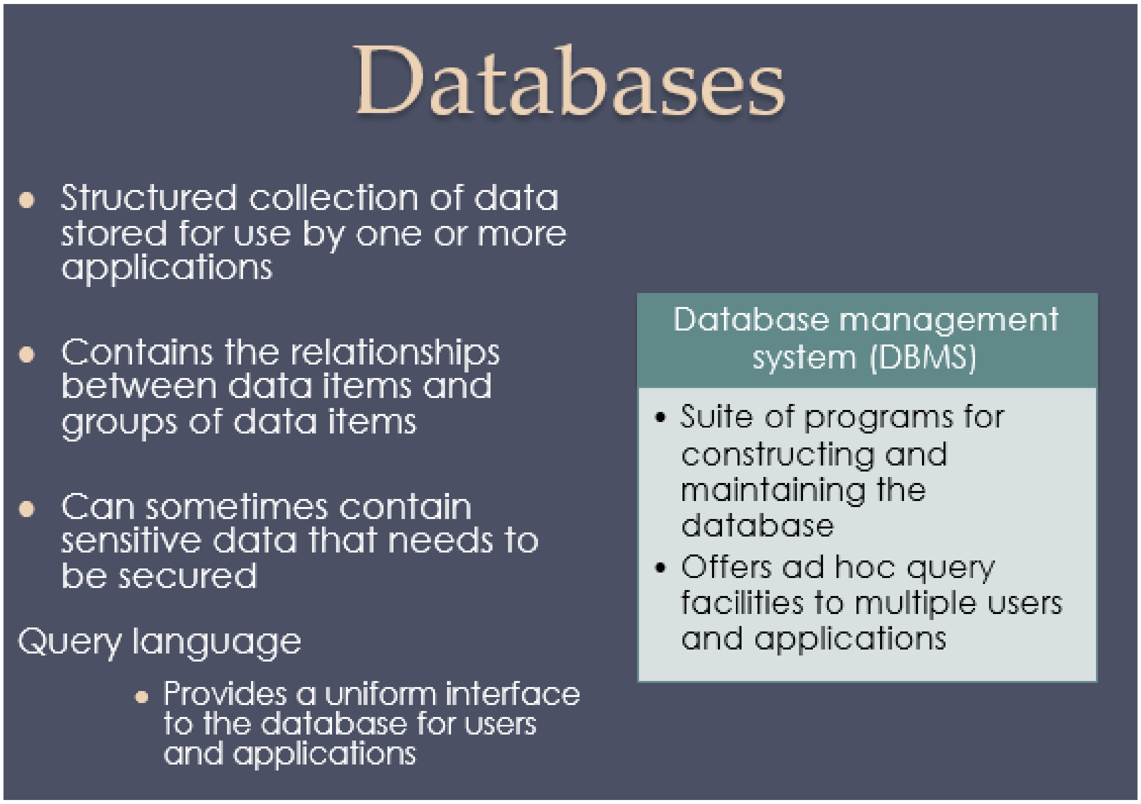
Chapter 5(5.1 to 5.7)

Database Security:

Reasons why database security has not kept pace with the increased reliance on databases:

1. Databases have a sophisticated interaction protocol called the Structured Query Language (SQL), which is complex.
2. There is a dramatic imbalance between the complexity of modern database management systems (DBMS) and the security techniques used to protect these critical systems.
3. The typical organization lacks full-time database security personnel. The result is a mismatch between requirements and capabilities
4. Most enterprise environments consist of a heterogeneous mixture of database platforms (Oracle, IBM DB2 and Informix, Microsoft, Sybase, etc.), enterprise platforms (Oracle E-Business Suite, PeopleSoft, SAP, Siebel, etc.), and OS platforms (UNIX, Linux, z/OS, and Windows, etc.). This creates an additional complexity hurdle for security personnel.
5. Increase reliance on cloud technology to host part of or all of the corporate database.



Databases

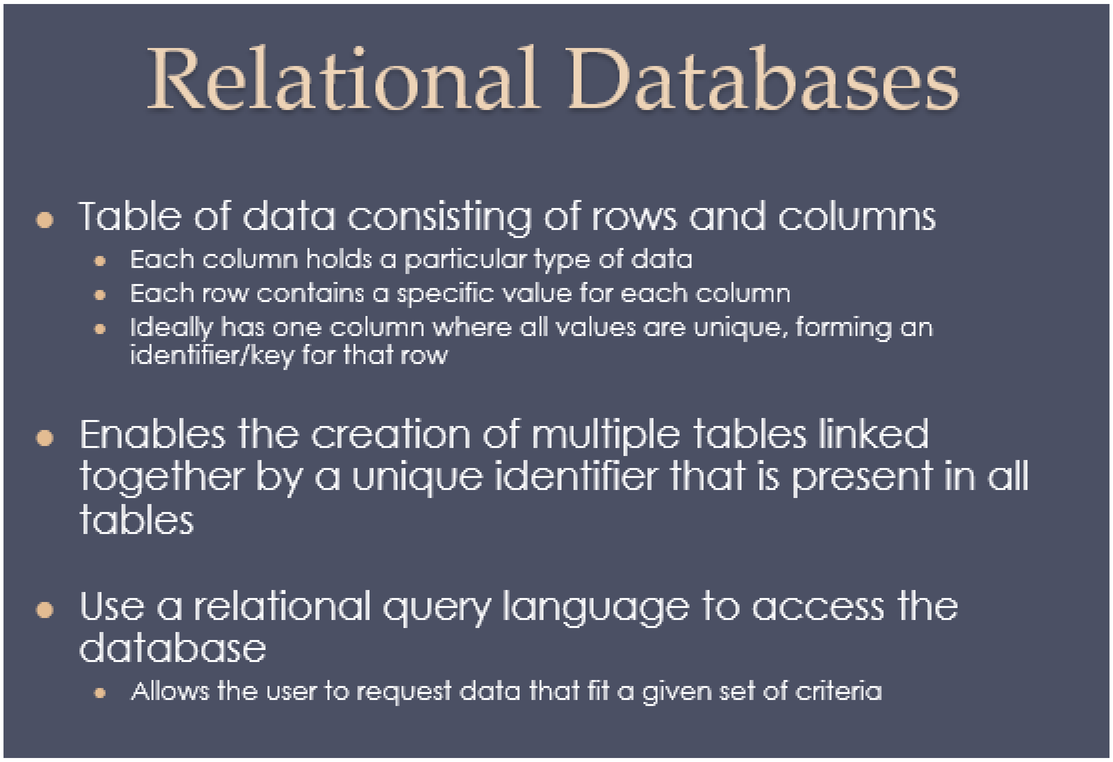
- Structured collection of data stored for use by one or more applications
- Contains the relationships between data items and groups of data items
- Can sometimes contain sensitive data that needs to be secured

Query language

- Provides a uniform interface to the database for users and applications

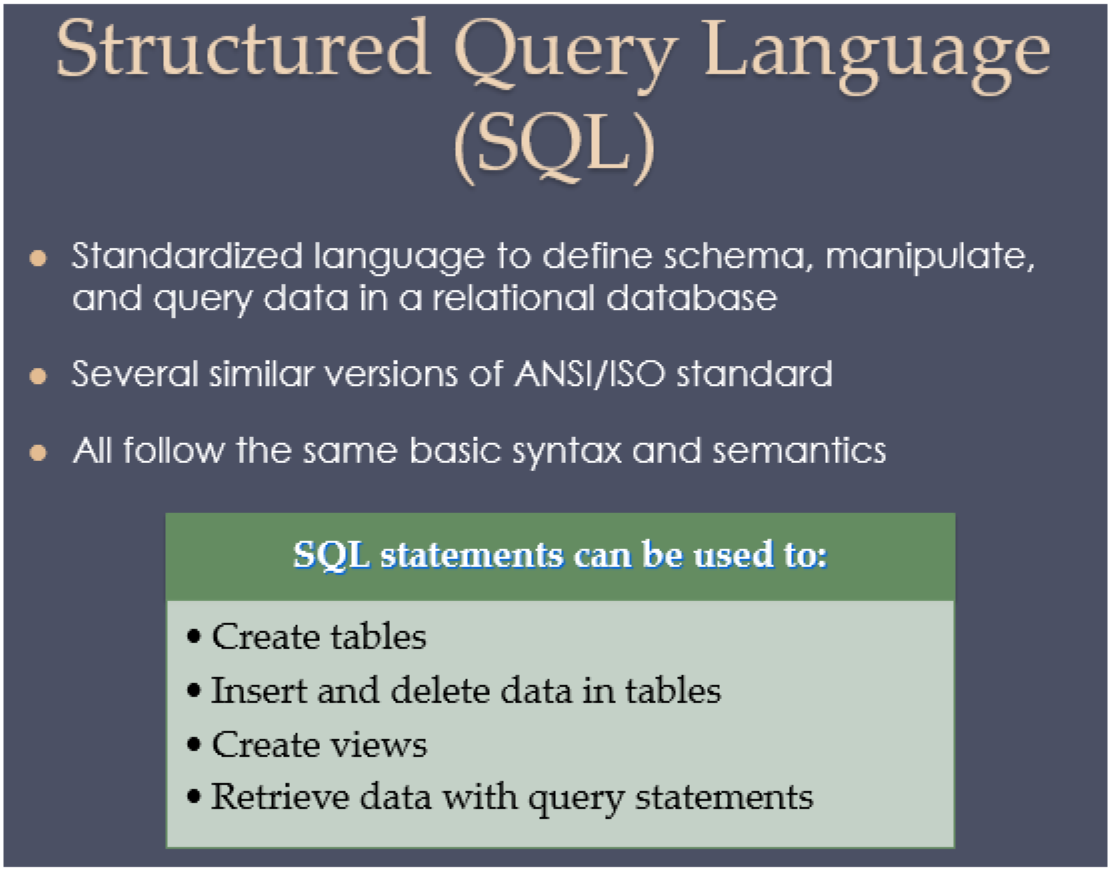
Database management system (DBMS)

- Suite of programs for constructing and maintaining the database
- Offers ad hoc query facilities to multiple users and applications



Relational Databases

- Table of data consisting of rows and columns
  - Each column holds a particular type of data
  - Each row contains a specific value for each column
  - Ideally has one column where all values are unique, forming an identifier/key for that row
- Enables the creation of multiple tables linked together by a unique identifier that is present in all tables
- Use a relational query language to access the database
  - Allows the user to request data that fit a given set of criteria

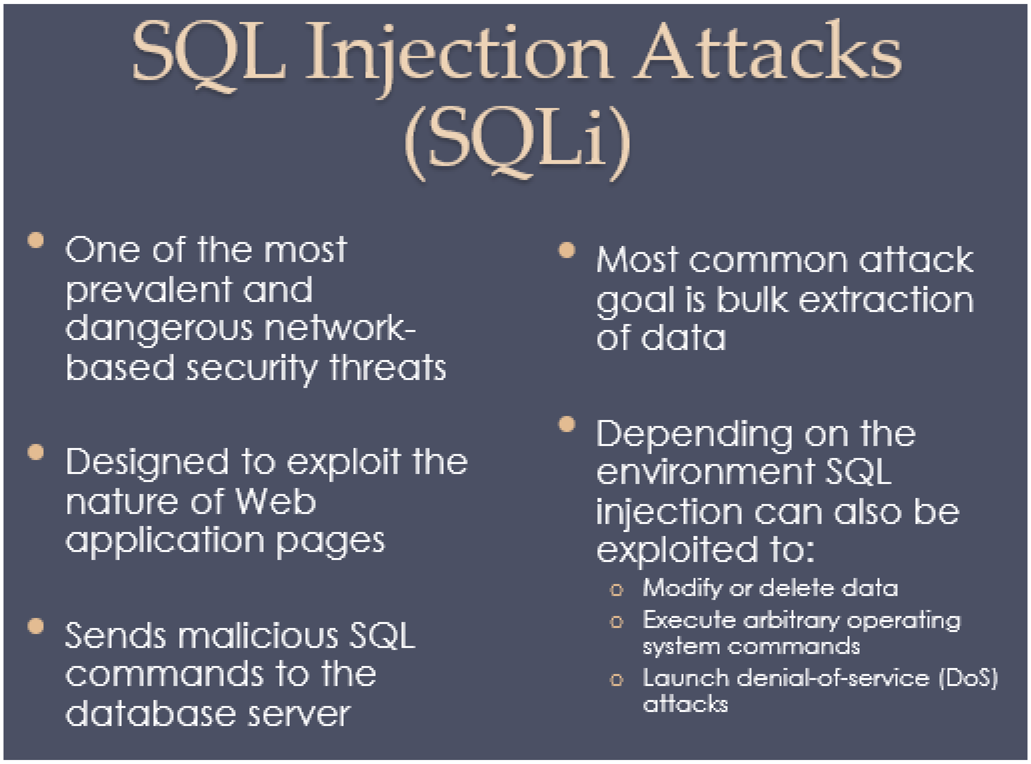


Structured Query Language (SQL)

- Standardized language to define schema, manipulate, and query data in a relational database
- Several similar versions of ANSI/ISO standard
- All follow the same basic syntax and semantics

SQL statements can be used to:

- Create tables
- Insert and delete data in tables
- Create views
- Retrieve data with query statements



SQL Injection Attacks (SQLi)

- One of the most prevalent and dangerous network-based security threats
- Designed to exploit the nature of Web application pages
- Sends malicious SQL commands to the database server
- Most common attack goal is bulk extraction of data
- Depending on the environment SQL injection can also be exploited to:
  - Modify or delete data
  - Execute arbitrary operating system commands
  - Launch denial-of-service (DoS) attacks

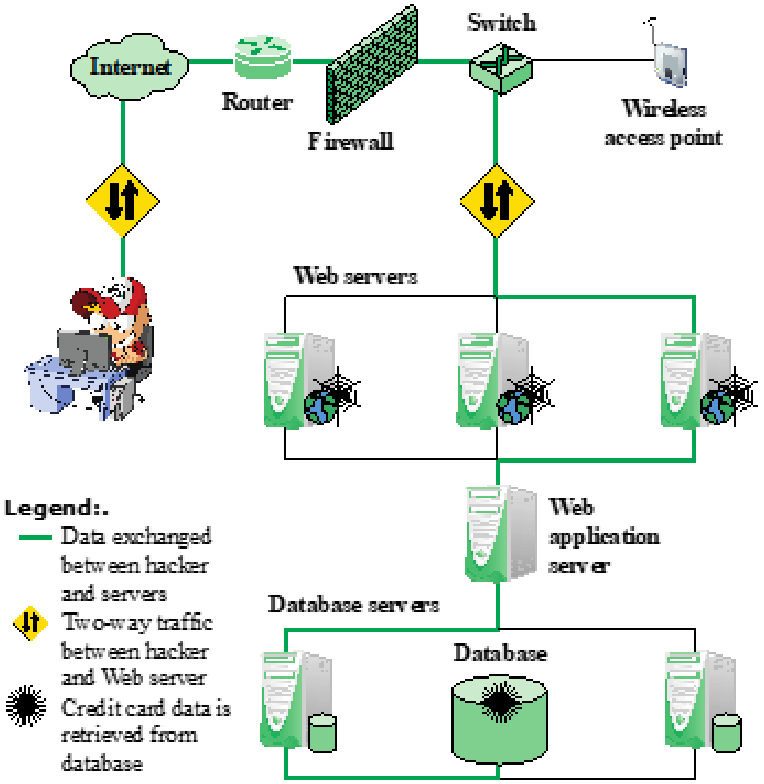

Internet
Router
Firewall
Switch
Wireless access point
Web servers
Web application server
Database servers
Database
Legend:.
Data exchanged between hacker and servers
Two-way traffic between hacker and Web server
Credit card data is retrieved from database

Figure 5.5 Typical SQL Injection Attack

1. Hacker finds a vulnerability in a custom Web application and injects an SQL command to a database by sending the command to the Web server. The command is injected into traffic that will be accepted by the firewall.

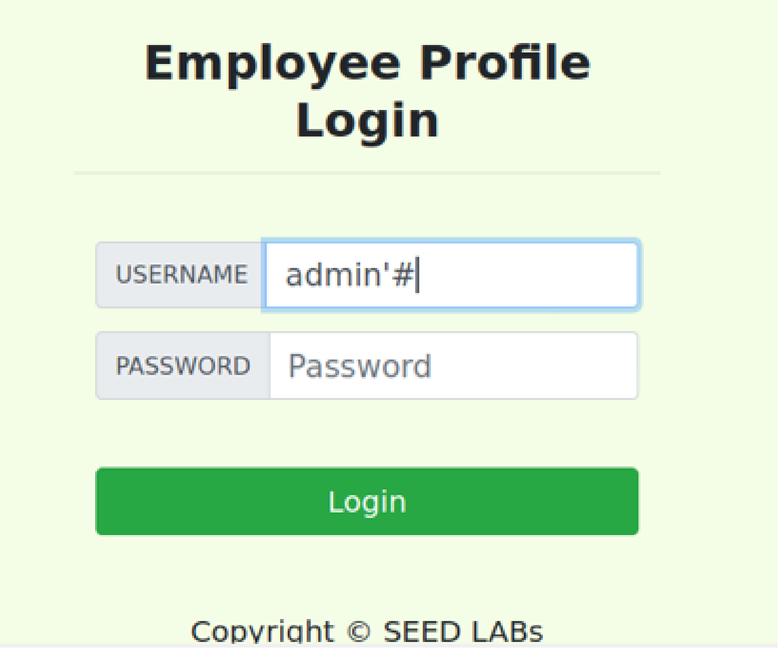
2. The Web server receives the malicious code and sends it to the Web application server.

3. The Web application server receives the malicious code from the Web server and sends it to the database server.

4. The database server executes the malicious code on the database. The database returns data from credit cards table.

5. The Web application server dynamically generates a page with data including credit card details from the database.

6. The Web server sends the credit card details to the hacker.



USERNAME admin'#

By entering the username as “admin’#”, the effective SQL query generated at the backend is as:

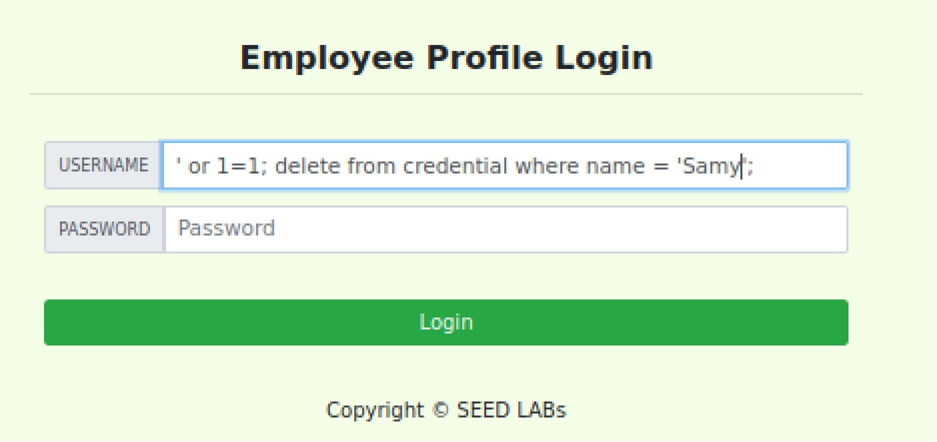
SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE

name= ’admin’#’ and Password=’’

Where the highlighted part is commented from the query. And thus the query returns all the results from the database and is displayed on the interface as follows:

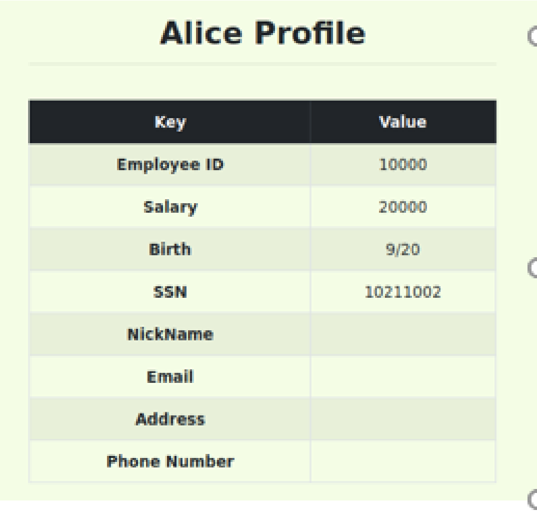
Task 2.3: Append a new SQL statement:

Username = ' or 1=1; delete from credential where name = 'Samy';



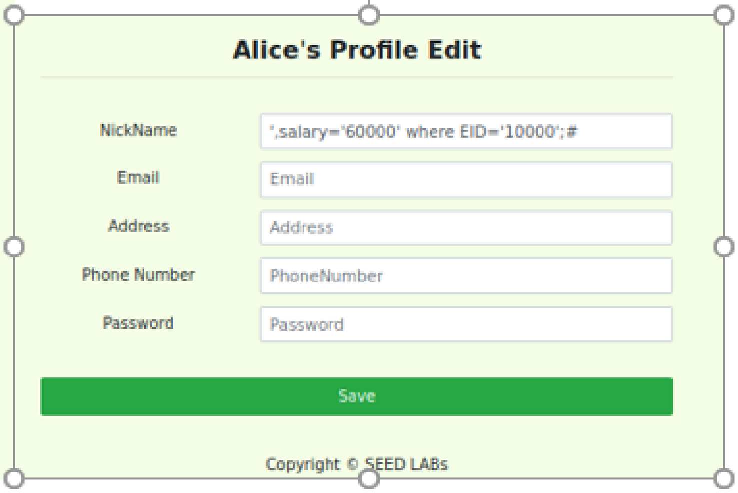
USERNAME ' or 1=1; delete from credential where name = 'Samy';

Salary is 20000, lets suppose Alice wants to triple it to 60000



Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	



Alice's Profile Edit

NickName	',salary='60000' where EID='10000';#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Copyright © SEED LABs

Command: ',salary='60000' where EID='10000';#

Query on Backend:

update table user_details set NickName=’’,salary =’60000’ where EID=’10000’;# -- the rest of the fields are commented

SQLi Attack Avenues:

- **User Input:** Attackers inject SQL commands by providing suitable crafted user input
- **Server variables**: Attackers can forge the values that are placed in HTTP and network headers and exploit this vulnerability by placing data directly into the headers
- **Second-order injection**: type of attack in which the attacker is able to inject malicious code into a website's database, and then subsequently retrieve and execute that code at a later time.
- **Cookies**: An attacker could alter cookies such that 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**: Applying user input that constructs an attack outside the realm of web requests

Different types of attacks:

Three main categories: inband, inferential, and out-of-band.

1. **Inband Attacks:** uses the same communication channel for injecting SQL code and retrieving results.
    1. **Tautology:** This form of attack injects code in one or more conditional statements so that they always evaluate to true. **Username = ‘,1=1;**
    2. **End-of-line comment**: After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments. An example would be to add”- -” or “#” after inputs so that remaining queries are not treated as executable code, but comments.
    3. **Piggybacked queries**: The attacker adds additional queries beyond the intended query, piggy-backing the attack on top of a legitimate request.

2. **Inferential attack:** there is no actual transfer of data, but the attacker is able to reconstruct the information by sending particular requests and observing the resulting behavior of the Website/database server
    1. **Illegal/logically incorrect queries**: This attack lets an attacker gather important information about the type and structure of the backend database of a Web application. The vulnerability leveraged by this attack is that the default error page returned by application servers is often overly descriptive.
    2. **Blind SQL injection**: Blind SQL injection allows attackers to infer the data present in a database system even when the system is sufficiently secure to not display any erroneous information back to the attacker. The attacker asks the server true/false questions. If the injected statement evaluates to true, the site continues to function normally. If the statement evaluates to false, although there is no descriptive error message, the page differs significantly from the normally functioning page.
3. **Out-of-band attack**: An out-of-band attack is a type of SQL injection attack in which the attacker sends data to the database through a channel that is separate from the main communication channel. Example of an out-of-band attack would be an attacker using a phone or email to communicate with the database, sending commands and receiving data through a channel that is separate from the main communication channel.

SQL Injection Countermeasures:

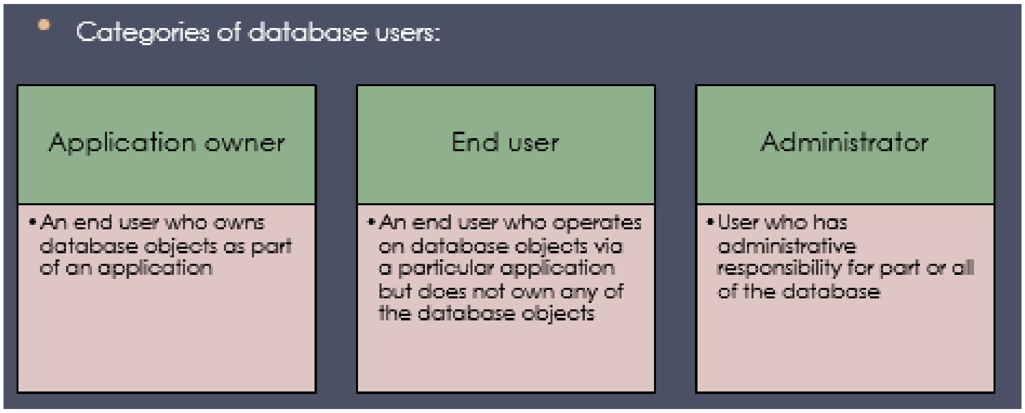
1. **Defensive Coding:**
    a. **Manual defensive coding practices:** input validation. Input type checking. Pattern Matching before sending query to database.
    b. **Parameterized query insertion**: specify the structure of an SQL query, and pass the value parameters to it separately.
    c. **SQL DOM**: set of classes that enables automated data type validation and escaping. Use of API to apply best coding practices.
2. **Detection:**
    a. **Signature based**: This technique attempts to match specific attack patterns. Such an approach must be constantly updated and may not work against self-modifying attacks.
    b. **Anomaly based**: This approach attempts to define normal behavior and then detect behavior patterns outside the normal range.
    c. **Code analysis**: Code analysis techniques involve the use of a test suite to detect SQLi vulnerabilities. The test suite is designed to generate a wide range of SQLi attacks and assess the response of the system.
3. **Run-time prevention:** check queries at runtime to see if they conform to a model of expected queries.

Database Access Control:

Determines whether a user may have access to the database as a whole or a portion of it. What access rights the user has(create, insert, delete, update, read, and write). Access rights can be to the entire database, to individual tables, or to selected rows or columns within a table

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.



- Categories of database users:

Application owner

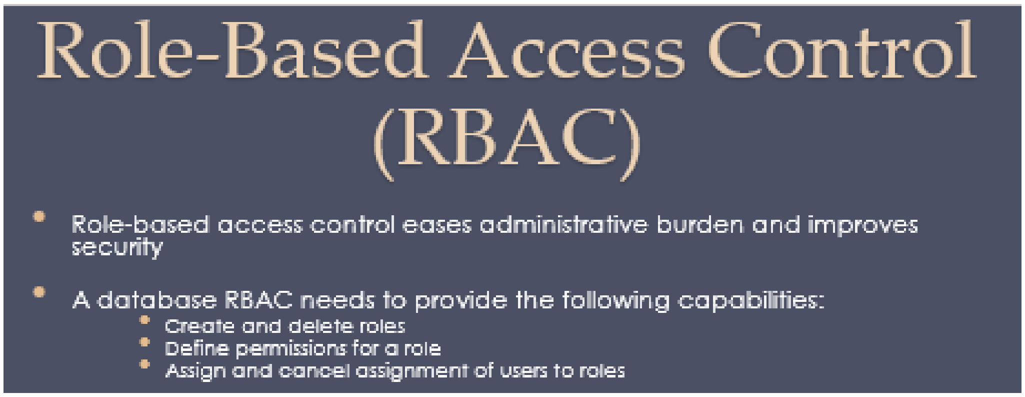
- An end user who owns database objects as part of an application

End user

- An end user who operates on database objects via a particular application but does not own any of the database objects

Administrator

- User who has administrative responsibility for part or all of the database



Role-Based Access Control (RBAC)

- Role-based access control eases administrative burden and improves security
- A database RBAC needs to provide the following capabilities:
  - Create and delete roles
  - Define permissions for a role
  - Assign and cancel assignment of users to roles

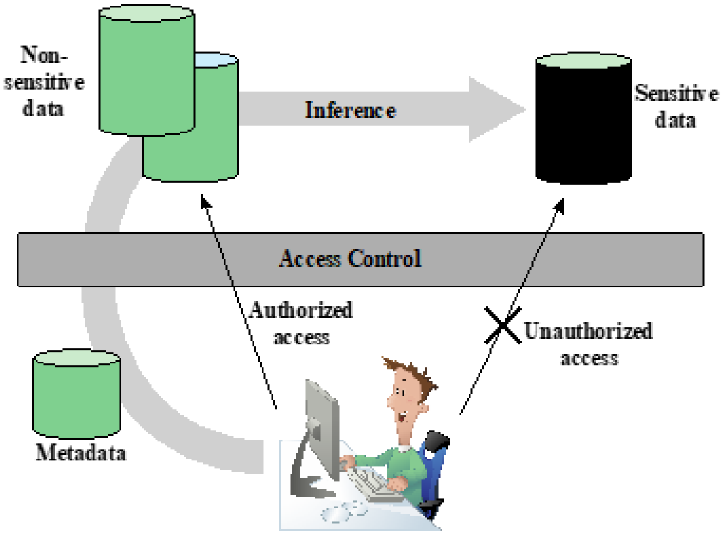
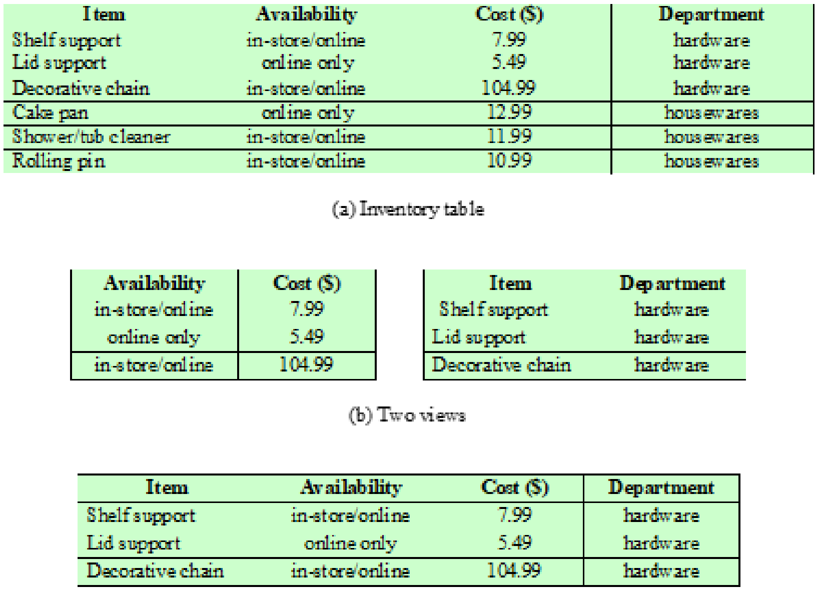


Figure 5.7 Indirect Information Access Via Inference Channel

The inference problem arises when a combination of data items can be used to infer data of a higher sensitivity. The attacker may make use of non-sensitive data as well as metadata to deduce information not otherwise available to a particular user.

Two inference techniques can be used to derive additional information:

1. Analyzing functional dependencies between attributes within a table or across tables.
2. Merging views with the same constraints.



Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

Availability	Cost ($)
in-store/online	7.99
online only	5.49
in-store/online	104.99

Item	Department
Shelf support	hardware
Lid support	hardware
Decorative chain	hardware

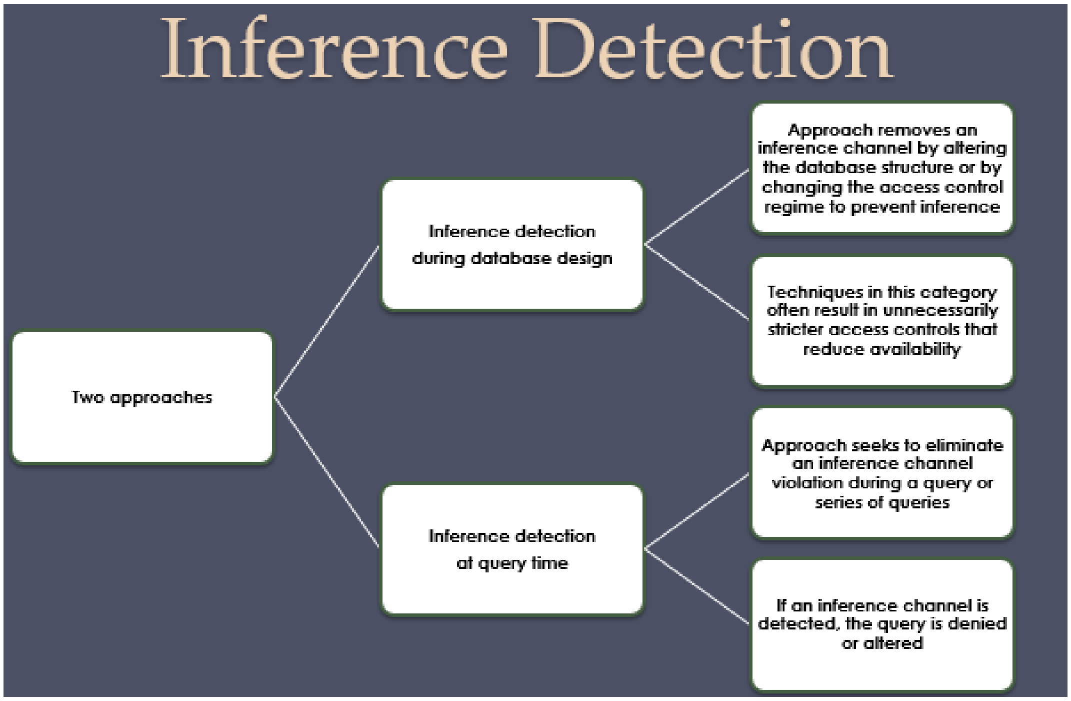
(b) Two views

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

(c) Table derived from combining query answers

Figure 5.8 Inference Example

Users of these views are not authorized to access the relationship between Item and Cost. A user who has access to either or both views cannot infer the relationship by functional dependencies. That is, there is not a functional relationship between Item and Cost such that knowing Item and perhaps other information is sufficient to deduce Cost. However, suppose the two views are created with the access constraint that Item and Cost cannot be accessed together. A user who knows the structure of the Inventory table and who knows that the view tables maintain the same row order as the Inventory table is then able to merge the two views to construct the table shown in Figure 5.8c. This violates the access control policy that the relationship of attributes Item and Cost must not be disclosed.



Inference Detection

Two approaches

Inference detection during database design

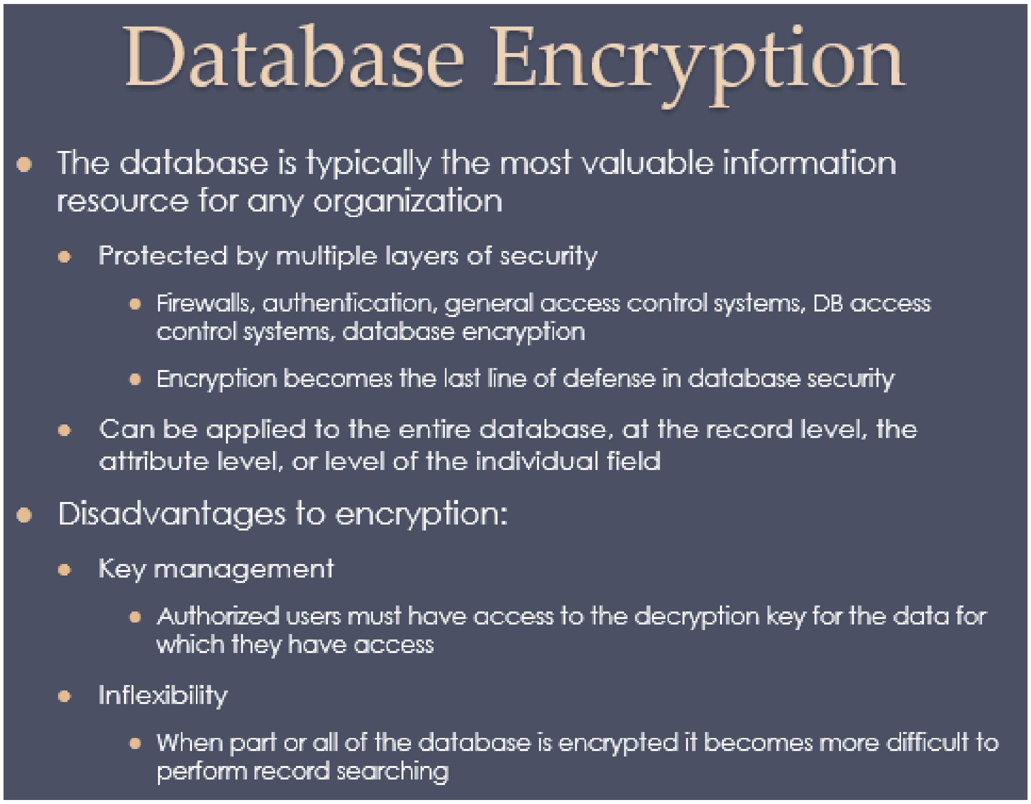
Approach removes an inference channel by altering the database structure or by changing the access control regime to prevent inference

Techniques in this category often result in unnecessarily stricter access controls that reduce availability

Inference detection at query time

Approach seeks to eliminate an inference channel violation during a query or series of queries

If an inference channel is detected, the query is denied or altered



Database Encryption

- The database is typically the most valuable information resource for any organization
  - Protected by multiple layers of security
    - Firewalls, authentication, general access control systems, DB access control systems, database encryption
    - Encryption becomes the last line of defense in database security
  - Can be applied to the entire database, at the record level, the attribute level, or level of the individual field
- Disadvantages to encryption:
  - Key management
    - Authorized users must have access to the decryption key for the data for which they have access
  - Inflexibility
    - When part or all of the database is encrypted it becomes more difficult to perform record searching

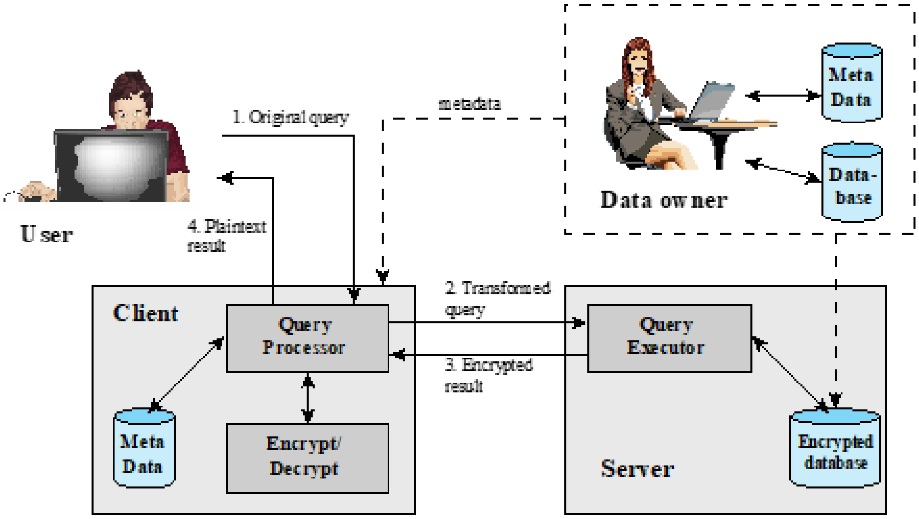
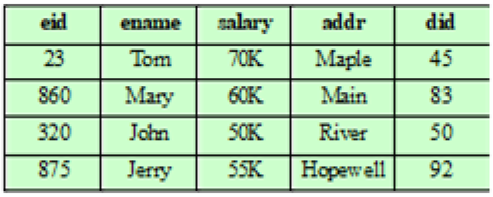


Figure 5.9 A Database Encryption Scheme

1. The user issues an SQL query for fields from one or more records with a specific value of the primary key.
2. The query processor at the client encrypts the primary key, modifies the SQL query accordingly, and transmits the query to the server.
3. The server processes the query using the encrypted value of the primary key and returns the appropriate record or records.
4. The query processor decrypts the data and returns the results.

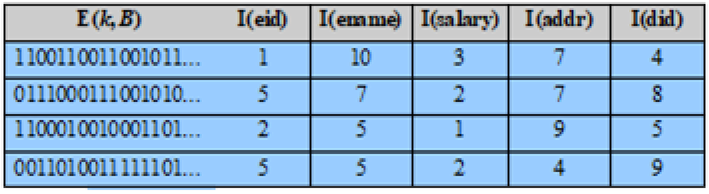
Table 5.3 Encrypted Database Example

(a) Employee Table



eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes



E(*k*, *B*)	I(eid)	I(ename)	I(salary)	I(addr)	I(did)
1100110011001011...	1	10	3	7	4
0111000111001010...	5	7	2	7	8
1100010010001101...	2	5	1	9	5
0011010011111101...	5	5	2	4	9

The values in the first column represent the encrypted values for each row. The actual values depend on the encryption algorithm and the encryption key. The remaining columns show index values for the corresponding attribute values. The mapping functions between attribute values and index values constitute metadata that are stored at the client and data owner locations but not at the server.