Security and Authorization

UVic C SC 370

Dr. Daniel M. German
Department of Computer Science
University of Victoria

July 9, 2003 Version: 1.1.0

Introduction

- There are 3 main objectives when designing a secure database application:
  1. **Secrecy**: Information should not be disclosed to unauthorized users.
  2. **Integrity**: Only authorized users should be allowed to modify data.
  3. **Availability**: Authorized users should not be denied access.
- To achieve them we need a **security policy**

Security Policy

- Describes the security measures that should be enforced
  - What data should be protected
  - Which users should have access to the data
- It should be clear and consistent
- In order to enforce the policy, we should use **security mechanisms** both in the DBMS an the rest of the world (access to buildings, superuser accounts, etc).
- Being able to **authenticate** a user is a fundamental requirement to be able to enforce a security policy

Access Control

- Most users need to access only a small part of the database to carry out their tasks
- The DBMS provides 2 main approaches to access control:
  1. **Discretionary** access control (DAC). Users are given access rights or **privileges** on the objects in the database that can be passed from one users to other users.
  2. **Mandatory** access control (MAC). The access rights cannot be changed by users.
- Using DAC a user could give access to sensitive information to an unauthorized user. MAC makes sure this does not happen.

Discretionary AC (DAC)

- SQL supports DAC using the GRANT and REVOKE commands

```
GRANT privileges ON object TO users [WITH GRANT
```

- For our purposes, an object is base table, a view.
- The privileges are:
  - **SELECT**: The right to access (read) all the columns (and future columns) of a table and all its rows
  - **INSERT**: The right to add rows to the table. Optionally, **INSERT(column-name)** allows to insert a row with NULL in every column except ’column-name’

Discretionary AC (cont...)

- The privileges are (cont...)
  - **DELETE**: The right to delete a row
  - **REFERENCES(column-name)**: The right to define foreign keys (in other table) that refer to the column ’column-name’
- Users who have a privilege with the **GRANT OPTION** can pass it to other users

Discretionary AC (cont...)

- A user who creates a **table** has **all** the privileges on that table, and the ability to GRANT rights to others
- A user who creates a **view**:
  - has the privileges on that view that he or she has on **every** one of the underlying views or base tables,
  - must have **SELECT** on each underlying view or table
  - has **INSERT**, **DELETE**, or **UPDATE** on each underlying table, and the view is **update-able**, the user gets the same rights on the view.

Views

- Views are an important security component
- We can use views to give access to some information while hiding some other information

```
CREATE VIEW ActiveSailors(name, age, day) AS
   SELECT S.name, S.age, R.day
   FROM Sailors S, Reserves R
   WHERE S.sid = R.sid and S.Rating > 6
```

- This view presents the name, and age of sailors who have reserved a boat, and the day of the reservation. But it does not show the sid, or bid of the reserved boat. Also, it only shows sailors with rating bigger than 6.

Schema

- Only the **owner** of the schema can execute the statements **CREATE**, **ALTER**, **DROP** to create a table, a view. etc.
- The right to execute these commands cannot be granted to others or revoked.

Authorization IDs

- Privileges are given to authorization IDs who can denote users or groups of users
- A user that connects to the DBMS must provides an authorization ID and a password

Mandatory AC (MAC)

- DAC has some weaknesses
- Users might GRANT rights to the wrong users
- A popular model for MAC is the **Bell-LaPadula** model
  - **Objects** (tables, views, ...) are accessed by **subjects** (users, groups, programs)
  - There are **security classes**. Every object belongs to a security class
  - And **clearances**. Each subject is assigned a clearance for each security class (whether the user can or cannot access the given security class)

Bell-LaPadula model...

- Security classes are organized according to a partial order, with a **most secure class** and a *least secure class*
- For example, assume four classes: *top secret (TS), secret (S), confidential (C)* and *unclassified (U)*: $TS > S > C > U$, so that class TS is more sensitive than S

Bell-LaPadula model...

- The model imposes 2 restrictions on all reads and writes to the DB objects:
- **Simple Security Property**. Subject S is allowed to read object O only if $class(S) \geq class(O)$
  - For example, a user with TS clearance can read a table with C clearance, but a user with C clearance will not be allowed to read a table in the TS class.
- It makes sense that only people with clearance equal or above can read a given object

Bell-LaPadula model...

- ***-Property**: A subject S is allowed to write object O if $class(S) \leq class(0)$.
  - For example, a user with clearance S can write only objects with S or TS classification.
- It looks strange, but the rational is: a user cannot, by mistake or willingly, write sensitive data (say TS) to a table that is low sensitive (say U class)