Outline

1. Introduction
2. Discretionary Access Control
   - Granting and Revoking Privileges
   - Trojan Horse Attack
3. Mandatory Access Control
   - The Bell-LaPadula Model
   - Multilevel Relations
4. Summary

Related Concepts

Authentication: confirming the identity of users (or programs)
Authorization*: specifying access rights to resources
Encryption: encoding data to prevent unauthorized persons from reading it (if they managed to access it)

*Our topic today.
Objectives in Securing a Database System

**Secrecy**: protection of data against unauthorized disclosure
  - e.g. a student cannot see other students’ grades

**Integrity**: prevention of unauthorized data modification
  - e.g. only the instructor may assign grades

**Availability**: ensuring authorized access is possible
  - e.g. students are not denied seeing their own grades

Access Control in a Database System

A **security policy** specifies who is authorized to do what in the system.

- A DBMS provides **access control** mechanisms to help implement a security policy.

  - Two complementary types of mechanisms:
    - Discretionary access control
    - Mandatory access control

Discretionary Access Control

**Idea**

Achieve security based on:

- privileges (certain access rights for tables, columns, etc.), and
- a mechanism for granting and revoking such privileges at a user’s own discretion

**Authorization administration policy**: specifies how granting/revoking of privileges is organized (i.e., who may grant and revoke)

- Centralized administration: only some privileged users
- Ownership-based administration: creator of the object

**Administration delegation**: If authorized to do so, a user may assign other users the right to grant or revoke.

In SQL-92, privileges are given to users.
In SQL:1999, privileges are given to roles; those are assigned to users.
Granting and Revoking Privileges in SQL

**GRANT privileges ON object TO users [WITH GRANT OPTION]**

- Possible privileges:
  - SELECT
  - INSERT(column)
  - UPDATE(column)
  - DELETE
  - REFERENCES(column)
- **WITH GRANT OPTION** allows user to pass on privilege (with or without passing on grant option)

**REVOKE [GRANT OPTION FOR] privileges ON object FROM users [RESTRICT | CASCADE]**

- When a privilege is revoked from user X, it is also revoked from all users that were granted the privilege solely from X

---

**Trojan Horse Attack**

- Suppose user Bob has privileges to read a secret table T.
- User Malory wants to see the data in T (but does not have the privileges to do so).

1. Malory creates a table T' and gives INSERT privileges to Bob.
2. Malory tricks Bob into copying data from T to T' (e.g., by extending the "functionality" of a program used by Bob).
3. Malory can then see the data that comes from T.

---

**Mandatory Access Control**

**Idea**

*Achieve security based on system-wide policies that cannot be changed by individual users.*
The Bell-LaPadula Model

- Basis: a partially ordered set of security classes
  - Example: TopSecret > Secret > Confidential > Unclassified

- DB objects (e.g. tables, rows, columns) are assigned such a class
- Subjects (users, programs) are assigned clearance for such a class

- Goal: Information should never flow from a higher to a lower class.

- Restrictions enforced by the DBMS:
  1. Subject $S$ can read object $O$ only if $\text{clearance}(S) \geq \text{class}(O)$
  2. Subject $S$ can write object $O$ only if $\text{clearance}(S) \leq \text{class}(O)$

Trojan Horse Attack Revisited

- Suppose user Bob has privileges to read a secret table $T$.
  - $\text{clearance}(Bob) = \text{Secret}$
- User Malory wants to see the data in $T$ (but does not have the privileges to do so).
  - $\text{clearance}(Malory) < \text{Secret}$
- Malory creates a table $T'$ and gives INSERT privileges to Bob.
  - $\text{class}(T') = \text{clearance}(Malory)$
  - i.e. $\text{class}(T') < \text{Secret}$
- Malory tricks Bob into copying data from $T$ to $T'$.
  - Writing to $T'$ fails for Bob because $\text{clearance}(Bob) \not> \text{class}(T')$
- Malory can then see the data that comes from $T$.

Multilevel Relations

- Individual tuples or columns can be assigned security classes
  - users with different clearances see different tables

- Example:

<table>
<thead>
<tr>
<th>EID</th>
<th>PID</th>
<th>EmpRole</th>
<th>Security Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>886</td>
<td>Manager</td>
<td>Unclassified</td>
</tr>
<tr>
<td>2</td>
<td>881</td>
<td>Researcher</td>
<td>TopSecret</td>
</tr>
</tbody>
</table>

- Users with clearance TopSecret see two rows;
- other users see only one.

- To avoid revealing any information, the Security Class attribute must be treated as part of the primary key.
Summary

- Three main security objectives:
  - Secrecy
  - Integrity
  - Availability

- Discretionary access control
  - Based on notion of privileges
  - GRANT and REVOKE
  - Susceptible to trojan horse attack

- Mandatory access control
  - Based on notion of security classes
  - Not widely supported