Database Security

Introduction to Computer Security

Naercio Magaia and Imran Khan

Contents

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

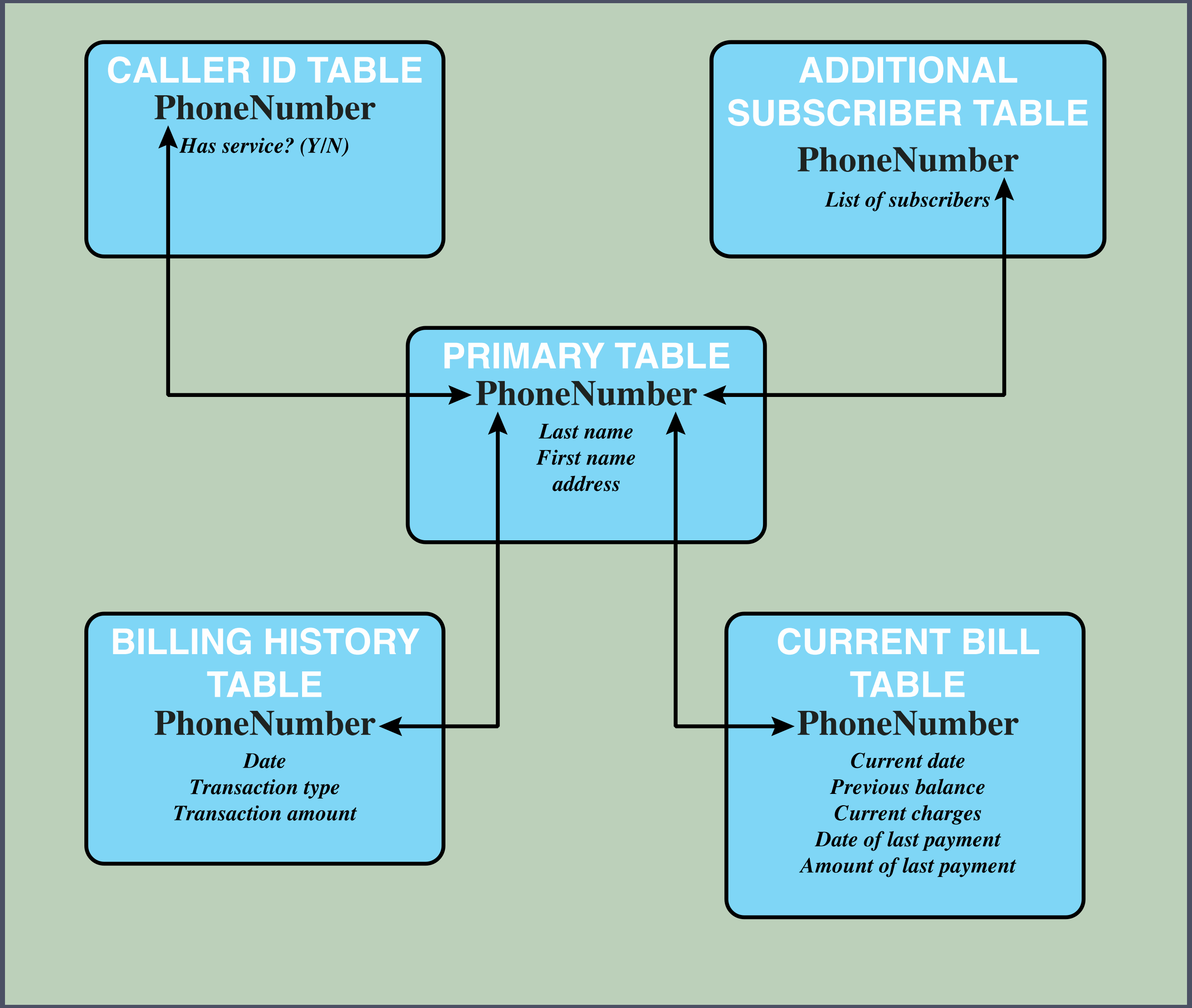
Database management system (DBMS)

- **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

Example Relational Databases Model



Relational Database Elements

- Relation
  - Table/file
- Tuple
  - Row/record
- Attribute
  - Column/field

Primary key

- Uniquely identifies a row
- Consists of one or more column names

Foreign key

- Links one table to attributes in another

View/virtual table

- Result of a query that returns selected rows and columns from one or more tables
- Views are often used for security purposes

Basic Terminology for Relational Databases

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

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 Example

CREATE VIEW newtable (Dname, Ename, Eid, Ephone) AS SELECT D.Dname, E.Ename, E.Eid, E.Ephone

FROM Department D, Employee E WHERE E.Did = D.Did

Department Table

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

$\underbrace{\text{Did}}_{\text{primary key}}$

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

$\underbrace{\text{Did}}_{\text{foreign key}}$ $\underbrace{\text{Eid}}_{\text{primary key}}$

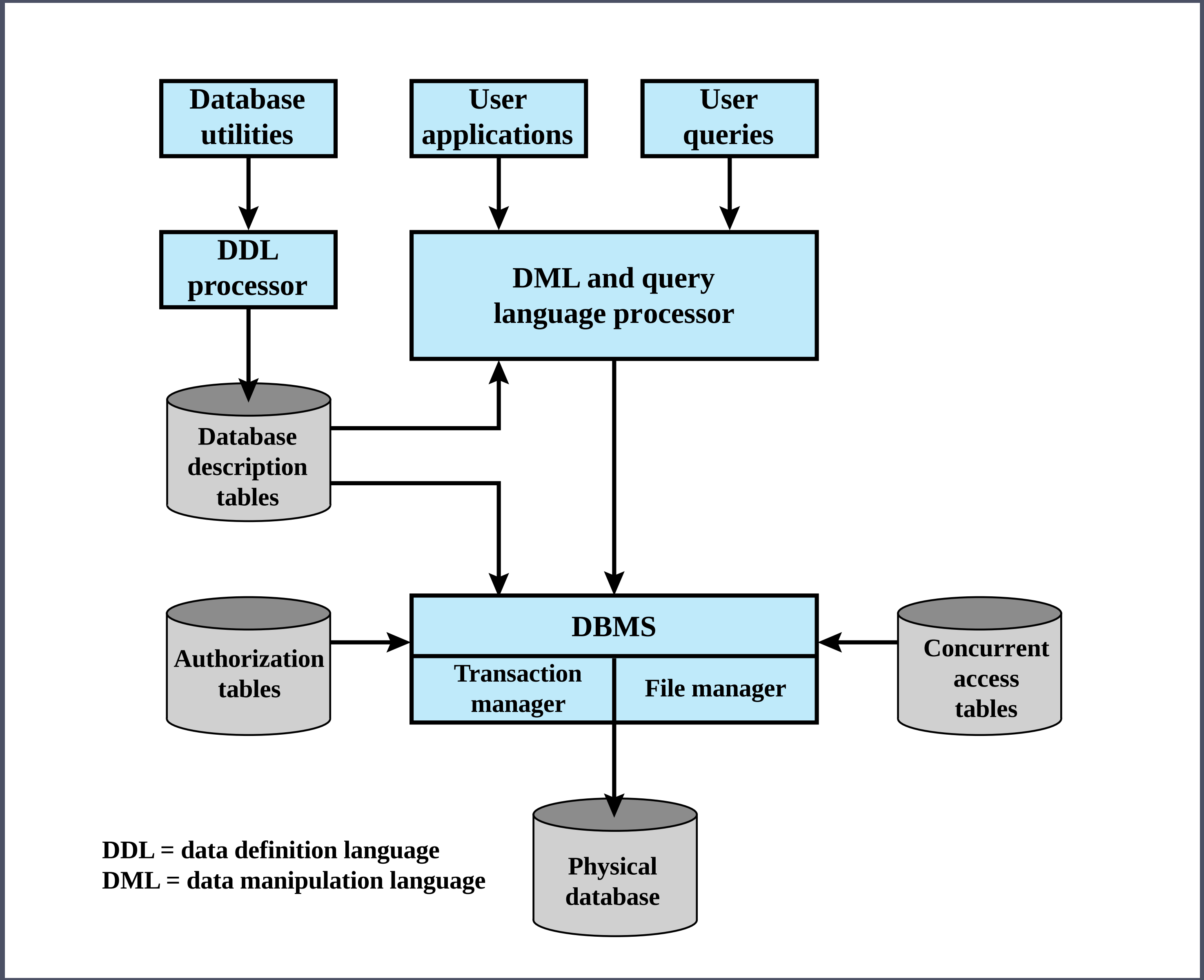
(a) Two tables in a relational database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Figure 5.4 Relational Database Example

DBMS Architecture



Database Security

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

There is a dramatic imbalance between the **complexity of modern database management systems** (DBMS) and the security technique used to protect these critical systems

Databases have a sophisticated interaction protocol, Structured Query Language (**SQL**), which **is complex**

Effective database security requires a strategy based on a **full understanding** of the security vulnerabilities of SQL

The typical organization **lacks full-time** database security personnel

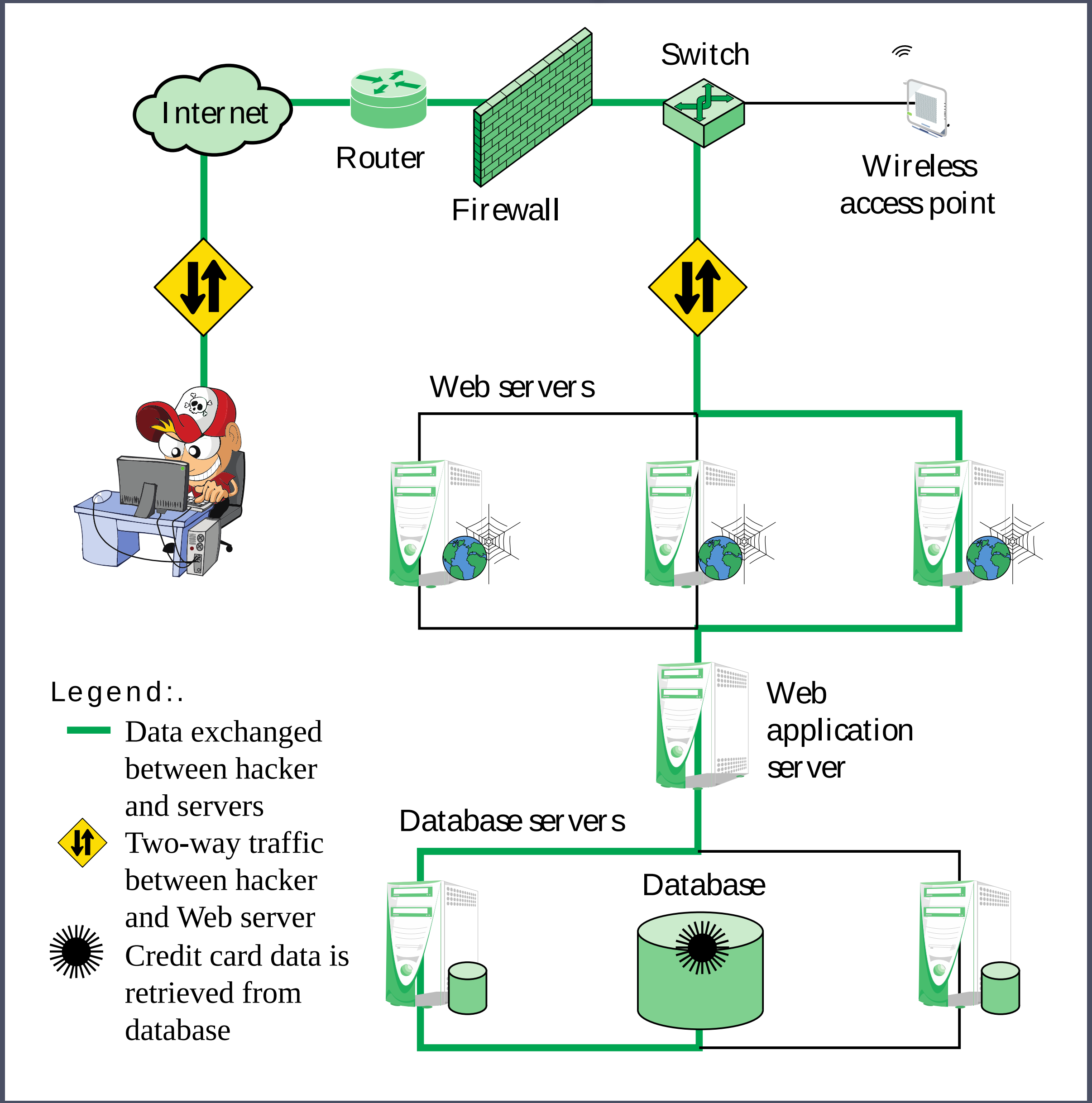
Most enterprise environments consist of **a heterogeneous mixture of database platforms**, enterprise platforms, and OS platforms, creating an additional complexity hurdle for security personnel

The increasing **reliance on cloud technology** to host part or all of the corporate database

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

Typical SQL Injection Attack



Injection Technique

The SQLi attack typically works by **prematurely terminating a text string and appending a new command**

Because the inserted command may have additional strings appended to it before it is executed, the attacker terminates the injected string with a comment mark "- -"

Subsequent text **is ignored** at execution time

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

- A malicious user could rely on data already present in the system or database to trigger an SQL injection attack, so when the attack occurs, the input that modifies the query to cause an attack does not come from the user, but from within the system itself

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

Inband Attacks

- Uses the same communication channel for injecting SQL code and retrieving results
- The retrieved data are presented directly in application Web page
- Include:

Tautology

This form of attack injects code in one or more conditional statements so that they always evaluate to true – 1=1

End-of-line comment

After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments

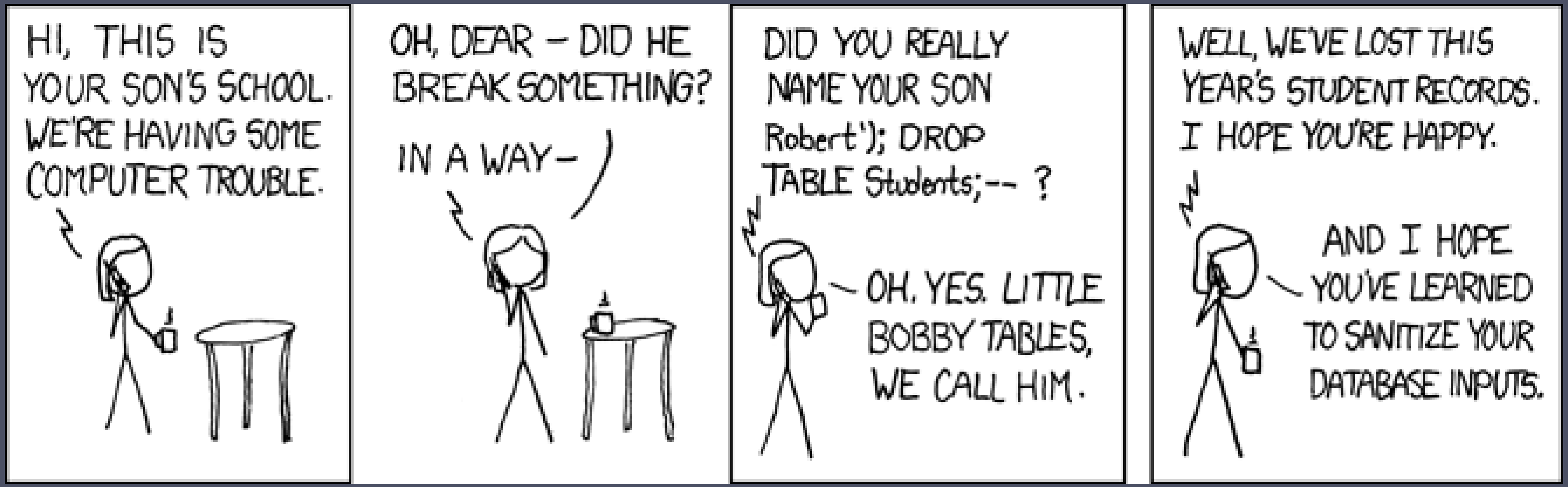
Piggybacked queries

The attacker adds additional queries beyond the intended query, piggy-backing the attack on top of a legitimate request

SQL for Inband Attacks

- Tautology using a PHP example:
  - `$query="SELECT info FROM user WHERE name='$_GET["name"]' AND pwd = '$_GET["pwd"]'"`
  - User submits " `' OR 1=1 --"` for the name field
  - Query becomes: `SELECT info FROM user WHERE name= ' ' OR 1=1 -- AND PWD= ' '`
  - The password check is ignored because of the `--`
- Attacks use the comment field to disable other parts of the SQL
- Piggybacked queries use the semi-colon to execute additional statements – see "Little Bobby Tables"

Little Bobby Tables

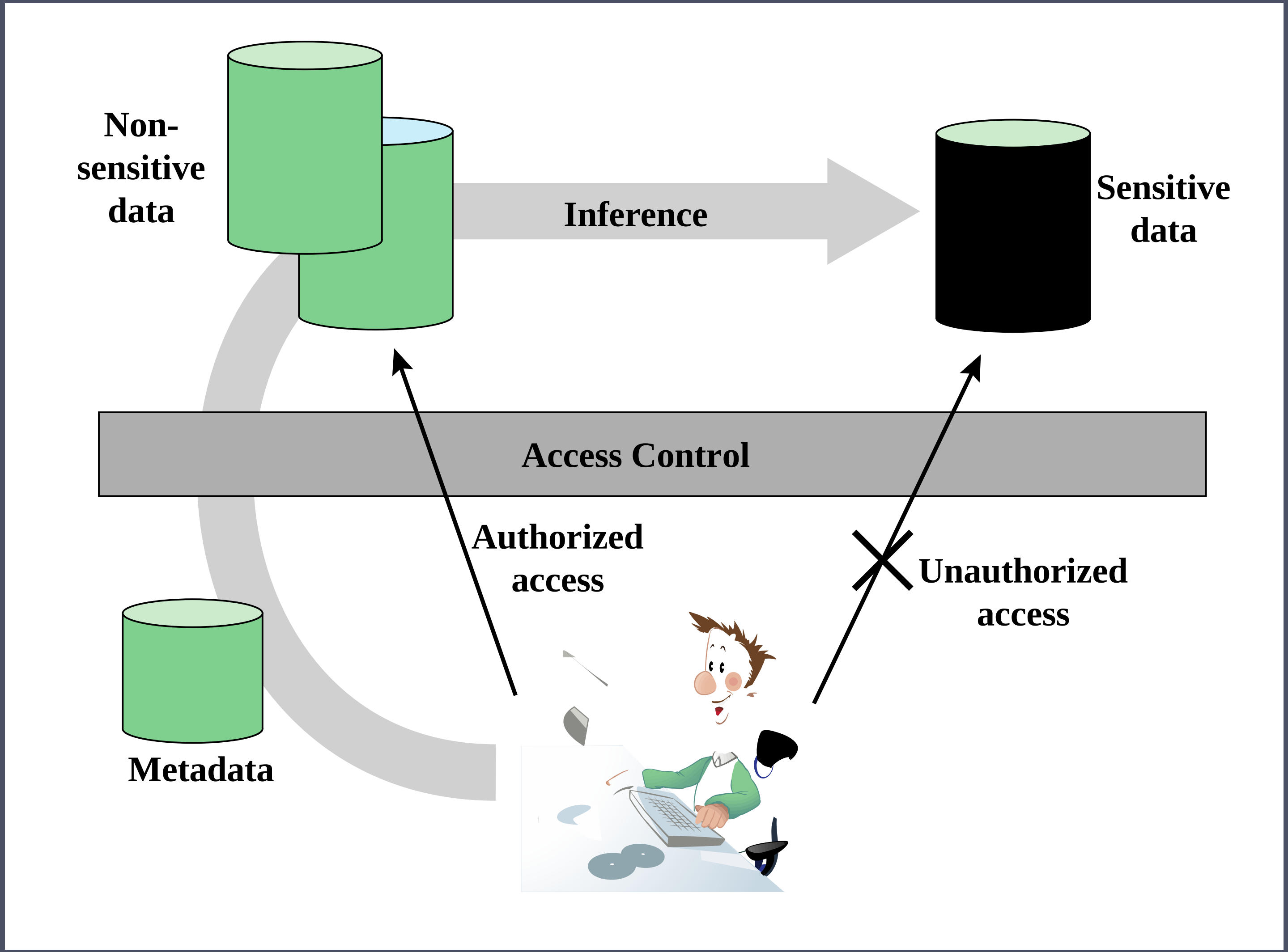


https://xkcd.com/327/

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
- Include:
    - 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 attack is considered a **preliminary, information-gathering step** for other attacks
    - 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, by injecting statements which are true or false – false terminates the SQL and reveals internal structure

Indirect Information Access via Inference Channel



Inference Detection

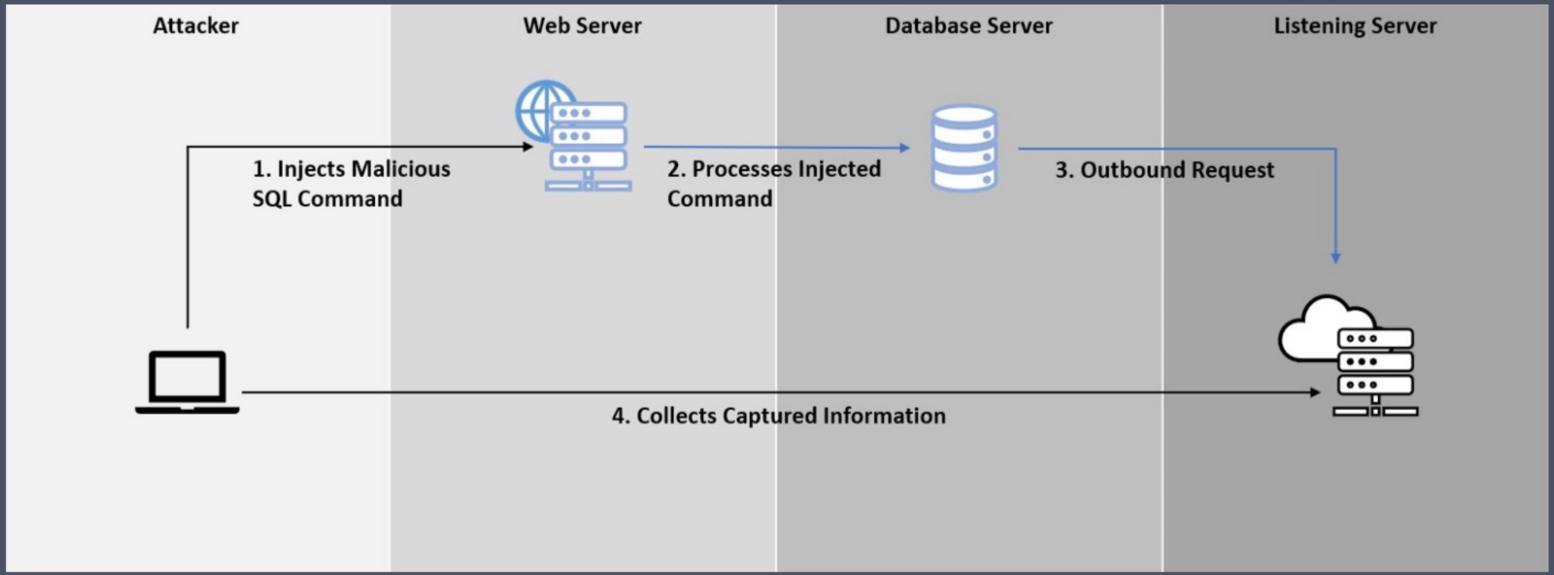
- **Two approaches**
  - Inference detection **during database design**
    - Approach removes an inference channel **by altering the database structure or by changing the access control regime** to prevent inference
    - Techniques in this category often result in **unnecessarily stricter access controls** 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**

- Some inference detection algorithm is needed for either of these approaches
- Progress has been made in devising specific inference detection techniques for multilevel secure databases and statistical databases

Out-of-Band Attack

- Data are retrieved using a different channel

- This can be used when **there are limitations on information retrieval**, but outbound connectivity from the database server is lax, e.g., when the results are returned in an email



SQLi Countermeasures

- Three types:

Defensive coding

- Manual defensive coding practices
- Parameterized query insertion
- SQL DOM

Detection

- Signature-based
- Anomaly-based
- Code analysis

Run-time prevention

- Check queries at runtime to see if they conform to a model of expected queries

Database Access Control

Database access control system determines:

If the user has access to the **entire database or just portions** of it

What access rights the user has (**create, insert, delete, update, read, write**)

Can support a range of administrative policies

Centralized administration

- **Small number of privileged users** may grant and revoke access rights

Ownership-based administration

- **The creator of a table** may grant and revoke access rights to the table

Decentralized administration

- **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

SQL-based Access Controls

- Nearly all DBMS have an inbuilt DAC Mechanism
- Two commands for managing access rights:
  - Grant
    - Used to grant one or more access rights or can be used to assign a user to a role
  - Revoke
    - Revokes the access rights

- Typical access rights are:
  - Select
  - Insert
  - Update
  - Delete
  - References

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
- Categories of database users:

Application owner	End user	Administrator
• An end user **who owns database objects** as part of an application	• An end user **who operates** on database objects via a particular application **but does not own** any of the database objects	• User who has **administrative responsibility** for part or all of the database

Summary

- The need for database security
- Database management systems
- Relational databases
  - Elements of a relational database system
  - Structured Query Language
- SQL injection attacks
  - A typical SQLi attack
  - The injection technique
  - SQLi attack avenues and types
  - SQLi countermeasures
- Database access control
  - SQL-based access definition
  - Role-based access control
- Inference