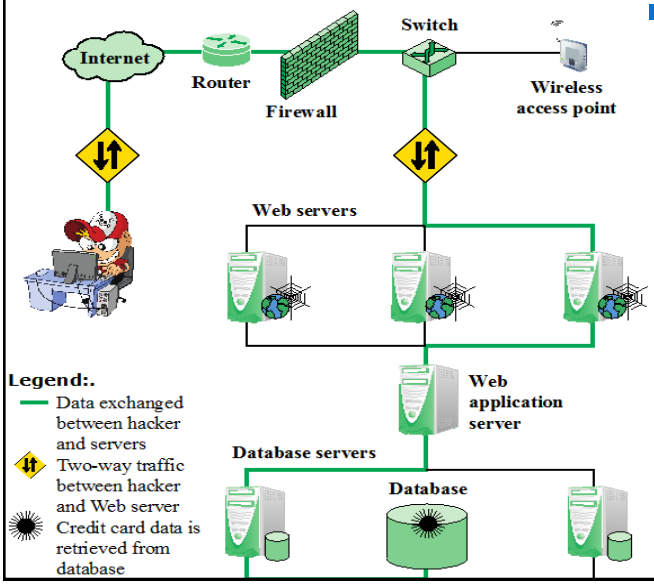
# Attacks on Databases

- SQL Injection
- Inference attacks

# SQL Injections

- Malicious SQL commands are sent to a database
- Can impact both
  - confidentiality (extraction of data) and
  - integrity (corruption of data)
- In a web application environment, typically a script takes user input and builds an SQL query
- Web application vulnerability can be used to craft an SQL injection
- SQL injection attack is one of the most prevalent and dangerous network-based security threats

# A typical SQL Injection




send malicious SQL commands to the database server
=> Results:

- dump database tables
- modify or delete data,
- execute arbitrary OS commands,
- Launch DoS attacks.

## Technique of SQL Injections

The SQLi attack typically works:

- early terminating a text string
- appending a new command.
- terminates the injected string with a comment mark "--".

**Example:**

```
Var Shipcity;
Shipcity = Request.form ("Shipcity");
Var sql = "select * from OrdersTable
where
Shipcity = '" + Shipcity + "'";
```

a user will enter the name of a city. Ex, REDMOND,

- Script generates:

SELECT * FROM OrdersTable Where Shipcity = 'Redmond'.

## SQL Injection Example

- What if user enters:

Redmond' ; DROP table OrdersTable--

- In this case, script is generated:

SELECT * FROM OrdersTable WHERE Shipcity = 'Redmond' ;
DROP OrdersTable

⇒ Server will:

- select all records in OrdersTable where ShipCity is Redmond.
- Then, it executes the DROP request

- Malicious user is able to inject code to delete the table
- Many other code injection examples exist

# SQLi Attack Avenues

- User input: In this case, attackers inject SQL commands by providing suitably crafted user input.
- Server variables: variables are logged to a database without sanitization, this could create an SQL injection vulnerability.
- Second-order injection: a malicious user could rely on data already present in the system or database to trigger an SQL injection attack
- Cookies: an attacker could alter cookies when the application server builds an SQL query based on the cookie's content, the structure and function of the query is modified.
- Physical user input: could be scanned using optical character recognition and passed to a database management system.

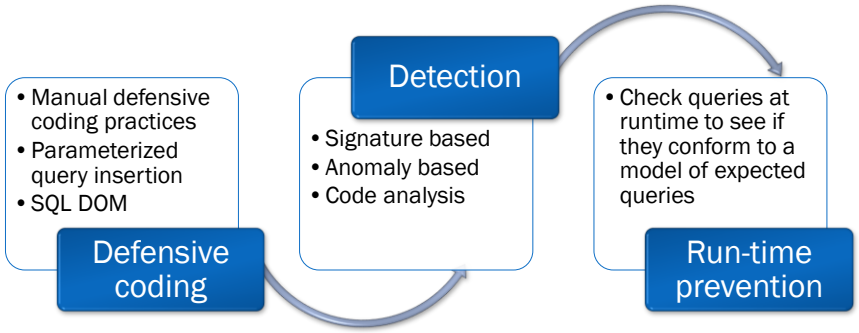


# SQLi attacks

- SQL Injections can do more harm than just by passing the login algorithms. Some of the attacks include
  - Deleting data
  - Updating data
  - Inserting data
  - Executing commands on the server that can download and install malicious programs such as Trojans
  - Exporting valuable data such as credit card details, email, and passwords to the attacker's remote server
  - Getting user login details etc

# SQL Injection Defenses

- An integrated set of techniques is necessary:






# SQL Login Quiz

Mark all applicable answers.

A web application script uses the following code to generate a query:

**Query = "SELECT accounts FROM users WHERE login = ' " + login + " ' AND pass = ' " + password + " ' AND pin = " + pin; The various arguments are read from a form to generate Query.**

This query is executed to get a user's account information when the following is provided correctly...

☐ Login name ☐ Password ☐ PIN

## SQL Login Quiz #2

Choose the best answer.

**Query = "SELECT accounts FROM users WHERE login = ' " + login + " ' AND pass = ' " + password + " ' AND pin = " + pin; The various arguments are read from a form to generate Query.**

If a user types "'or 1 = 1 --" for login in the above query...

- ☐ Query will fail because the provided login is not a correct user
- ☐ An injection attack will result in all users' account data being returned

## Inference Attacks on Databases

- Inference attacks:
  - relates to database security
  - is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- Problem:
  - the combination of a number of data items is more sensitive than the individual items,
  - the combination of data items can be used to infer data of a higher sensitivity

# Inference

**Anonymous medical data:**

| SSN | Name | Race | DOB | Sex | Zip | Marital | Heath |
|---|---|---|---|---|---|---|---|
| | | Asian | 09/07/64 | F | 22030 | Married | Obesity |
| | | Black | 05/14/61 | M | 22030 | Married | Obesity |
| | | White | 05/08/61 | M | 22030 | Married | Chest pain |
| | | White | 09/15/61 | F | 22031 | Widow | Aids |

**Public available voter list:**

| Name | Address | City | Zip | DOB | Sex | Party |
|---|---|---|---|---|---|---|
| …. | …. | …. | …. | …. | …. | …. |
| Sue Carlson | 900 Market St. | Fairfax | 22031 | 09/15/61 | F | Democrat |

**Sue Carlson has Aids!**



# Inference

- Types of attack
  - **direct attack**: aggregate computed over a small sample so individual data items leaked
  - **indirect attack**: combines several aggregates;
  - **tracker attack**: type of indirect attack (very effective)
  - linear system vulnerability: takes tracker attacks further, using algebraic relations between query sets to construct equations yielding desired information

# Inference, ex

| NAME | SEX | RACE | AID | FINES | DRUGS | DORM |
|---|---|---|---|---|---|---|
| Adams | M | C | 5000 | 45 | 1 | Holmes |
| Bailey | M | B | 0 | 0 | 0 | Grey |
| Chin | F | A | 3000 | 20 | 0 | West |
| Dewitt | M | B | 1000 | 35 | 3 | Grey |
| Earhart | F | C | 2000 | 95 | 1 | Holmes |
| Fein | F | C | 1000 | 15 | 0 | West |
| Groff | M | C | 4000 | 0 | 3 | West |
| Hill | F | B | 5000 | 10 | 2 | Holmes |
| Koch | F | C | 0 | 0 | 1 | West |
| Liu | F | A | 0 | 10 | 2 | Grey |
| Majors | M | C | 2000 | 0 | 2 | Grey |

# Inference - Direct Attack

- Direct Attack
  - determine values of sensitive fields by seeking them directly with queries that yield few records
  - request LIST which is a union of 3 sets

    LIST NAME where (SEX =M $\wedge$ DRUGS = 1) $\vee$
    (SEX $\neq$ M $\wedge$ SEX $\neq$F) $\vee$ (DORM = Ayres)
    - No dorm named Ayres , Sex either M or F
  - "*n* items over *k* percent" rule helps prevent attack

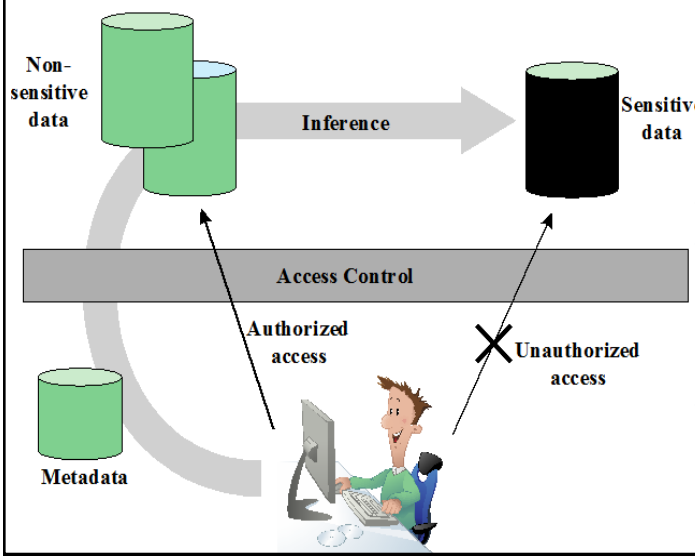
# Inference - Indirect attack

## Indirect attack: combines several aggregates

| Sums of Financial Aid by Dorm and Sex | | | | |
|---|---|---|---|---|
| | **Holmes** | **Grey** | **West** | **Total** |
| M | 5000 | 3000 | 4000 | 12000 |
| F | 7000 | 0 | 4000 | 11000 |
| Total | 12000 | 3000 | 8000 | 23000 |
| **Students by Dorm and Sex** | | | | |
| | **Holmes** | **Grey** | **West** | **Total** |
| M | 1 | 3 | 1 | 5 |
| F | 2 | 1 | 3 | 6 |
| Total | 3 | 4 | 4 | 11 |

- 1 Male in Holmes receives 5000
- 1 Female in Grey received no aid
  - request a list of names by dorm (non sensitive)

# Indirect Information Access via Inference Channel



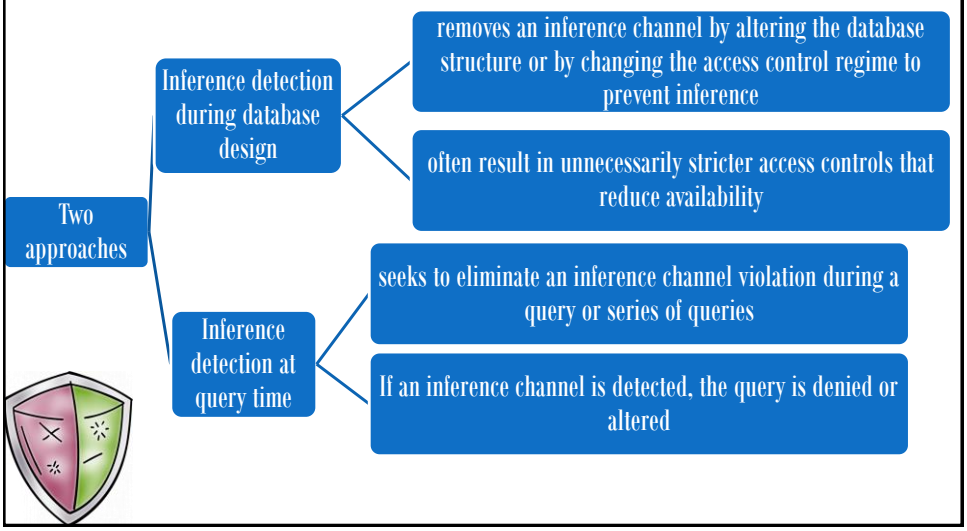

2 inference techniques:

- analyzing functional dependencies between attributes within a table or across tables,
- merging views with the same constraints.

# Inference - tracker attack

- Often databases protected against delivering small response sets to queries
- Trackers can identify unique value
  - request (n) and (n-1) values
  - given *n* and *n* – 1, we can easily compute the desired single element

# Defenses Against Inference Attacks



## SQL Inference Attack Quiz

Choose the best answer.

The database that stores student exam scores allows queries that return average score for students coming from various states. Can this lead to an inference attack in this system?

- ☐ Yes, depending on how many students come from each state
- ☐ No, it is not possible

## SQL Inference Attack Quiz #2

Choose the best answer.

Assume in (1), the data in the database is de-identified by removing student id (and other information such as names). Furthermore, the field that has the state of the student is generalized by replacing it with the US region (e.g., Midwest). The generalization ensures that there are at least two students from each region. Are inference attacks still possible?

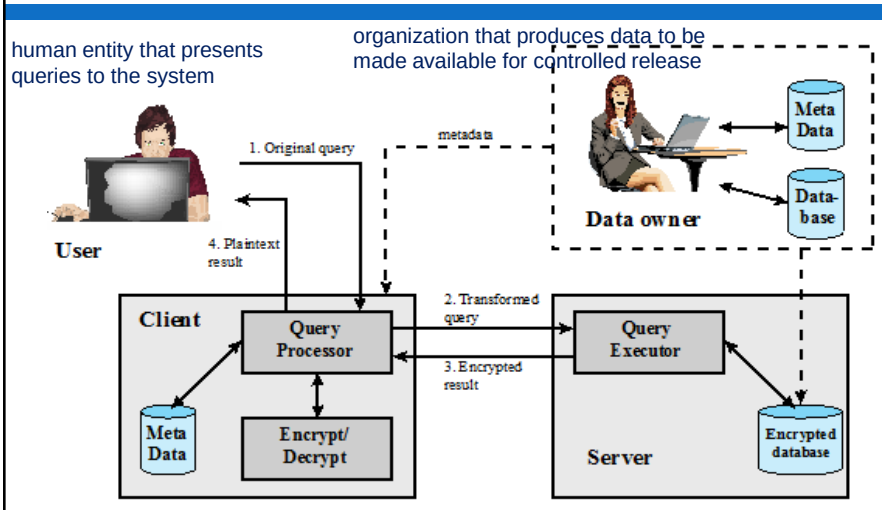
☐ Yes ☐ No

# Database encryption

- The database is protected by multiple layers of security:
  - Firewalls
  - Authentication mechanisms
  - General access control systems
  - Database access control systems.
- Database encryption is warranted and often implemented for particularly sensitive data
- There are two disadvantages to database encryption:
  - **Key management:** Authorized users must have access to the decryption key for the data. Providing secure keys to selected parts of the database to authorized users and applications is a complex task.
  - **Inflexibility:** When part or all of the database is encrypted, it becomes more difficult to perform record searching.



# A Database Encryption Scheme



# Summary

- Used to **store lots of sensitive data** that can be accessed via programs (queries)
- Access control must be **based on operations allowed by databases**
- New attacks on databases arise due to their unique characteristics
- Defenses **must address such attacks**

# Q & A