Database Security

CSE 565 - Fall 2025

Computer Security

Hongxin Hu (hongxinh@buffalo.edu)

Updates

- **Project 2 SQL Injection Attack**
  - Deadline: Tuesday, Oct 7
- **Assignment 2**
  - Deadline: Thursday, Oct 9
- **Midterm Exam**
  - Deadline: Thursday, October 16

Review of Access Control Types

- We previously studied four types of access control
  - mandatory access control (MAC)
  - discretionary access control (DAC)
  - role-based access control (RBAC)
  - attribute-based access control (ABAC)
- Many of them can be used in databases
- There are also security challenges unique to database management systems (DBMSs)

Overview

- Review of relational databases
- Database security issues
  - SQL injection attacks
  - Access control mechanisms
- Newer topics include outsourcing, database encryption

Relational Databases

- A database is a structured collection of data
- A database management system (DBMS) allows one to construct, manipulate, and maintain the database
  - it provides facilities for multiple users and applications
- A query language specifies how the data can be created, queried, updated, etc.
- In relational databases, all data are stored in tables (called relations)
  - each record (called tuple) corresponds to a row of a table
  - each column lists an attribute

Relational Databases

- Example of a table

EmployeeID	Name	Salary	DepartmentID
1	Alice	75	3
2	Bob	60	2
3	Carl	90	1
4	David	70	3

- A primary key uniquely identifies each row in a table
  - it can consist of one or more attributes
  - in the above table, Employee ID can be used as a primary key
- We create a relationship between tables by linking their attributes together
  - this is done by means of foreign keys

Relational Databases

- A foreign key is one or more attributes that appear as the primary key in another table

EID	Name	Salary	DID
1	Alice	75	3
2	Bob	60	2
3	Carl	90	1
4	David	70	3

DeptID	Name	Phone
1	Administration	1234567
2	HR	1234568
3	Sales	1234569

- A view is a virtual table that displays selected attributes from one or more tables

EID	Name	DID
1	Alice	3
2	Bob	2
3	Carl	1
4	David	3

EID	Name	DeptName
1	Alice	Sales
2	Bob	HR
3	Carl	Administration
4	David	Sales

Relational Databases

- Structured Query Language (SQL) is a widely used language that allows one to manipulate databases
- SQL statements can be used to
  - Create tables
  - Insert and delete data in tables
  - Create views
  - Retrieve data with query statements

Relational Databases

- SQL examples
  - table creation

```
CREATE TABLE Employee (
EmployeeID INTEGER PRIMARY KEY,
Name CHAR (30),
Salary INTEGER,
DepartmentID INTEGER )
```

  - retrieving (querying) information

```
SELECT EmployeeID, Name
FROM Employee
WHERE Salary >= 70
```

Relational Databases

- SQL examples (cont.)
  - View creation

```
CREATE VIEW Employee2 (EID, Name, DeptName)
AS SELECT E.EmployeeID, E.Name, D.Name
FROM Employee E Department D
WHERE E.DepartmentID = D.DeptID
```

- Limited views are common as a security mechanism

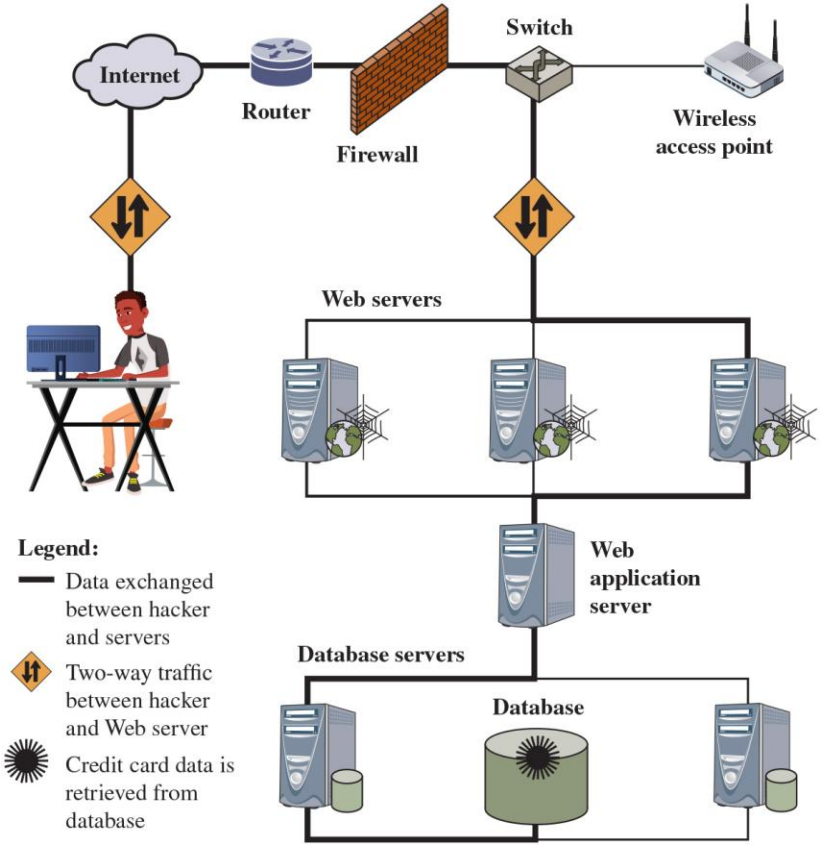
Database Security

- Database security issues
  - users and authentication
    - authenticating users, assigning privileges correctly
  - secure communication between client and server
  - vulnerabilities in DBMS implementation
    - sanitizing input
    - SQL worms
    - limiting who can connect to DBMS server

SQL Injection Attacks (SQLi)

- Most common attack goal is to extract sensitive 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
- 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

SQL Injection Attacks



SQL Injection Attack

- Many web applications take user input from a form
- Often this user input is used literally in the construction of an SQL query submitted to a database. For example:

```
SELECT user FROM table
    WHERE name = 'user_input';
```

- An SQL injection attack involves placing SQL statements in the user input

Login Authentication Query

- Standard query to authenticate users:

  select * from users where user='$usern' AND pwd='$password'

- Classic SQL injection attacks
  - Server side code sets variables $username and $passwd from user input to web form
  - Variables passed to SQL query

  select * from users where user='$username' AND pwd='$passwd'

- Special stringes can be entered by attacker

  select * from users where user='M' OR '1=1' AND pwd='M' OR '1=1'

- Result?
  - access obtained without password

Some improvements …

- Query modify:
  - select user, pwd from users where user='**$usern**‘
  - **$usern**=“M' OR '1=1”;
    - Result: the entire table
  - **$usern**=“M' ; drop table user;” ?

Correct Solution

- We can use an **Escape** method, where all “malicious” characters will be changed:
- Escape(“t ' c”) gives as a result “t \' c”

  select user, pwd from users where user='$usern'

  $usern=escape(“M' ;drop table user;”)
- The result is the safe query:

  select user,pwd from users

  where user='M\' drop table user;\''

Database Access Control

- Commercial DBMSs often provide discretionary or role-based AC
  - centralized administration
  - ownership-based administration
  - decentralized administration
- Key components in DBMS access control
  - privileges
  - views
  - roles
  - row-level access control

Database Access Control

- Privileges
  - access rights: create, select, insert, update, delete, add references
  - system privilege
    - a permission or authorization that grants a user or role certain administrative rights, such as creating, altering, or dropping database objects
    - e.g., ALTER DATABASE or SELECT ANY TABLE
  - object privilege
    - a right to perform a particular action on a specific object such as tables, views, procedures, and types
    - e.g., SELECT, INSERT, UPDATE, DELETE

Database Access Control

- Granting and revoking privileges (or roles) with SQL
  - granting privileges has the following syntax

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

  - revoking privileges

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

Database Access Control

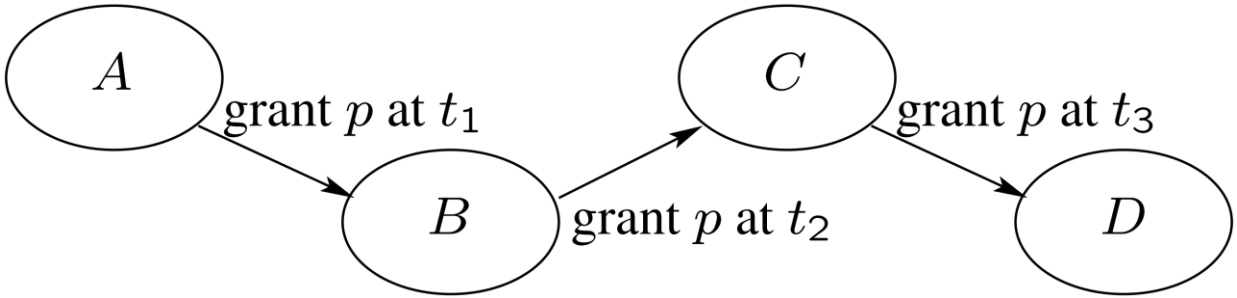
- Examples of granting and revoking privileges
  - system privileges
    - `GRANT create table TO Bob [WITH GRANT OPTION]`
    - `REVOKE create table FROM Bob`
    - users with GRANT OPTION can not only grant the privilege to others, but also revoke the privilege from any user

Database Access Control

- Examples of granting and revoking privileges
  - object privileges
    - `GRANT select ON table1 TO Bob [WITH GRANT OPTION]`
    - `REVOKE select ON table1 FROM Bob`
    - user who revokes a particular object privilege must be the direct grantor of the privilege
    - there is a cascading effect when an object privilege is revoked

Database Access Control

- Cascading effect:
  - when a privilege is being revoked, all other privileges that resulted from it get revoked as well
  - for example, the privilege is being revoked from C or B



- Difficulties arise if a privilege has been granted through different paths
  - the cascading effect can either apply to all privileges or be based on timestamps

Database Access Control

- Views
  - access control is based on attributes (columns) and their contents
  - example: some users can see employees and their departments, but not salaries
    - given table `Employee(EmployeeID, Name, Salary, DepartmentID)`
    - `CREATE VIEW Employee1 AS SELECT EmployeeID, Name, DepartmentID from Employee`
    - grant select privileges on the view `Employee1`

Database Access Control

- To create a view
  - the creator must have been explicitly (not through roles) granted one of SELECT, INSERT, UPDATE, or DELETE object privileges on all base objects underlying the view or corresponding system privileges
- To grant access to the view
  - the creator must have been granted the corresponding privileges with GRANT OPTION to the base tables
- To access the view
  - the creator must have the proper privilege for the underlying base tables

Database Access Control

- RBAC naturally fits database access control
- The use of roles allows for
  - management of privileges for a user group (user roles)
    - DB admin creates a role for a group of users with common privilege requirements
    - DB admin grants required privileges to a role and then grants the role to appropriate users
  - management of privileges for an application (application roles)
    - DB admin creates a role (or several roles) for an application and grants necessary privileges to run the application
    - DB admin grants the application role to appropriate users

Database Access Control

- User-roles assignment
  - to grant a role, one needs to have GRANT ANY ROLE system privilege or have been granted the role with GRANT OPTION
    - `GRANT ROLE clerk TO Bob`
  - to revoke a role from a user, one needs to have the GRANT ANY ROLE system privilege or have been granted the role with GRANT OPTION
    - `REVOKE ROLE clerk FROM Bob`
  - users cannot revoke a role from themselves

Database Access Control

- Role-permission assignment
  - to grant a privilege to a role, one needs to be able to grant the privilege
    - `GRANT insert ON table1 TO clerk`
  - to revoke a privilege from a role, one needs to be able to revoke the privilege
    - `REVOKE insert ON table1 FROM clerk`
- DBMS implementation can have different types of roles
  - e.g., server roles, database roles, user-defined roles

Statistical Databases

- Common data protection models include:
  - K-anonymity
    - each individual's information is indistinguishable from at least "k" other individuals in the same dataset
    - designed for anonymized dataset release
    - protection is achieved via removing some attributes and generalizing others
  - Differential privacy
    - the presence of a single individual in a dataset cannot be determined
    - was formulated for statistical queries
    - protection is achieved via adding noise

New Trends in Database Security

- Outsourced databases or third-party publishing
  - data owner creates and maintains the database
  - service provider stores the database and answers queries on behalf of the database owner
  - users direct their queries to the service provider
- There are unique security challenges when the service provider is not completely trusted
  - users want a proof that query answers are complete (data haven't been deleted)
  - users want a proof that query answers are authentic (extra data haven't been added)

Database Encryption

- Parts of or the entire database can be encrypted
  - can be useful for protecting highly sensitive information
  - protects information in case of database outsourcing
- Working with encrypted databases is not easy
  - must properly distribute and manage different encryption keys
  - regular search doesn't work over encrypted contents
- Search over encrypted data is an active area of research
  - techniques that hide data well are not very efficient
  - simpler approaches leak significant amount of information about the stored data

Summary

- Database security covers several aspects
  - SQL injection attacks
  - Access control
    - discretionary, RBAC, views, stored procedures, row-level access control

- Newer topics include outsourcing, database encryption