CSC 474/574
Information Systems Security

Topic 7.1: DAC and MAC in Databases

Outline

• DAC in DBMS
  – Grant and revoke
  – View
• MAC in DBMS
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
    - Our focus

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 privileges may be further propagated.
  - The owner can take back privileges (revoke).
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

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
MAC in DBMS

- Attribute values and tuples are considered as objects.
  - Each attribute A is associated with a classification attribute C (the label)
  - In some models, a tuple classification attribute TC is added to the relation.
  - Example:
    - Employee (SSN, Name, Salary, Performance) →
    - Employee (SSN, CSSN, Name, CName, Salary, CSalary, Performance, CPerformance, TC)
    - Such a relation is called a multi-level relation.
MAC in DBMS (Cont’d)

- Employee \((SSN, C_{SSN}, Name, C_{Name}, BDate, C_{BDate}, Salary, C_{Salary}, TC)\)

- Primary key:
  - The set of attributes that can uniquely identify each tuple.

- 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.

<table>
<thead>
<tr>
<th>Mission</th>
<th>Target</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack S</td>
<td>Mars S</td>
<td>S</td>
</tr>
<tr>
<td>Explore U</td>
<td>Moon C</td>
<td>C</td>
</tr>
<tr>
<td>Explore C</td>
<td>Mars S</td>
<td>S</td>
</tr>
</tbody>
</table>
Is this possible?

<table>
<thead>
<tr>
<th>ShipID</th>
<th>C_S</th>
<th>Mission</th>
<th>C_M</th>
<th>Target</th>
<th>C_T</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>S</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Explore</td>
<td>U</td>
<td>Moon</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Enterprise</td>
<td>C</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

What could be the real key?

What if?

<table>
<thead>
<tr>
<th>ShipID</th>
<th>C_S</th>
<th>Mission</th>
<th>C_M</th>
<th>Target</th>
<th>C_T</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>S</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Explore</td>
<td>U</td>
<td>Moon</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Enterprise</td>
<td>C</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

What could be the real key?
UPDATE Mission
SET Mission = ‘Explore’, Target = ‘Moon’
WHERE ShipID = ‘Voyager’

Class C user sees

<table>
<thead>
<tr>
<th>ShipID</th>
<th>C_S</th>
<th>Mission</th>
<th>C_M</th>
<th>Target</th>
<th>C_T</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>S</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Enterprise</td>
<td>C</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

Class C user:

UPDATE Mission
SET Mission = ‘Explore’, Target = ‘Moon’
WHERE ShipID = ‘Voyager’

After Update

Mission

<table>
<thead>
<tr>
<th>ShipID</th>
<th>C_S</th>
<th>Mission</th>
<th>C_M</th>
<th>Target</th>
<th>C_T</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>S</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Enterprise</td>
<td>C</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

What should be returned to a class C user?
How about a class S user?
What is the general method?
Mission

<table>
<thead>
<tr>
<th>ShipID</th>
<th>CS</th>
<th>Mission</th>
<th>CM</th>
<th>Target</th>
<th>CT</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>S</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Attack</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Voyager</td>
<td>U</td>
<td>Explore</td>
<td>U</td>
<td>Moon</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Enterprise</td>
<td>C</td>
<td>Explore</td>
<td>C</td>
<td>Mars</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

What to return to Class C user?
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
  – Purpose: make the retrieved information meaningful.

• 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.
  – Purpose: Reduce redundancy