CSE 565 Computer Security
Fall 2018

Lecture 12: Database Security

Department of Computer Science and Engineering
University at Buffalo
We previously studied four types of access control:

- mandatory AC
- discretionary AC
- RBAC
- attribute-based AC

Many of them can be used in databases.

There are also challenges unique to database management systems (DBMSs).
• Review of relational databases

• Database security issues
  – threats
  – access control mechanisms

• Inference in databases

• Statistical databases
Relational Databases

- A database is a structured collection of data

- A database management system (DBMS) allows 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

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Salary</th>
<th>DepartmentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>75</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>70</td>
<td>3</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>EID</th>
<th>Name</th>
<th>Salary</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>75</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>70</td>
<td>3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>DeptID</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Administration</td>
<td>1234567</td>
</tr>
<tr>
<td>2</td>
<td>HR</td>
<td>1234568</td>
</tr>
<tr>
<td>3</td>
<td>Sales</td>
<td>1234569</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EID</th>
<th>Name</th>
<th>DeptName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>Sales</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>HR</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>Administration</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>Sales</td>
</tr>
</tbody>
</table>
Structured Query Language (SQL) is a widely used language that allows to manipulate databases

- **table creation**

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

- **retrieving (querying) information**

  ```sql
  SELECT EmployeeID, Name  
  FROM Employee  
  WHERE Salary >= 70
  ```
• **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 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** are among the most prevalent and dangerous types of network-based security threats
  
  – they are consistently rated among most frequent and critical Web security risks by multiple reporting agencies
  
  – an attack consists of entering maliciously crafted input on a web form
    
    • this can also include maliciously modified cookies and other variables
  
  – the entered fields are used as inputs to an SQL query
  
  – a successful attack can lead to bulk extraction of customer records, corruption of data, or execution of arbitrary commands
  
  – we’ll discuss SQL injection attacks when we talk about software security and input validation in particular
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
- **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
• **Granting and revoking privileges** (or roles) with SQL

  – granting privileges has the following syntax

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

  – revoking privileges

    ```sql
    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
  - 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
• 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 $B$

  ![Diagram]

  

• 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
• **Views**

  – access control is based on attributes (columns) and their contents

  – example: some users can see employees and their departments, but not salaries

    • 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
• **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
• **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
• 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
• 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
• **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
Inference in Databases

- Access control policy defines what information users are authorized to access

- **Inference channel** refers to obtaining access to unauthorized data by making inferences about authorized data
  - a combination of data may be more sensitive than individual items

- **Inferences within a single database**
  - certain items may be considered sensitive
  - the policy might specify that certain attributes cannot be accessed together (to remove the association between them)
• Example

  – we have Employee table for a company’s branch

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Salary</th>
<th>DepartmentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>75</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>70</td>
<td>3</td>
</tr>
</tbody>
</table>

  – the policy states that Name and Salary cannot be queried together

  – authorized views of the table

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Salary</th>
<th>DepartmentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>75</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>70</td>
<td>3</td>
</tr>
</tbody>
</table>
• **Example** (cont.)
  
  – can we make a connection between names and salaries?
  – it is trivial if the order of elements in the displayed queries is unchanged
  – what if the records are displayed in random order?
  – if narrower queries are allowed, a connection can still be made

• **Outside information** can significantly simplify making inferences
  
  – e.g., people might know that Bob works at HR department

• **How can we eliminate inference channels?**
Inferences in Databases

- Inference detection is difficult, even without assuming outside information
  - there is no general solution
  - the process is very dependent on the specifics of the database and policy
    - what data items are sensitive
    - what the security policy is
    - what functionality is desired

- Techniques that can aid in reducing the possibility of inference
  - splitting data into multiple tables
  - employing more fine-grained access control roles or procedures
Inferences across multiple databases

- often related information can be stored in different databases
- designers of individual databases cannot prevent all inference channels
- example databases
  - marriage records, voting registration, census data, etc.
- public databases can be used for unintended purposes
  - e.g., identifying patients in anonymized medical records
- making information easily accessible in digital form makes it prone to abuse
A statistical database (SDB) allows users to obtain aggregate information of statistical nature.

This can be accomplished in two ways:
- the database already contains statistical data
- the database contains information about individual data items, but answer queries of aggregate nature

A SDB can support operations such as
- count, sum, avg, max, min, etc.

The goal is to prevent a user from inferring information about individual items
- such form of inference is called a compromise
• If queries are unrestricted in a statistical database, compromising it might be easy
  – if the database size is not very big, certain queries might have
    \( \text{count}(q_i) = 1 \)
  – querying \( \text{sum}(q_i) \) reveals the actual value
  – e.g., \( \text{sum}(\text{SELECT Salary WHERE DepartmentID = 2}) = 60 \) leaks Bob’s salary

• With larger databases, a combination of queries can also compromise individual entries
• **Proposed solutions**
  – query restriction: reject queries that lead to compromise
  – perturbation: answer all queries, but modify the data

• **Types of query restrictions**
  – **minimum query size**
    • e.g., rejects all queries covering fewer than \( k \) records
    • can also specify to reject all queries covering more than \( N - k \), where \( N \) is the total number of records
    • statistics on the entire database often are still permitted
    • a compromise can still happen by querying overlapping sets
• **Types of query restrictions (cont.)**
  
  – query set overlap control
    - mandates that overlap between the current and all past queries is at most $r$
    - with enough queries, compromise is still possible
    - the method is not effective if parties can collude
    - information on both a set and its subset will not be released
    - history-based access control that require logging of all previous queries
  
  – partitioning
    - data is partitioned into groups, and only querying whole groups is allowed
The mere fact that a query is denied can leak information!

Types of data perturbation

- data swapping
  - exchange attribute values between different records
  - should be applied to many records to achieve data protection

- adding noise
  - numerical values are modified by adding a random in a range $[-t, t]$ for some fixed value $t$
  - individual values might be incorrect, but the distribution and aggregate statistics are preserved
• **Types of data perturbation** (cont.)
  
  – replacing the data with an estimation
    
    • a modified database is generated using the estimated probability distribution of the real data
    
    • the values are replaced with estimations
    
    • ordering of the elements is preserved: the smallest value is replaced with the generated smallest value

• **Finding the right level of perturbation is hard**
  
  – there is trade-off between data hiding and data accuracy
  
  – large amount of perturbation is often needed to achieve a reasonable level of hiding
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)
• 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

• Searches 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
Conclusions

- **Database security** covers several aspects
  - access control
    - discretionary, RBAC, views, stored procedures, row-level access control
  - data inference
    - within a single database, across databases, in statistical databases
- Newer topics include outsourcing, database encryption, key management