CS 4910: Intro to Computer Security

Database Security I

Instructor: Xi Tan

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 challenges unique to database management systems (DBMSs)

Outline

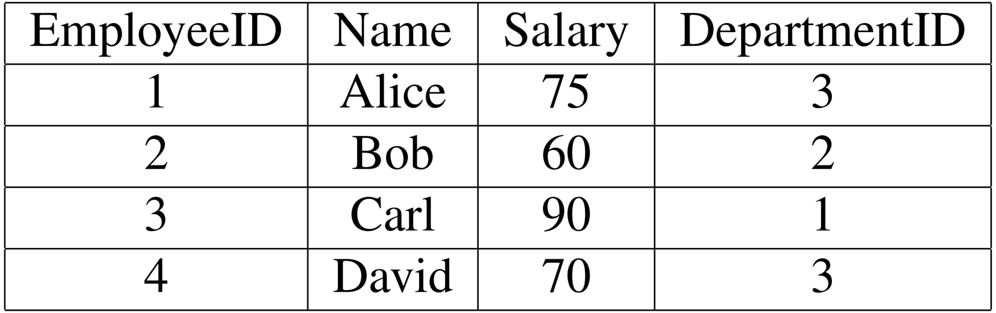
- Review of relational databases
- Database security issues
  - threats
  - access control mechanisms
- Inference in databases
- Statistical databases
- Data center security

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

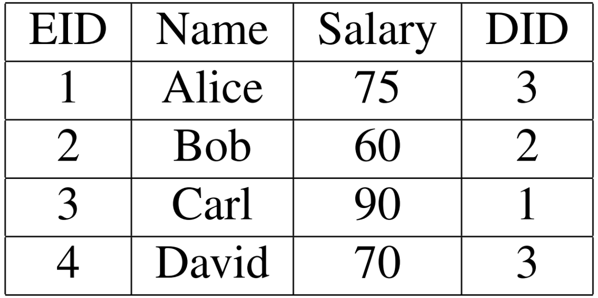


EmployeeID	Name	Salary	DepartmentID
1	Alice	75	3
2	Bob	60	2
3	Carl	90	1
4	David	70	3

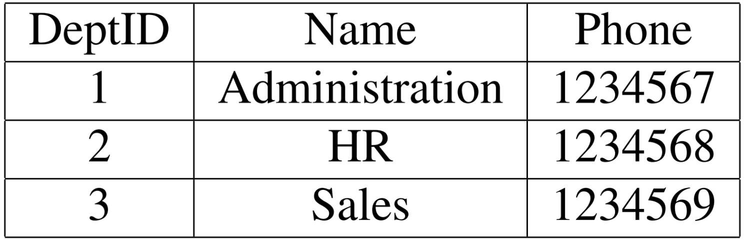
- A primary key uniquely identifies each row in a table
  - it can consist of one or more attributes
  - in the above table, Employee ID can be used as a primary key
- We create a relationship between tables by linking their attributes together
  - this is done by means of foreign keys

Relational Databases

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

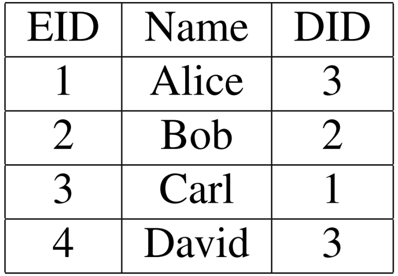


EID	Name	Salary	DID
1	Alice	75	3
2	Bob	60	2
3	Carl	90	1
4	David	70	3

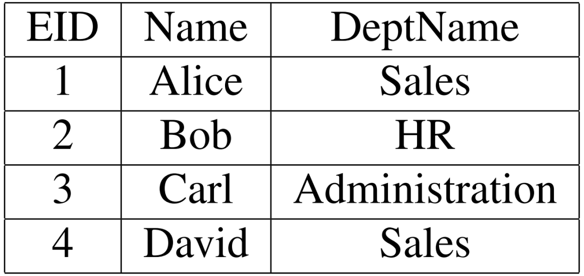


DeptID	Name	Phone
1	Administration	1234567
2	HR	1234568
3	Sales	1234569

- 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

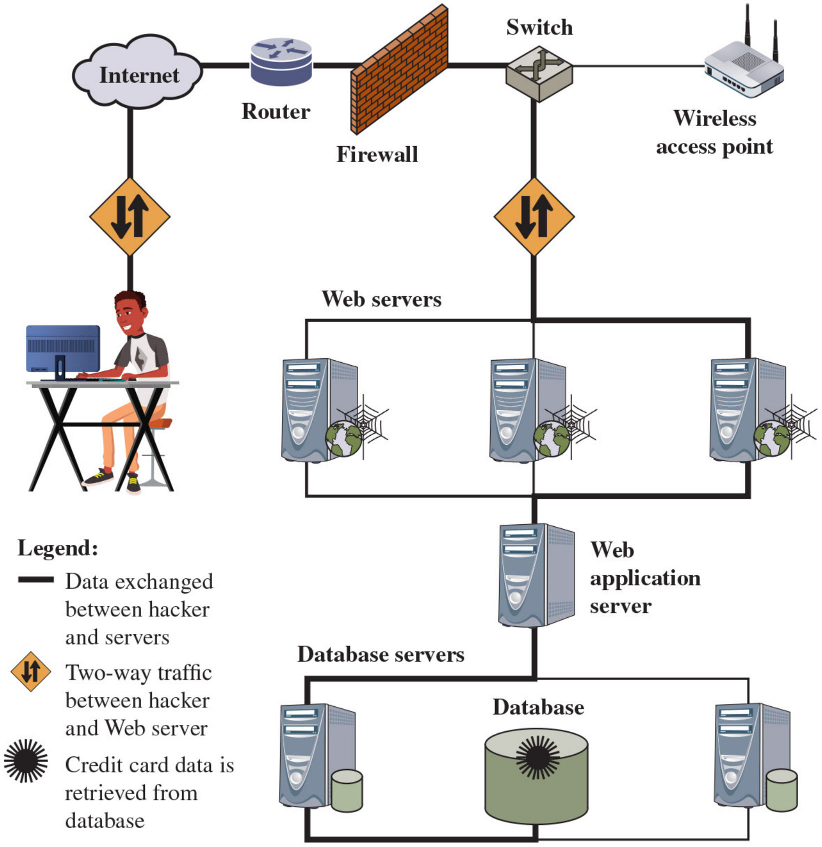
- Database security issues
  - users and authentication
    - authenticating users, assigning privileges correctly
  - secure communication between client and server
  - vulnerabilities in DBMS implementation
    - sanitizing input
    - SQL worms
    - limiting who can connect to DBMS server

SQL Injection Attacks (SQLi)

- The 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

- 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


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

SQL Injection Attacks

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 “- -” or “#”
- Subsequent text is ignored at execution time

SQL Injection Attacks

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

SQL Injection Attacks

SQLi Attack Avenues (cont.)

- 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

SQL Injection Attacks

User input

- user-supplied input is used to construct SQL request
- injection attack convinces the application to run SQL code that was not intended
- example 1.1: web application allows to query a table

```
SELECT office, building, phone
FROM employees
WHERE name = ‘$name’;
```

- now assume that the supplied input is **NOT** simply Bob

```
SELECT office, building, phone
FROM employees
WHERE name = ‘Bob’; DROP TABLE employees; --’;
```

SQL Injection Attacks

User input

- example 1.2: web application allows to query a table

```
SELECT office, building, phone
FROM employees
WHERE name= ‘$name’ and Password=‘$pwd’;
```

- now assume that the supplied input is not simply Bob

```
SELECT office, building, phone
FROM employees
WHERE name = ‘Bob’ #’ and Password=‘$pwd’;
```

SQL Injection Attacks

User input (cont.)

- example 2: web authentication mechanism that emails forgotten passwords
  - the SQL query can look like

```
SELECT somefields
FROM table
WHERE field = ‘$email’;
```

  - by manipulating the query, information about the field names, table name, and stored information can be guessed
  - e.g., the query below will give an different error if the guessed field `email` does not exist

```
SELECT somefields
FROM table
WHERE field = ‘x’ AND email IS NULL;--’;
```

SQL Injection Attacks

User input (cont.)

- example 2 (cont.)
  - after guessing field names, other information can be guessed

```
SELECT email, passwd, name
FROM members
WHERE email = ‘x’ OR name LIKE ‘%Bob%’;

SELECT email, passwd, name
FROM members
WHERE email = ‘bob@example.com’ AND passwd=‘hello1’;
```

SQL Injection Attacks

User input (cont.)

- example 2 (cont.)
  - furthermore, we can alter the table

```
SELECT email, passwd, name
FROM members
WHERE email=‘x’;
INSERT INTO members (‘email’, ‘passwd’, ‘name’,)
VALUES (‘user@buffalo.edu’, ‘pwd’, ‘Jen Smith’);--’;
```

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

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

Database Access Control

- Privileges
  - access rights: create, select, insert, update, delete, add references
  - system privilege
    - a right to perform a particular action or to perform an action on any schema object of a particular types
    - e.g., ALTER DATABASE or SELECT ANY TABLE
  - object privilege
    - a right to perform a particular action on a specific schema 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

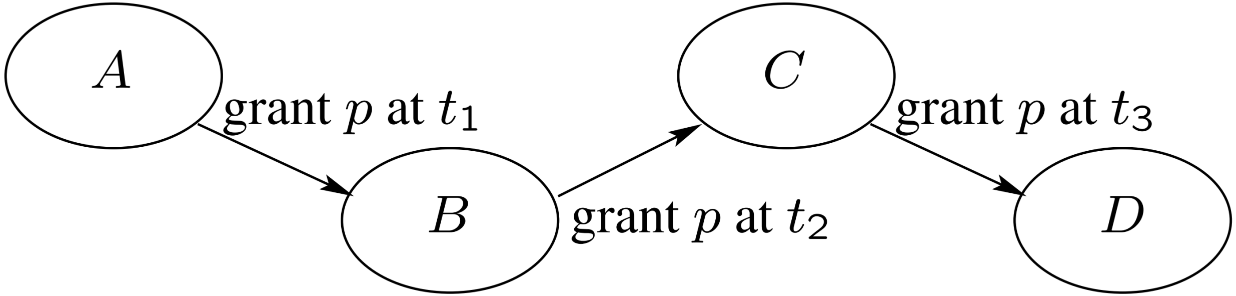
- Examples of granting and revoking privileges
  - system privileges
    - `GRANT create table TO Bob [WITH GRANT OPTION]`
    - `REVOKE create table FROM Bob`
    - users with GRANT OPTION can not only grant the privilege to others, but also revoke the privilege from any user

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

- Stored procedures
  - a stored procedure is a set of commands that are compiled into a single function
  - stored procedures can be invoked using the CALL statement
  - such procedures can allow for fine grained access control
    - some users may be permitted to access the database only by means of stored procedures
    - can precisely define access control privileges
  - the rights relevant to access control are
    - definer rights
    - invoker rights

Database Access Control

- Definer right procedures
  - a stored procedure is executed with the definer rights (i.e., owner of the routine)
  - a user requires only the privilege to execute the procedure and no privileges on the underlying objects
  - fewer privileges have to be granted to users
  - at runtime, owner’s privileges are always checked
  - a user with CREATE procedure privilege can effectively share any privilege she has without GRANT OPTION
    - create a definer right procedure and grant execute privilege to others
    - CREATE procedure privilege is very powerful

Database Access Control

- Invoker right procedures
  - a user of an invoker right procedure needs privileges on the objects that the procedure accesses
  - invoker right procedures can prevent illegal privilege sharing
    - similar to function calls in operating systems
  - invoker right procedures can be embedded with malicious code
    - e.g., the body of a stored procedure can be

```
begin
    do something useful;
    grant some privileges to the owner;
    do something useful;
end
```

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

Database Access Control

- Row-based access control can be implemented using a Virtual Private Database (VPD)
  - Oracle’s VPDs allow for fine-grained access control
  - e.g., customers can see only their own bank accounts
- How does it work?
  - a table (or view) can be protected by a VPD policy
  - when a user accesses such a table, the server invokes the policy function
  - the policy function returns a predicate, and server rewrites the query adding the predicate to the WHERE clause
  - the modified query is executed

Database Access Control

- VPD example
  - suppose Alice creates Employee table with attributes employee ID, name, and salary code
  - Alice creates a policy that an employee can access all names, but only their own salary
  - when Bob queries the table, his identity is retrieved from the session
  - if Bob queries salary from Employee table, ‘`WHERE name = Bob`’ is added to the query

So far …

- Review of relational databases
  - Primary key, foreigner key
- Database security issues
  - Threats: SQLi attack via user input
  - Access control mechanisms
    - Discretionary or role-based AC
  - Key components in DBMS access control
    - Privileges, views, stored procedures, roles, row-level access control

Next

- Review of relational databases
- Database security issues
  - threats
  - access control mechanisms
- Inference in databases
- Statistical databases
- Data center security