CSC12001

Data Security in Information Systems

C03 - Access Control - Basic Concepts

Dr. Phạm Thị Bạch Huệ
MSc. Lương Vĩ Minh

Information System Department – Faculty of Information Technology
Univeristy of Science, VNU-HCM

@ 2020 – University of Science – Vietnam National University HCM

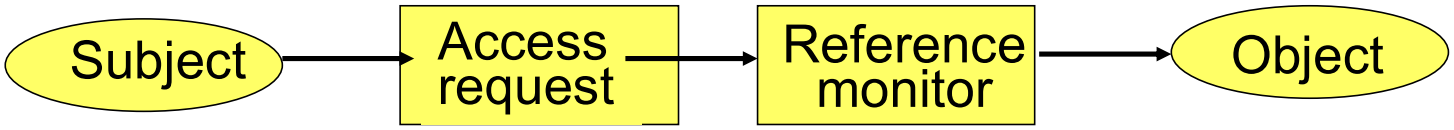
Outline

1. Basic concepts
2. DAC and MAC
3. DAC
4. Content-based access control
5. RBAC

Access Control - basic concepts

- An access control system regulates the operations that can be executed on data and resources to be protected
- Its goal is to control operations executed by subjects in order to prevent actions that could damage data and resources
- Access control is typically provided as part of the operating system and of the database management system (DBMS)

Access Control - basic concepts



- The very nature of access control suggests that there is an *active* subject *requiring access* to a passive *object* to perform some specific *access operation.*
- A *reference monitor* grants or denies access

Access Control Mechanism

- It is typically a software system implementing the access control function
- It is usually part of other systems
- The access control mechanism uses some access control policies to decide whether to grant or deny a subject access to a requested resource

Object

- Anything that holds data, such as relations, directories, interprocess messages, network packets, I/O devices, or physical media
- We often refer to objects, controlled by the access control system, as *protection objects*
- Note that not all resources managed by a system need to be protected

Subject

- An abstraction of any active entity that performs computation in the system
- Subjects can be classified into:
  - *users* -- single individuals connecting to the system
  - *groups* -- sets of users
  - *roles* -- named collections of privileges / functional entities within the organization
  - *processes* -- executing programs on behalf of users
- Relations may exist among the various types of subject

Access Operations - Access Modes

- Operations that a subject can exercise on the protected objects in the system
- Each type of operation corresponds to an *access mode*
- The basic idea is that several different types of operation may be executed on a given type of object; the access control system must be able to control the specific type of operation
- The most simple example of access modes is:
  - read look at the contents of an object
  - write change the contents of an object
- In reality, there is a large variety of access modes
- The access modes supported by an access control mechanism depend on the resources to be protected (read, write, execute, select, insert, update, delete, …)
- Often an access control system uses modes with the same name for different types of object; the same mode can correspond to different operations when applied to different objects

Access Operations - Access Modes
An example

- Unix operating system
  - Access modes defined for files
    - read: reading from a file
    - write: writing to a file
    - execute: executing a (program) file
  - Access modes defined for directories
    - read: list a directory contents
    - write: create or rename a file in a directory
    - execute: search a directory

Access Operations - Access Modes
An example

- Database management systems
  - Access modes defined for tables
    - Read: reading from a table (Select statement)
    - Write: writing to a table (Insert, Update, Delete)

Access Operations

Access Permissions and Attributes

- How does the reference monitor decides whether to give access or not?
- Main approaches:
  - It uses *access permissions*
    - Typical of discretionary access control (DAC) models
  - It uses information (often referred to as *attributes*) concerning subjects and objects
    - Typical of mandatory access control (MAC) models
- More innovative approaches have been developed where access permissions can be also expressed in terms of object and subject attributes and even context parameters

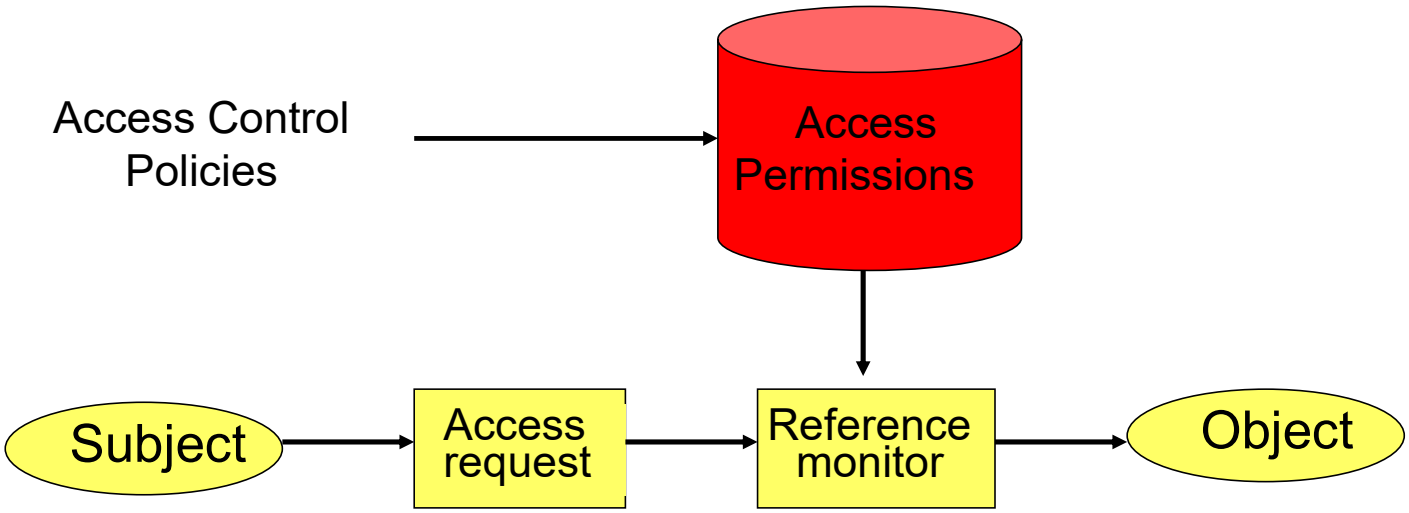
DAC and MAC

Two main categories:

- Discretionary Access Control Models (DAC)
  - Definition If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a *discretionary access control (DAC*), also called an *identity-based access control* (*IBAC*).
- Mandatory Access Control Models (MAC)
  - Definition When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a *mandatory access control* (MAC), occasionally called a *rule-based access control*.

Access Operations

Access Permissions



Access Permissions

- Access permissions, also called *authorizations*, are expressed in terms of subjects, objects, and access modes
- From a conceptual point of view an access permission is a tuple <*s, o, a*> where
  - *s* is a subject
  - *o* is an object
  - *a* is an access mode

  It states that subject *s* has the permission to execute operation *a* on object *o*

  We also say that *s* has access right *a* on object *o*
- Example: the access permission <Bob, Read, F1> states that Bob has the permission to read file F1

Ownership and Administration

- A key question when dealing with access control is who specifies which subjects can access which objects for which operations
- In the case of permissions, this means specifying which are the subjects that can enter permissions

Ownership and Administration
Two basic options

- *Discretionary* approach
  - the owner of a resource decrees who is allowed to have access
  - But then: who is the owner of a resource?
- *Mandatory* approach
  - a system-wide policy decrees who is allowed to have access

Access Control Structures

The most well known access control structures for DAC models are based on the notion of *Access Control Matrix.* Let:

- *S* be a set of subjects
- *O* be a set of objects
- *A* be a set of access modes

An access control matrix *M* on *S, O,* and *A* is defined as

$$M = (M_{so})_{s \in S,\, o \in O} \text{ with } M_{so} \subset A$$

The entry M_{so} specifies the set of access operations subject *s* can perform on object *o.*

Access Control Structures Example

	bill.doc	**edit.exe**	**fun.dir**
Alice	-	{execute}	{execute, read}
Bill	{read, write}	-	{execute, read, write}

Access Control Structures

Access Control Lists and Capabilities

- Directly implementing access control matrices is quite inefficient, because in most cases these matrices are sparse
- Therefore two main implementations have been developed
  - Access control lists
    - Used in DBMS and Operating Systems
  - Capabilities

DAC

- DAC policies govern the access of subjects to objects on the basis of subjects' identity, objects' identity and permissions
- When an access request is submitted to the system, the access control mechanism verifies whether there is a permission authorizing the access
- Such mechanisms are discretionary in that they allow subjects to grant other subjects authorization to access their objects at their discretion

Basic Operations in Access Control

- *Grant* permissions
  - Inserting values in the matrix's entries
- *Revoke* permissions
  - Remove values from the matrix's entries
- *Deny permissions*
  - *Prevent a subject s operating a privilege on an object o*

Grant operation

GRANT *PrivilegeList*| ALL[PRIVILEGES]
ON *Relation* | *View*
TO *UserList* | PUBLIC
[WITH GRANT OPTION]

- it is possible to grant privileges on both relations and views
- privileges apply to entire relations (or views)
- for the update privilege, one needs to specify the columns to which it applies

Grant operation - example

Bob: GRANT select, insert ON Employee TO Ann
WITH GRANT OPTION;
Bob: GRANT select ON Employee TO Jim
WITH GRANT OPTION;
Ann: GRANT select, insert ON Employee TO Jim;

- Jim has the select privilege (received from both Bob and Ann) and the insert privilege (received from Ann)
- Jim can grant to other users the select privilege (because it has received it with grant option); however, he cannot grant the insert privilege

Grant operation

- The authorization catalogs keep track for each users of the privileges the user possesses and of the ones that the user can delegate
- Whenever a user u executes a Grant operation, the system intersects the delegable privileges of u with the set of privileges specified in the command
- If the intersection is empty, the command is not executed

Grant operation - example

Bob: GRANT select, insert ON Employee TO Jim WITH GRANT OPTION;
Bob: GRANT select ON Employee TO Ann WITH GRANT OPTION;
Bob: GRANT insert ON Employee TO Ann;
Jim: GRANT update ON Employee TO Tim WITH GRANT OPTION;
Ann: GRANT select, insert ON Employee TO Tim;

Grant operation - example

- The first three GRANT commands are fully executed (Bob is the owner of the table)
- The fourth command is not executed, because Jim does not have the update privilege on the table
- The fifth command is partially executed; Ann has the select and insert but she does not have the grant option for the insert --> Tim only receives the select privilege

Revoke operation

REVOKE *PrivilegeList*| ALL[PRIVILEGES]
ON *Relation* | *View*
FROM *UserList* | PUBLIC

- a user can only revoke the privileges he/she has granted; it is not possible to only revoke the grant option
- upon execution of a revoke operation, the user from whom the privileges have been revoked looses these privileges, unless has them from some source *independent* from that that has executed the revoke

Revoke operation - example

Bob: GRANT select ON Employee TO Jim WITH GRANT OPTION;
Bob: GRANT select ON Employee TO Ann WITH GRANT OPTION;
Jim: GRANT select ON Employee TO Tim;
Ann: GRANT select ON Employee TO Tim;
Jim: REVOKE select ON Employee FROM Tim;

- Tim continues to hold the select privilege on table Employee after the revoke operation, since he has independently obtained such privilege from Ann.

Revoke operations

- Recursive revocation: whenever a user revokes an authorization on a table from another user, all the authorizations that the revokee had granted because of the revoked authorization are removed
- The revocation is iteratively applied to all the subjects that received the access authorization from the revokee

Views and content-based authorization

- Views are a mechanism commonly used to support content-based access control in RDBMS
- Content-based access authorizations should be specified in terms of predicates
- Only the tuples of a relation verifying a given predicate are considered as the protected objects of the authorization

Views and content-based authorization

- The approach to support content-based access control in RDBMS can be summarized as follows:
  - Define a view containing the predicates to select the tuples to be returned to a given subject S
  - Grant S the select privilge on the view, and not on the underlying table

Views and content-based authorization

- Example: suppose we want authorize user Ann to access only the employees whose salary is lower than 20000.
- Steps:
  - CREATE VIEW Vemp AS
    SELECT * FROM Employee
    WHERE Salary < 20000;
  - GRANT Select ON Vemp TO Ann;

Views and content-based authorization

- Queries against views are transformed through the *view composition* in queries against base tables
- The view composition operation combines in AND the predicates specified in the query on the view with the predicates which are part of the view definition

Views and content-based authorization

Ann: SELECT * FROM Vemp
WHERE Job = 'Programmer';

Query after view composition:

SELECT * FROM Employee
WHERE Salary < 20000 AND
Job = 'Programmer';

Steps in Query Processing

- Parsing
- Catalog lookup
- Authorization checking
- View Composition
- Query optimization

Note that authorization is performed before view composition; therefore, authorization checking is against the views used in the query and not against the base tables used in these views

Views and content-based authorization

- Views can also be useful to grant select privileges on specific columns: we only need to define a view as projection on the columns on which we want to give privileges
- Views can also be used to grant privileges on simple statistics calculated on data (such as AVG, SUM,..)

Authorizations on views

- The user creating a view is called the *view definer*
- The privileges that the view definer gets on the view depend from:
  - The view semantics, that is, its definition in terms of the base relation(s)
  - The authorizations that the definers has on the base table(s)

Authorizations on views

- The view definer does not receive privileges corresponding to operations that cannot be executed on the view
- For example, **alter** and **index** do not apply to views

Authorizations on views

- Consider the following view

  Bob: CREATE VIEW V1 (Emp#, Total_Sal)
  AS SELECT Emp#, Salary + Bonus
  FROM Employee WHERE
  Job ='Programmer';

  The update operation is not defined on column Total_Sal of the view; therefore, Bob will not receive the update authorization on such column

Authorizations on views

- Basically, to determine the privileges that the view definer has on the view, the system needs to *intersect the set of privileges that the view definer has on the base tables with the set of privileges corresponding to the operations that can be performed on the view*

Authorizations on views - example

- Consider relation Employee and assume Bob is the creator of Employee
- Consider the following sequence of commands:
  - Bob: GRANT Select, Insert, Update ON Employee to Tim;
  - Tim: CREATE VIEW V1 AS SELECT Emp#, Salary FROM Employee;
  - Tim: CREATE VIEW V2 (Emp#, Annual_Salary) AS SELECT Emp#, Salary*12 FROM Employee;

Authorizations on views - example

- Tim can exercise on V1 all privileges he has on relation Employee, that is, Select, Insert, Update
- By contrast, Tim can exercise on V2 only the privileges of Select and Update on column Emp#;

Authorizations on views

- It is possible to grant authorizations on a view: the privileges that a user can grant are those that he/she owns with grant option on the base tables
- Example: user Tim cannot grant any authorization on views V1 and V2 he has defined, because he does not have the authorizations with grant option on the base table

Authorizations on views - example

- Consider the following sequence of commands:
  - Bob: GRANT Select ON Employee TO Tim WITH GRANT OPTION;
  - Bob: GRANT Update, Insert ON Employee TO Tim;
  - Tim: CREATE VIEW V4 AS SELECT Emp#, Salary FROM Employee;

  Authorizations of Tim on V4:

  - Select with Grant Option;
  - Update, Insert without Grant Option;

DAC

- Advantages:
  - Flexibility in terms of policy specification
  - Supported by all OS and DBMS
- Drawbacks:
  - No information flow control (suffered from Trojan Horses attacks)

Access Control in Commercial DBMSs

- Most of the commercial DBMSs also support RBAC features.
- RBAC (Role-based Access Control)

RBAC – SQL Commands

- CREATE ROLE role-name IDENTIFIED BY passw |NOT IDENTIFIED;

  Example:

  CREATE ROLE teller IDENTIFIED BY cashflow;

- DROP ROLE role-name;

RBAC – SQL Commands

- GRANT role TO user | role | PUBLIC [WITH ADMIN OPTION];
  to perform the grant of a role, a user must have the privilege for the role with the ADMIN option, or the system privilege GRANT ANY ROLE
  The ADMIN option allows the receiver to modify or drop the role
- Example:

  GRANT teller TO Bob;

RBAC – SQL Commands

- The grant command for authorization granting can have roles as subjects

  Example:

  ```
  GRANT select ON Employee TO teller;
  ```

RBAC – SQL Commands

- SET ROLE role-name IDENTIFIED BY passwd;
  The set command is used enable and disable roles during sessions
  Example: SET ROLE teller IDENTIFIED by cashflow;
- SET ROLE ALL [EXCEPT role-name]
  it can only be used for roles not requiring passwords
  SET ROLE ALL; SET ROLE ALL EXCEPT banker;
- SET ROLE NONE;
  It disables roles for the current session

- MAC (Mandatory Access Control)

Bell and LaPadula Model

- Subjects are assigned clearance levels and they can operate at a level up to and including their clearance levels
- Objects are assigned sensitivity levels
- The clearance levels as well as the sensitivity levels are called access classes

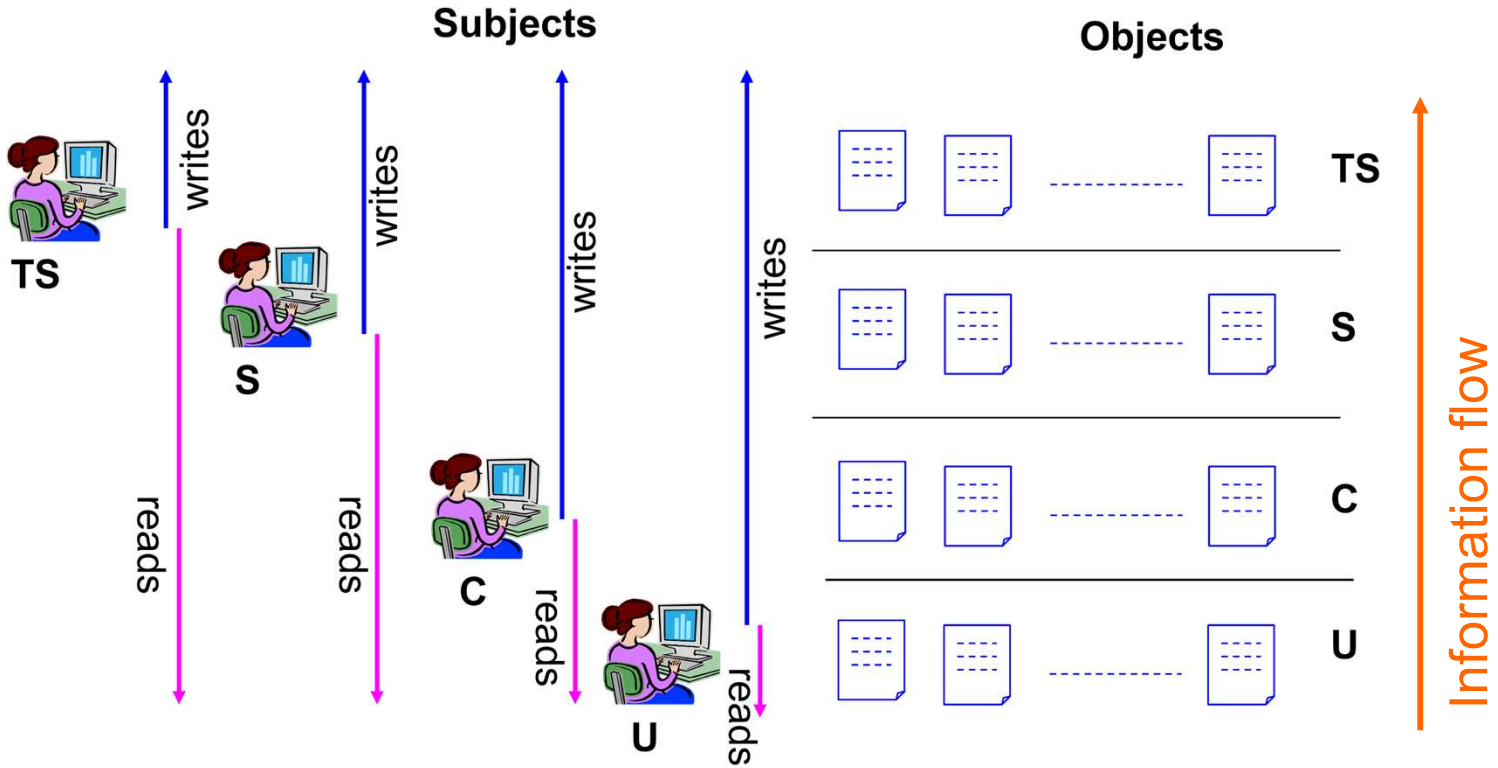
MAC

- **Object:** tables.
- **Subject:** users.

- **Security class (or level, or labels)**
  - ✓ Top Secret (TS), Secret (S), Confidential (C), Unclassified (U)
  - ✓ Trong đó: TS > S > C > U

- Rules:
  - ✓ No read – up: S can read O if and only if Class(S) >= Class(O).
  - ✓ No write – down: S can write O if Class(S) <= Class(O).
- However, in reality, write-up is not allowed but write to O at the same class. Please investigate in Oracle?

MAC


Subjects
Objects
TS
S
C
U
writes
reads
TS
S
C
U
Information flow

- Oracle's implementation of MAC: OLS-Oracle Label Security.

- **Q & A**