Database Security & SQL Injection

Dr. Yeonjoon Lee

Outline

- Database Security Background
- SQL Injection Attacks
- Access Control
- Inference

Overview

- We first briefly introduce the basic concepts of database management systems (DBMS) and relational database.
- Then we look at some attacks, specifically SQL injection attacks.
- Finally we talk about access control in modern DBMS and Inference threat.
- Chapters 5.1 ~ 5.5 of the first book (Computer Security: Principles and Practice) is covered in the slides.

The Need for Database Security

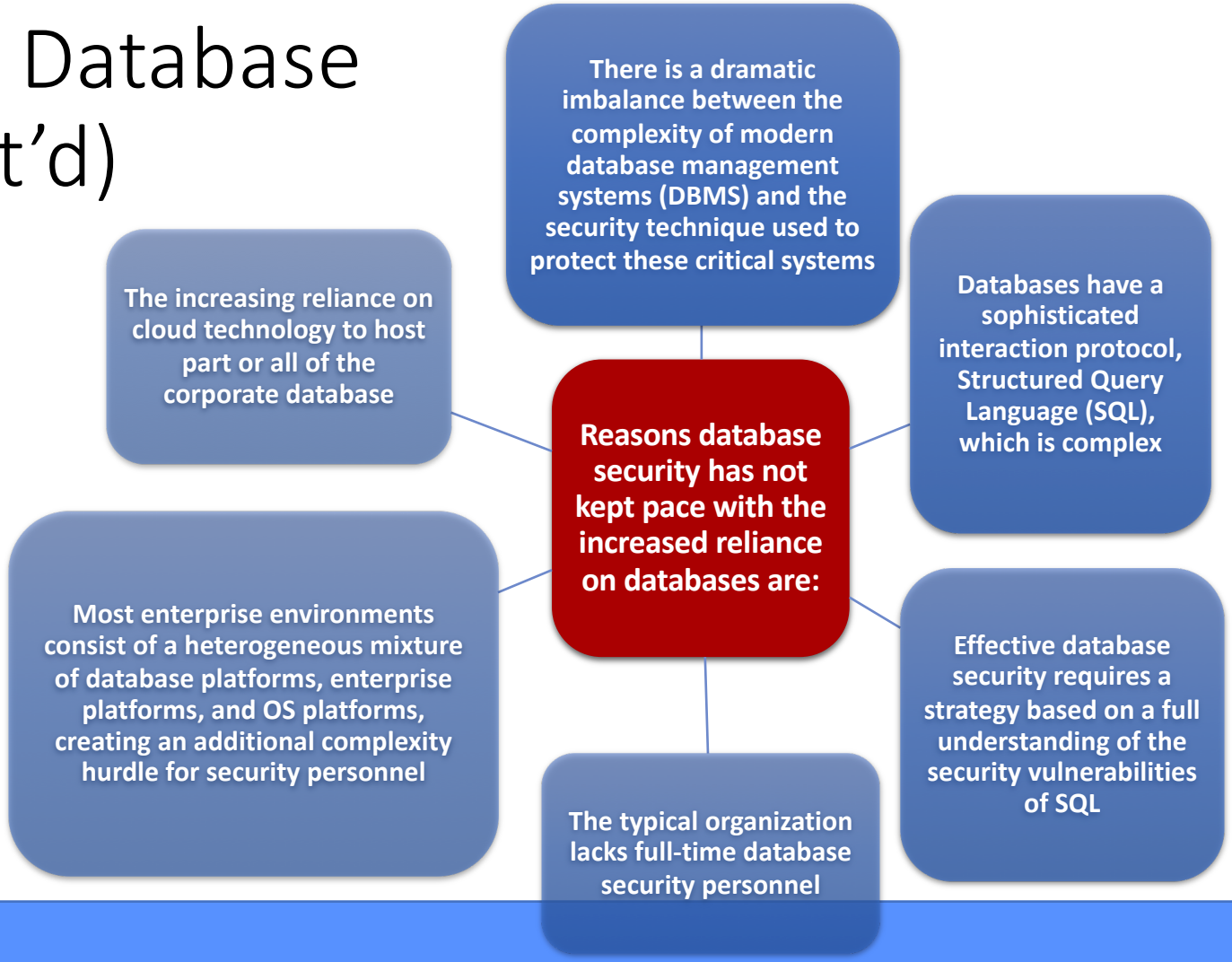
- It is important to provide customers and business partners with access to information.

- A database concentrates information in a single logical system. Some information are sensitive. Examples:
  - Corporate Financial data
  - Phone records
  - Customer & employee information
  - Product information
  - Medical records ...

The Need for Database Security (cont'd)

- Database security fails to keep pace with the reliance on databases:
  - The increasing complexity of DBMS brings in new vulnerabilities together with new features and services. There is a gap between such complexity and the techniques used to protect them.

The Need for Database Security (cont'd)



Database Management Systems

➢ **Definition of Database:**

- A *database* is a structured collection of data electronically stored and accessed by one or more applications.
- In addition to data, it contains the relationship between data or groups of data.

➢ **Definition of DBMS:**

- A *database management system* (DBMS) is a suite of programs for constructing and maintaining the database and for offering adhoc query facilities to multiple users and applications. A *query language* provides a uniform interface to the database for users and applications.

DBMS Architecture

- **Languages**
  - **Data Definition Languages (DDL)** define the database logical structure and procedural properties, which are represented by a set of database description tables.
  - **Data Manipulation Languages (DML)** provide a powerful set of tools for application developers.
  - **Query languages** are declarative languages designed to support end users.

DBMS Architecture (cont'd)

- **Tables**
  - Database description tables manage the physical database.
  - Authorization tables ensure the user has permission to execute the query language statement on the database.
  - Concurrent access tables prevent conflicts when simultaneous, conflicting commands are executed.

- **Interfaces**
  - File manager module
  - Transaction manager module

DBMS Architecture (cont'd)

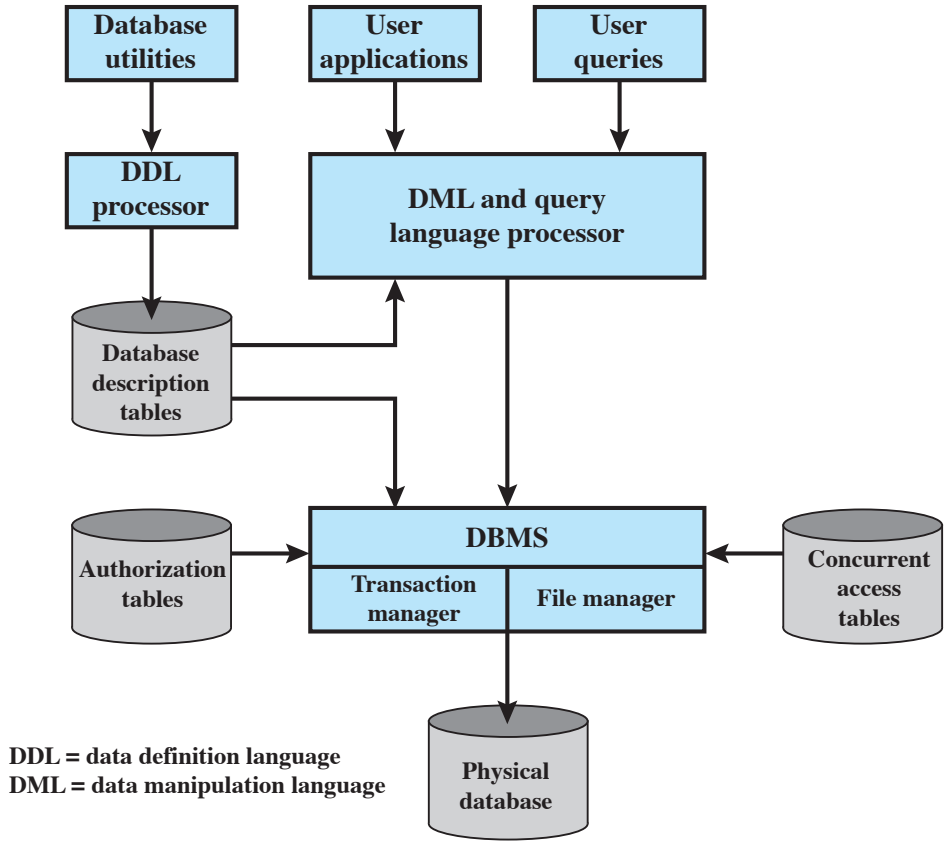


Figure 5.1 DBMS Architecture

Relational Database

- Flat Table:
  - A single large table.
  - Has rows and columns, like a spreadsheet.
  - Hard to update when adding other types of information (has to add columns).

- Relational Database:
  - Multiple tables tied together by a unique identifier.
  - Uses a query language that composes declarative statements to retrieve information.

Building Blocks of Relational Database

- Relation:
  - a flat table/file
- Primary key:
  - a portion of a row to uniquely identify a row. May consist one or more column names.
- Foreign key:
  - the attributes that define the primary key in one table while appear as well in another table. Unlike primary key, a foreign key value can appear multiple times.

Building Blocks of Relational Database

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

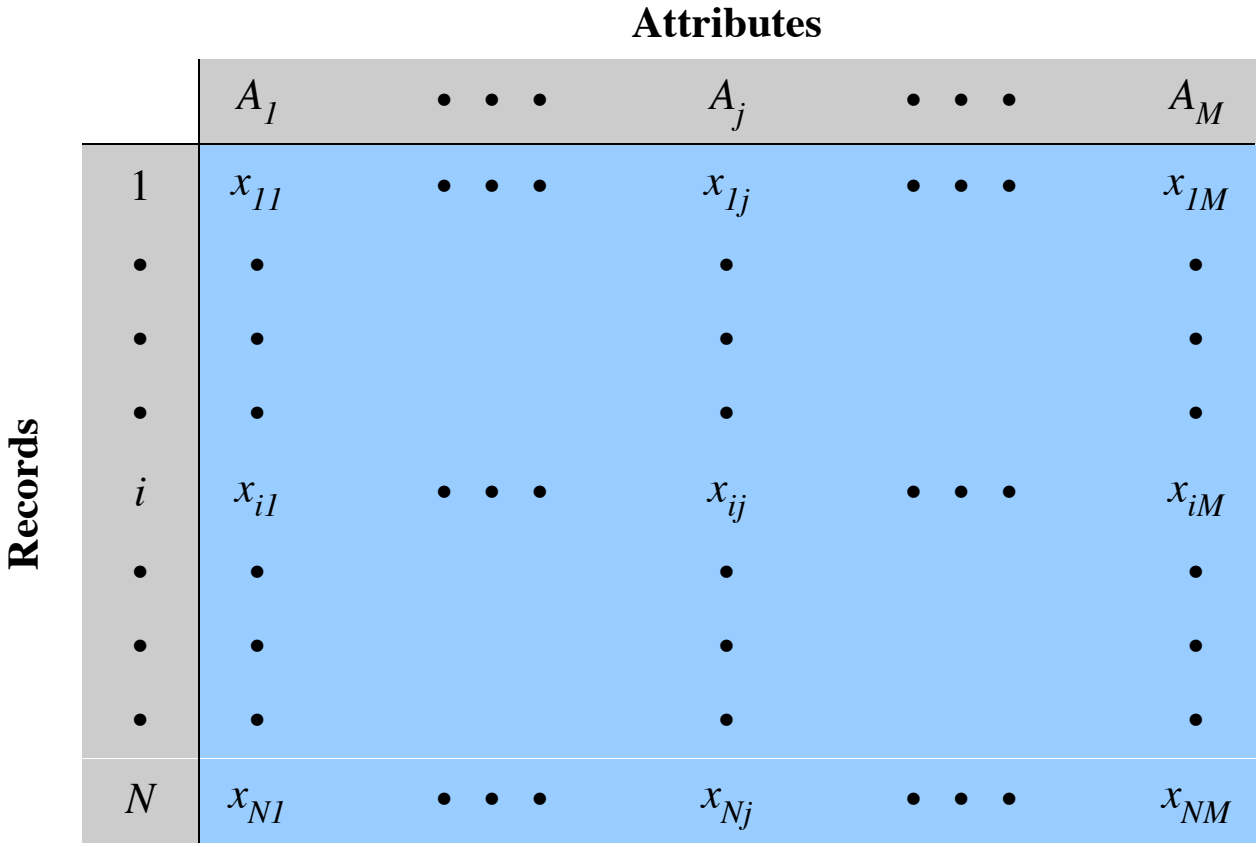


Figure 5.3 Abstract Model of a Relational Database

An Example Relational Database

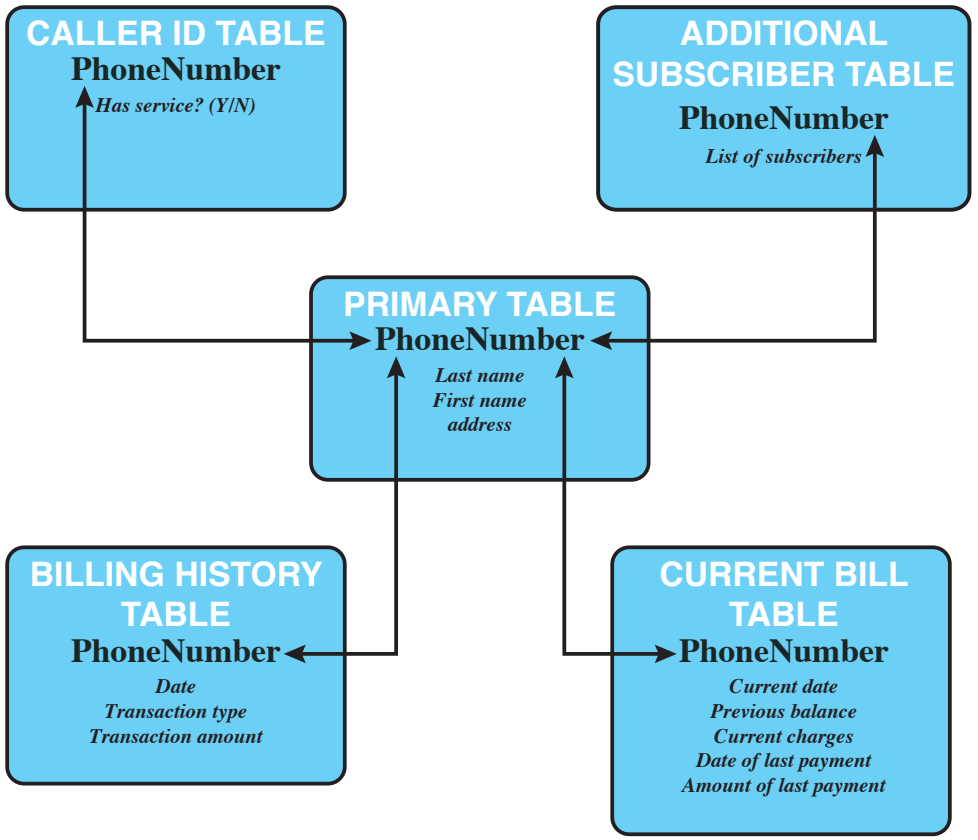


Figure 5.2 Example Relational Database Model. A relational database uses multiple tables related to one another by a designated key; in this case the key is the PhoneNumber field.

Views

- A view is a virtual table that contains the result of a query.
- Selected rows & columns from one / more tables.
- Can provide restricted access to a relational database so that a user or application only has access to certain rows / columns.

An Example View

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

(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

Structured Query Language (SQL)

- Structured Query Language (SQL) is a standardized language that can be used to define schema, manipulate, and query data in a relational database.
- To create two tables department and employee:

```
CREATE TABLE department (
   Did INTEGER PRIMARY KEY,
   Dname CHAR (30),
   Dacctno CHAR (6) )
CREATE TABLE employee (
   Ename CHAR (30),
   Did INTEGER,
   SalaryCode INTEGER,
   Eid INTEGER PRIMARY KEY,
   Ephone CHAR (10),
   FOREIGN KEY (Did) REFERENCES department (Did) )
```

Retrieve Information Using Select and View

- Not only can SQL statements be used to create tables, it can also insert and delete data in tables, create views, and retrieve data with query statements. A few examples:

  - To return the Ename, Eid, and Ephone fields from the employee table for all employees assigned to department 15:

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

  - To create a view:

```
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
```

Basics About SQL Injection Attacks

- Modern web applications involve interactions with backend databases that contain volumes of information. A webpage can make SQL queries to ask the databases for critical user information.
- An SQL injection (SQLi) attack is designed to send malicious SQL commands to the database server. In other words, it exploits vulnerabilities in the database layer of a web application.
- The most common attack goal is bulk extraction of data (to dump database tables). Can also be exploited to modify or delete data, execute arbitrary system commands or launch denial-of-service (DoS) attacks.

A Typical SQLi Attack

1. The hacker finds a vulnerability in a web application and injects an SQL command to a database by sending it to the web server.
2. The web server receives the malicious code and forwards it to the web application server.
3. The web application server receives the malicious code from the web server and forwards it to the database server.
4. The database server executes the malicious code on the database.
5. The web application server generates a page with sensitive data from the database.
6. The web server sends sensitive information to the hacker.

A Typical SQLi Attack (cont'd)

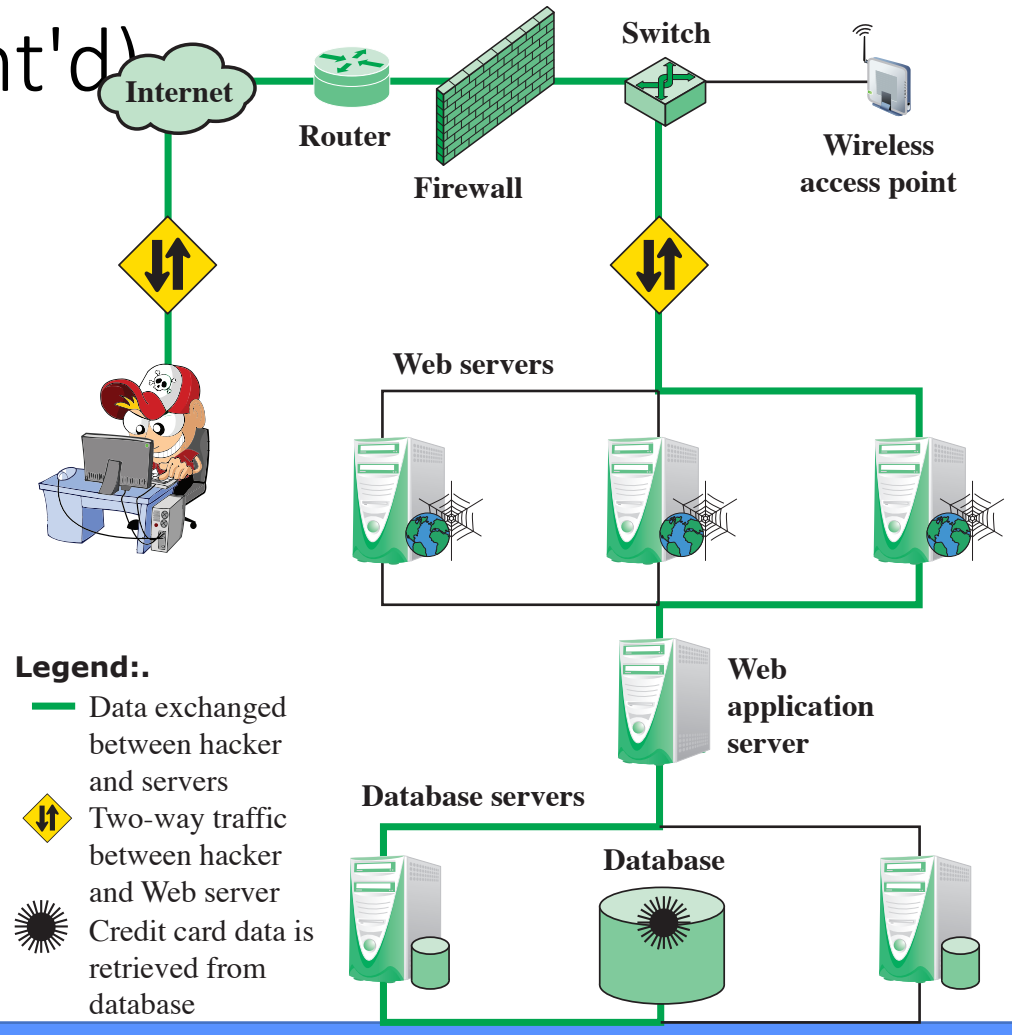


Figure 5.5 Typical SQL Injection Attack

The Injection

- Suppose we build an SQL query string using the following code snippet:

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

- If the user enters Redmond as the name of the city, the composed SQL query string is:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'
```

The Injection (cont'd)

- The goal of the attacker, however, is to prematurely terminate the string and append a new command.
- Suppose the user enters another string Boston'; DROP table OrdersTable --, the resulted SQL query:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Boston';
-- Oh NO!
DROP table OrdersTable -- ...
-- NOTE that all commands following the double dash are dropped!
```

- The ; separates two commands. The -- is an indicator that the remaining text of the current line is a comment and not to be executed. The SQL server first selects all records in OrdersTable where ShipCity is Redmond. Then it drops that table, which forms an attack.

SQLi Attack Avenues

1. User input: Attackers inject SQL commands by providing crafted input. User input typically comes from form submissions that are sent to the web application via HTTP GET or POST requests.
2. Server variables:
   - Server variables are a collection of variables that contain HTTP headers, network protocol headers, and environmental variables.
   - If these variables are logged to a database without sanitization, it creates an SQL injection point since an attacker can place data directly into the headers.
3. Second-order injection: The attacker relies on data already present in the system to trigger an SQL injection attack. The input that modifies the query to cause an attack does not come from the user but from the system itself.

SQLi Attack Avenues (cont'd)

4. Cookies: Cookies are used to restore the client's state information when the user returns to the same web application. Since the client has control over cookies, an attacker could alter cookies such that when the application server builds an SQL query based on the cookie's content, the structure of that query is modified.

5. Physical input: The input may not be electronic. It can also be conventional barcodes, RFID tags, or even paper forms which are scanned using optical character recognition and passed to a DBMS.

SQLi Type: Inband Attacks

- Inband attacks: Uses the same communication channel for injecting SQL code and retrieving results.
  - Tautology: Injects code in one or more conditional statements so that they evaluate to true. Can be used to bypass checks.
  - End-of-line comment: After injecting code into a field, legitimate code that follows are disabled through usage of end of line comments (adding --).
  - Piggybacked queries: The attacker adds additional queries beyond the intended query, piggybacking the attack on top of a legitimate request.

An Example of Tautology Attack

- The following code snippet requires the user to enter a valid name and a password:

```
$query = "SELECT info FROM user WHERE name =
         '$_GET["name"]' AND pwd = '$_GET["pwd"]'";
```

- Suppose the attacker submits `' OR 1=1 --` for the name field:

```
SELECT info FROM users WHERE name = '' OR 1=1 -- AND pwpd = ''
```

- The injected code disables the password check and turns the entire WHERE clause into a tautology. Because the conditional is a tautology, the query always evaluates to true for each row in the table and returns all of records.

SQLi Type: Inferential Attack

- There is no actual transfer of data. The attacker is able to reconstruct the information by sending particular requests and observing the resulting behavior of the web application.
  - Illegal (or logically incorrect) queries: This attack is often considered a preliminary, information-gathering step for other attacks. The vulnerability leveraged by this attack is that the default error page is often descriptive and the error messages generated can reveal injectable parameters to an attacker.
  - Blind SQL injection: The attacker asks the server true / false questions. If true, the site functions normally; if false, although there is no descriptive error message, the page differs from the normal page. It allows attackers to infer the data present in a database system even when the system does not display any erroneous information to the attacker.

SQLi Type: Out-of-band Attack

- Data are retrieved using a different channel (like an email where the results of the query is generated).

- This can be used when there are limitations in information retrieval, but outbound connectivity from the database server is lax.

SQLi Countermeasures

- Examples of defensive coding:
  - Manual defensive coding practices: Doing input validation.
    - Input typechecking: To check that inputs that are supposed to be numeric contain no characters other than digits.
    - Pattern matching: To distinguish normal input from abnormal input.
  - Parameterized query insertion: Allows the application developer to more accurately specify the structure of an SQL query and pass the parameters to it separately such that any unsanitary user input is not allowed to modify the query structure.
  - SQL DOM: A set of APIs that enables automated data type validation and escaping. Always typecheck.

SQLi Detection

- Signature based: Attempts to match specific attack patterns. Such an approach must be constantly updated and may not work against self-modifying attacks.

- Anomaly based: Attempts to define normal behavior and then detect behavior patterns outside the normal range. There is a training phase, in which the system learns the range of normal behavior, followed by the actual detection phase.

- 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.

DBMS Access Control Overview

- For users who are granted access to the database, an access control system controls access to specific portions of the database. Administrative policies:
  1. Centralized administration: A small number of privileged users may grant and revoke access rights.
  2. Ownership-based administration: The owner (creator) of a table may grant and revoke access rights to the table.
  3. 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.
- As with any access control system, a database access control system distinguishes different access rights, including create, insert, delete, update, read, and write.

SQL-based Access Control: GRANT

- SQL provides two commands for managing access rights, GRANT and REVOKE.

```
GRANT            { privileges | role }
[ON              table]
TO               { user | role | PUBLIC }
[IDENTIFIED BY   password]
[WITH            GRANT OPTION]
```

- All other fields are straightforward. The GRANT OPTION field indicates that the grantee can grant this access right to other users with or without the grant option.
- An example:

```
GRANT SELECT ON ANY TABLE TO ricflair
```

- This statement enables user ricflair to query any table in the database.

Various Access Rights

- Select: Grantee may read entire database, individual tables or specific columns in a table.
- Insert: Grantee may insert rows in a table or insert rows with values for specific columns in a table.
- Update: Semantics is similar to INSERT.
- Delete: Grantee may delete rows from a table.

SQL-based Access Control: REVOKE

- The syntax of REVOKE:

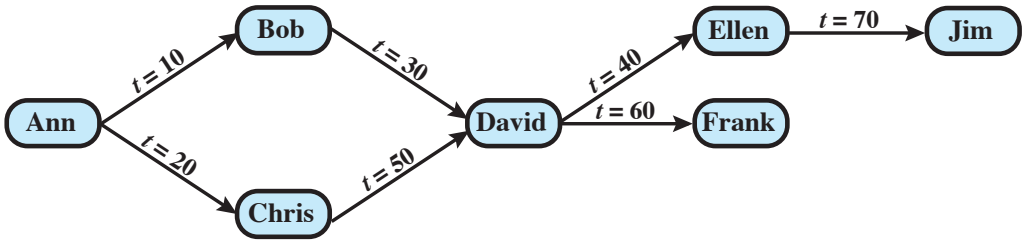
```
REVOKE { privileges | role }
[ON    table]
FROM   { user | role | PUBLIC }
```

- Revoking the access rights of the previous example:

```
REVOKE SELECT ON ANY TABLE FROM ricflair
```

Cascading Authorizations

- Suppose we always use grant options:
  1. Ann grants access to Bob at t=10 and to Chris at t=20.
  2. Bob grants access to David at t=30.
  3. Chris redundantly grants access to David at t=50.
  4. David grants access to Ellen at t=40, who in turn grants it to Jim at t=70.
  5. David grants access to Frank at t=60.



Revoking Cascaded Authorization

- Just as granting privileges, the revocation of privileges also cascades.
- If Ann revokes the access right to Bob and Chris, then the access right is also revoked from David, Ellen, Jim, and Frank.
- But what if Bob revokes privilege from David?
- A complication arises when a user (David) receives the same access right multiple times.

Revoking Cascaded Authorization (cont'd)

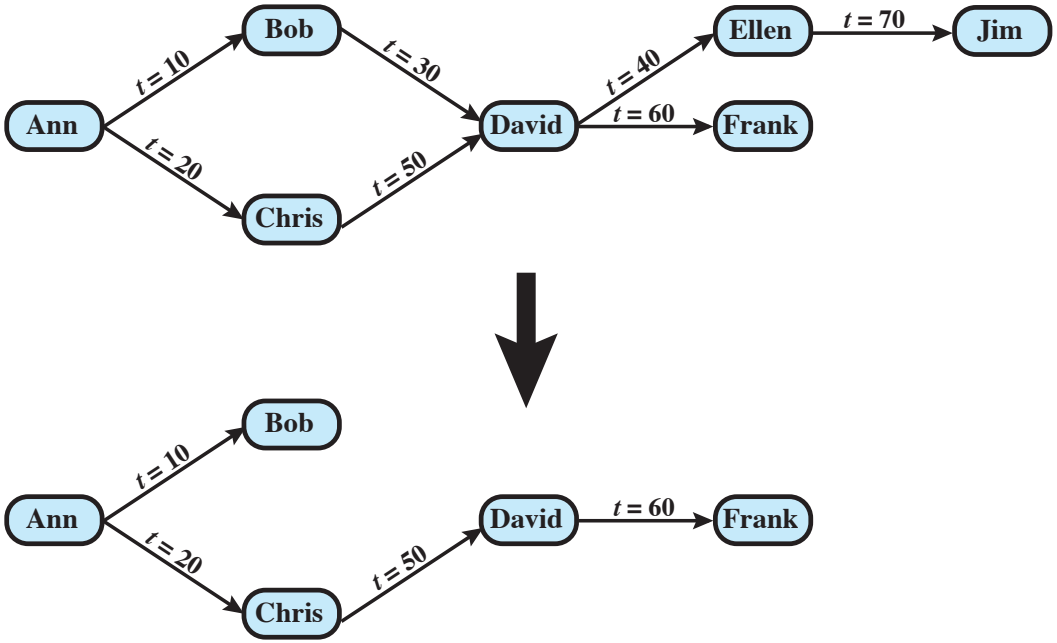
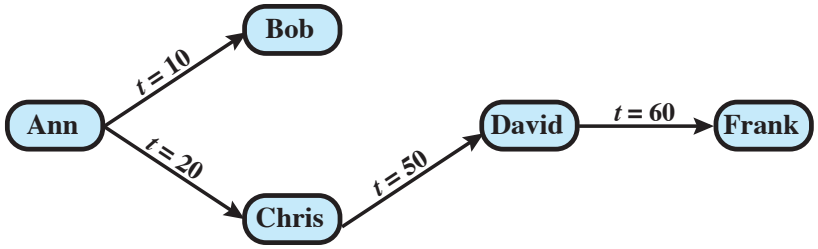


Figure 5.6 Bob Revokes Privilege from David

Revoking Cascaded Authorization (cont'd)



- Generalization:

  When user A revokes an access right, any cascaded access right is also revoked, unless that access right would exist even if the original grant from A had never occurred.

Three Types of Database Users

- An individual may use a variety of applications to perform a variety of tasks, each of which requires its own set of privileges. We can roughly classify users into 3 categories:

1. Application owner: An end user who owns database objects (tables, columns, rows) as part of an application. That is, the database objects are generated by the application or are prepared for use by the application.
2. End user other than application owner: An end user who operates on database objects via a particular application but does not own any of the database objects.
3. Administrator: User who has administrative responsibility for part or all of the database.

Role-based Access Control (RBAC)

- We can make statements concerning the three types of users:
  1. An application has associated with it a number of tasks. Each task requires specific access rights to portions of the database. One or more roles can be defined that specify the needed access rights.
  2. The application owner may assign roles to end users.
  3. Administrators are responsible for more sensitive or general roles. Administrators in turn can assign users to administrative-related roles.
- A database RBAC facility needs to provide the following capabilities:
  1. Create and delete roles.
  2. Define permissions for a role.
  3. Assign and cancel assignment of users to roles.

Case Study: RBAC of Microsoft SQL Server

- SQL server supports 3 types of roles:
  - Fixed server roles: Defined at the server level. Database administrators can use these fixed server roles to distribute administrative tasks to personnel and give them only the rights they absolutely need.
  - Fixed database roles: Operate at the level of an individual database.
    - Some roles like db_accessadmin and db_securityadmin are designed to assist a DBA with delegating administrative responsibilities.
    - Others roles like db_datareader and db_datawriter are designed to provide blanket permissions for an end user.
  - User-defined roles

Example

Role	**Permissions**
Fixed Server Roles	
sysadmin	Can perform any activity in SQL Server and have complete control over all database functions
serveradmin	Can set server-wide configuration options, shut down the server
setupadmin	Can manage linked servers and startup procedures
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords
processadmin	Can manage processes running in SQL Server
dbcreator	Can create, alter, and drop databases
diskadmin	Can manage disk files
bulkadmin	Can execute BULK INSERT statements
Fixed Database Roles	
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove user IDs
db_datareader	Can select all data from any user table in the database
db_datawriter	Can modify any data in any user table in the database
db_ddladmin	Can issue all Data Definition Language (DDL) statements
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_denydatareader	Can deny permission to select data in the database
db_denydatawriter	Can deny permission to change data in the database

Case Study: RBAC of Microsoft SQL Server (cont'd)

- SQL server supports 3 types of roles:
  - Fixed server roles
  - Fixed database roles
  - User-defined roles: Can be created by the users and assign access rights to portions of the database.
    - Standard role: An authorized user can assign other users to the role.
    - Application role: associated with an application instead of users. Activated when an application executes the appropriate code. A user who has access to the application can use the application role for database access.

Other Threats: Inference

- Inference: the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.

- The inference problem arises when the combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of a higher sensitivity.

Inference (cont'd)

- The attacker may make use of nonsensitive data as well as metadata. Metadata refers to knowledge about correlations or dependencies among data items that can be used to deduce information not otherwise available to a particular user.
- The information transfer path by which unauthorized data is obtained is referred to as an **inference channel**.

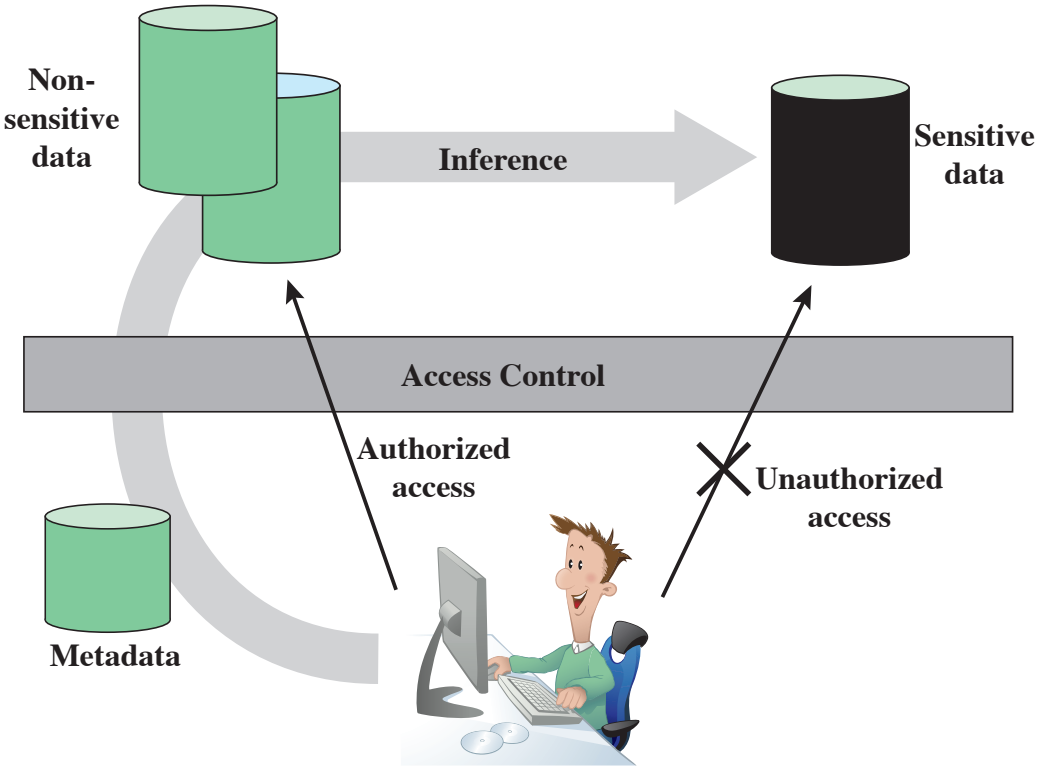


Figure 5.7 Indirect Information Access Via Inference Channel