**CSC 742**  
Database Management Systems

**Topic #11: Database Security**

---

**Security Goals**

- **Confidentiality:**  
  - prevent/deter/detect unauthorized access to information.

- **Integrity:**  
  - prevent/deter/detect unauthorized modification of information.

- **Availability:**  
  - prevent/deter/detect unauthorized denial of service.

---

**Discretionary Access Control**

- **Discretionary Access Control (DAC)**  
  - Allow access rights to be propagated from one subject to another.
  - Possession of an access right by a subject is sufficient to allow access to the object.

---

**Security Mechanisms**

- **Methods to achieve the security goals.**
  - Access control
  - Authentication
  - Encryption
  - Intrusion detection
  - Inference control
  - …

---

**DAC in DBMS**

- **Based on Granting and Revoking of privileges.**

- **Types of Discretionary Privileges**
  - **Account level privileges**
    - Independent of database content
    - Example: 
      - GRANT CREATETAB TO Alice;
  - **Relation level privileges**  
    - Based on Access Matrix Model  
    - Related to the database content
Access Matrix Model

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Ri</td>
</tr>
<tr>
<td>X</td>
<td>Rj</td>
</tr>
</tbody>
</table>

View

- View mechanism
  - Restrict access only to selected attributes and tuples.
- Example:
  
  ```
  CREATE VIEW Researchers AS
  SELECT Name, Bdate, Address
  FROM Employee
  WHERE Department='Research'
  GRANT SELECT ON Researchers TO Bob
  ```

DAC in DBMS (Cont’d)

- Relation level privileges
  - Each relation is assigned an owner account.
  - The owner of a relation can give privileges on the relation to other users (grant).
  - The owner can take back privileges (revoke).

Inherent Weakness of DAC

- Unrestricted DAC allows unexpected information flow which violates security policy.
- The user can be trusted not to do this deliberately. However, it is still possible for Trojan Horse Programs to do so.
  - A Trojan horse does what a user expects it to do, but in addition exploits the user’s legitimate privilege to cause a security breach.

Examples

- `GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO Alice`
- `GRANT SELECT ON EMPLOYEE TO Bob WITH GRANT OPTION`
- `REVOKE SELECT ON EMPLOYEE FROM Bob`
- `GRANT SELECT ON EMPLOYEE(SALARY) TO Bob`

Trojan Horse Example

User Alice executes

```
Program Goodies

Relation R

Alice: r

Relation S

Bob: r

Alice: w
```

The ACLs do not allow B to read R. But B can read the information with the help of the Trojan Horse.
Mandatory Access Control

- Basic idea:
  - put restrictions on access rights.
  - Label both the subjects and the objects.
  - Allow a subject to access an object only when certain constraints are satisfied.

MAC (Cont’d)

- Bell LaPadula (BLP) Model
  - *Simple security:* Subject S can read object O only if
    - Label(S) dominates label(O).
    - Information can flow from label(O) to label(S)
  - *Star property:* Subjects can write object O only if
    - Label(O) dominates label(S)
    - Information can flow from label(S) to label(O).
    - Intuitively, *no read up*

MAC in DBMS

- Attribute values and tuples are considered as objects.
  - Each attribute A is associated with a classification attribute C
  - In some models, a tuple classification attribute TC is added to the relation.
  - Example:
    - Employee (SSN, Name, BDate, Salary) →
    - Employee (SSN, CSSN, Name, CName, BDate, CDate, Salary, CSalary, TC)
    - Such a relation is called a *multi-level* relation.

BLP Model

- Dominance: Top secret > Secret > Confidential > Unclassified
- Can-flow: Top secret > Secret > Confidential > Unclassified

MAC in DBMS (Cont’d)

- Employee (SSN, CSSN, Name, CName, BDate, CDate, Salary, CSalary, TC)
- Apparent key:
  - The set of attributes that would have formed the primary key in a regular (single-level) relation.
Polyinstantiation

Several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Performance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>40000 C</td>
<td>Fair</td>
<td>S</td>
</tr>
<tr>
<td>22222222 C Brown C</td>
<td>80000 S</td>
<td>Good</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Employee (What class C users’ see)

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Performance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>40000 C</td>
<td>Null</td>
<td>C</td>
</tr>
<tr>
<td>22222222 C Brown C</td>
<td>Null C</td>
<td>Good</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Class C user:

```
UPDATE Employee
SET Performance = 'Excellent'
WHERE SSN='111111111'
```

Employee (What class U users’ see)

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Performance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>Null U</td>
<td>Null</td>
<td>U</td>
</tr>
</tbody>
</table>

Integrity Constraints for Multi-level relations

- **Entity integrity**
  - All attributes that are members of the apparent key must not be null and must have the same security class.
  - All other attribute values in the tuple must have a security class greater than or equal to that of the apparent key.

- **Null integrity**
  - If a tuple value at some security level can be derived from a higher-level tuple, then it’s sufficient to store the higher-level tuple.

Is this possible?

Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
<th>Performance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>50000 U</td>
<td>Excellent</td>
<td>U</td>
</tr>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>40000 C</td>
<td>Good</td>
<td>C</td>
</tr>
<tr>
<td>11111111 U</td>
<td>Smith U</td>
<td>40000 C</td>
<td>Fair</td>
<td>S</td>
</tr>
<tr>
<td>22222222 C Brown C</td>
<td>80000 S</td>
<td>Good</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>