Access Control Models

Dr. Natarajan Meghanathan
Associate Professor of Computer Science
Jackson State University
E-mail: natarajan.meghanathan@jsums.edu

Access Control Models

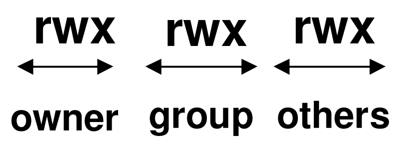
- Access Control – to regulate the actions of the subjects on the objects
- Discretionary Access Control (DAC) Model: The DAC model gives the owner of the object the privilege to grant or revoke access to other subjects.
- Mandatory Access Control (MAC) Model: The MAC model is enforced by the system administrator (rather than DAC approach of the individual subjects granting privileges to other subjects) that determines whether or not a specific subject has a particular access to a specific object.
- The MAC model is implemented through a multi-level security (MLS) scheme according to which each object has an associated classification (i.e., security level) and each subject is also associated with a security clearance. MLS allows subjects with higher security clearance to easily access objects with equal or lower authorization level.
- Dynamic Access Control Model (e.g.,Chinese Wall Model): It dynamically establishes the access rights of a user based on what the user has already accessed during the current login session.
- Role-based Access Control (RBAC) Model: A mapping is maintained between the different roles the subjects can take in a system (typically used in DBMS) and the specific objects. A separate mapping is maintained between the subjects and the different roles they can take.

Discretionary Access Control (DAC)

- In systems that employ discretionary access controls, the owner of an object can decide which other subjects may have access to the object and what specific access they may have.
- One common method to accomplish this is via the permission bits used in UNIX-based systems.
  - The owner of a fie can specify what permissions (read/write/execute) members in the same group may have and also what permissions all others may have.

UNIX File Permissions

- Each object (file) has access rights set for the three classes:
  - owner, the group of the owner and others.
- The owner uses the `chmod` command to set the access rights of a file and can use the `chown` command to change the owner or group of a file.
- The Access rights are: Read (r – 4), Write (w – 2) and Execute (x – 1), nothing (0).
- Each file has associated permissions of the form



- If a file has to be given more than one access right to a class, we have to add their corresponding values.
- Example:
  - Consider a file A.txt. To set read, write and execute permissions to its owner, read and execute only permission to the group and read-only permission to others, use the chmod command as follows:
    - chmod 754 A.txt
  - To set read, write and execute permissions to the owner, read and write permission to the group and no permission to others, use the chmod command as follows:
    - chmod 760 A.txt

UNIX Folder Permissions

- Folders also have permissions.
  - Read permissions for a folder allows a user to list the contents of the folder.
  - Write permissions for a folder allows a user to create new files in that folder
  - Execute permissions to a folder allows the user to change the current working directory to that one.
  - Note that UNIX-systems employ a path based approach for file access control.
    - Example: If the user Bob’s current working directory is /home/user/bob and wishes to access (say, read) a file readme.txt located at /home/user/alice/readme.txt; then, Bob should have execute permissions to the folder ‘alice’ and read permissions to the readme.txt file.

Disadvantages of DAC

- <u>The DAC model has two primary disadvantages</u>:
  - Since it is left to the discretion of the owner of an object do decide about giving access permissions, the users may have a different perception/ concept of security which may or may not be in line with the enterprise policy. Hence, <u>it becomes difficult to enforce uniform security throughout the enterprise policy</u>. Technically, the enterprise is the owner of all data.

  - <u>Revocations of access permissions granter earlier may not have the intended effect</u>. For example, assume a subject S1 has granted the Update access on object O to subject S2, which after a while grants subject S3 the access to object O. Later, if S1 has granted the Update access on object O to S3. Now, even if one of the two subjects (S1 or S2) revoke their access permission on object O to S3, S3 can still access object O by virtue of the grant permission granted by the subject (S1 or S2) which has not yet revoked the same.

Chinese Wall Model

- <u>Real World Analogy:</u>
- Let there be four companies: Oil-A, Oil-B and Car-A, Car-B. Companies Oil-A and Oil-B belong to the class of “Oil” companies and companies Car-A and Car-B belong to the class of “Car” companies.
- The two oil companies Oil-A and Oil-B have conflicts of interest; but, they do not have any conflict of interest with the two car companies. Similarly, the two car companies Car-A and Car-B have conflicts of interest; but, they do not have any conflict of interest with the two oil companies.
- Initially, a user is free to find a job in any of these four companies.
- But, once say a user joins Oil-A, he is not supposed to simultaneously work in Oil-B.
- However, if the user can handle multiple jobs, he could join either Car-A or Car-B. But, once he joins one of these two car companies, he cannot work in the other car company.
- In other words, in the above example, a user can at most simultaneously work with one of the two oil companies and one of the two companies.

Chinese Wall Model

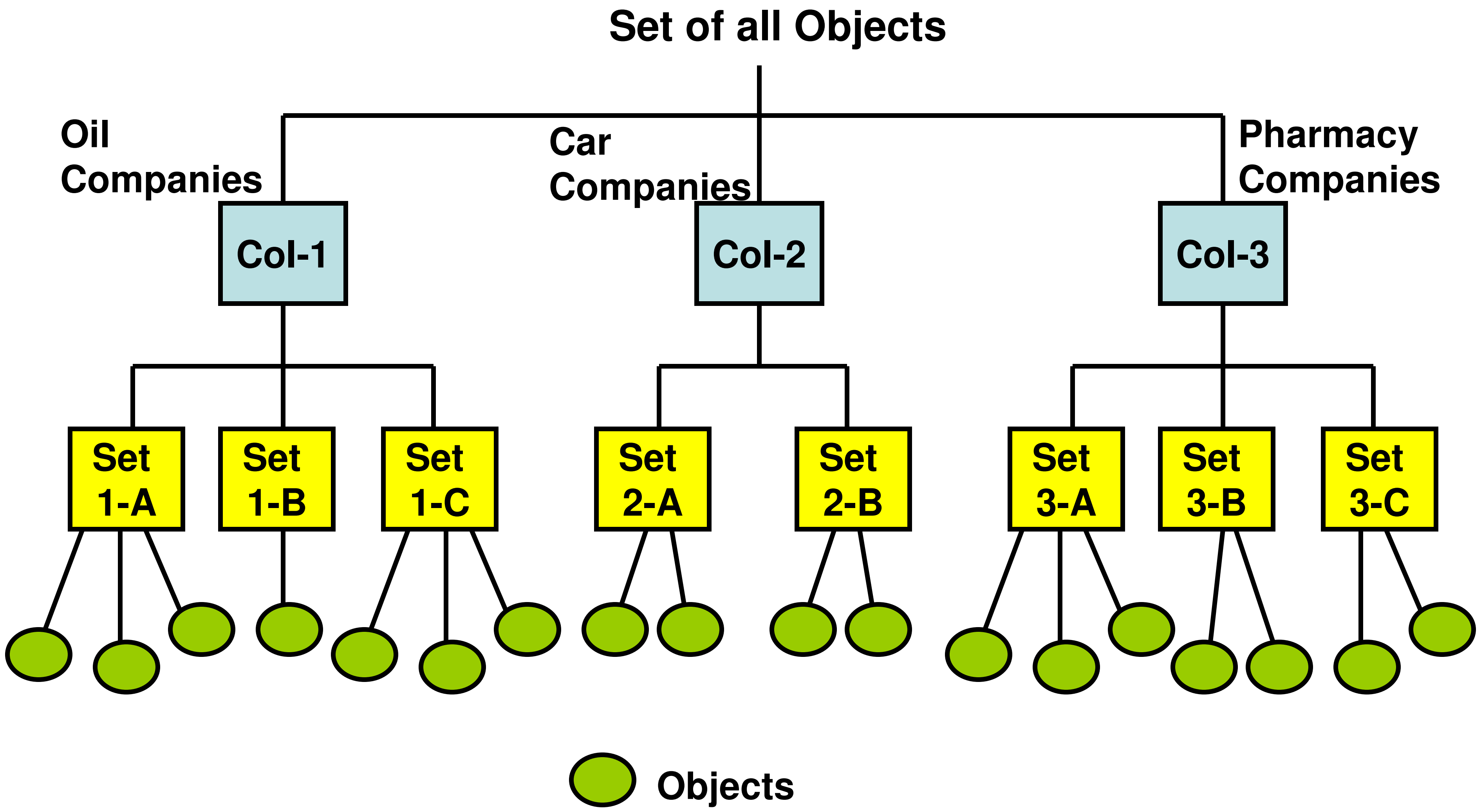
- It dynamically establishes the access rights of a user based on what the user has already accessed during the current login session.
- The Chinese Wall model groups objects into different conflict classes.
- Initially, a user is allowed to access any object belonging to any conflict class. But, once a user access an object, a wall engulfs around the user and the user cannot access any other object that is in a class conflicting with the current list of classes whose objects the user has already accessed.
- Analogy to Chinese Wall: A user standing on the top of the Great Wall of China, can jump on either sides of the wall. But, once the user jumps off to one side, it is not possible for the user to jump to the other side.

Chinese Wall Policy

- Subjects: Active entities accessing protected objects
- Objects: Data organized according to 3 levels:
  - Information, Data Set and Conflict-of-Interest (CoI) classes
- Access Rules:
  - Read rule and Write rule
- Read Rule: A subject S can read an object O if:
  - O is in the same Data Set as an object already accessed by S (OR)
  - O belongs to a Conflict-of-Interest (CoI) class from which S has not yet accessed any information
- Write Rule: A subject S can write an object O if:
  - S can read O according to the Read Rule

    (AND)
  - No object has been read by S which is in a different data set to the one on which write is performed.

Example Data Classification to Illustrate the Chinese Wall Model



Example for the Chinese Wall Model – Read Rule

- Assume a Subject S has so far accessed the following objects in the data sets of the different classes of Conflict of Interest:
  - An object in the Data Set 2-B
  - An object in the Data Set 3-A
- Question: Can the subject S read an object from the Data Sets in the following order:
  - An object in the Data Set 1-C
  - An object in the Data Set 2-A
  - An object in the Data Set 3-A
  - An object in the Data Set 1-A
  - An object in the Data Set 3-B
- Answer:
  - An object in the Data Set 1-C <u>would be Read</u> as the subject has not yet accessed any Data Set in the CoI-1.
  - An object in the Data Set 2-A <u>could not be Read</u> as the subject has already read (accessed) an object belonging to Data Set 2-B, which is in the same Conflict-of-Interest class of Data Set 2-A.

Example for the Chinese Wall Model – Read Rule (continued…)

- Answer:
  - An object in the Data Set 3-A would be Read as the subject has already accessed an object from the same Data Set.
  - An object in the Data Set 1-A could not be Read as the subject has already read (accessed) an object belonging to Data Set 1-C, which is in the same Conflict-of-Interest class of Data Set 1-A.
  - An object in the Data Set 3-B could not be Read as the subject has already read (accessed) an object belonging to Data Set 3-A, which is in the same Conflict-of-Interest class of Data Set 3-B.

- Question:
  - Can the Subject write an object from Data Set 1-A to Data Set 2-B?
  - Can the Subject write an object from Data Set 2-A to Data Set 3-C?

Chinese Wall Model – Why the Read Rule cannot be followed for Write?

- Assume in the previous example, Subject John had access to the Data Sets 1-A and 2-A and Subject Jane had access to the Data Sets 1-B and 2-A.
- Assume John wants to write an object read from Data Set 1-A to Data Set 2-A
- If the Read Rule is merely followed to decide on the write operation, then, John would be able to read the object from 1-A and write to 2-A. Jane would then be able to read the object from Data Set 2-A and write it to Data Set 1-B, which is in the same Conflict-of-Interest class as of Data Set 1-A.
- IMPORTANT: The second condition of the Write Rule: "No object has been read by S which is in a different data set to the one on which write is performed," constraints a subject from not being able to write any object if the subject has read objects from more than one data set. Hence, in order to be able to perform a write operation, the subject must read and write objects only from the same data set – typically, the native data set to which the subject could be provided access immediately after login.

Refinement to the Write Rule of Chinese Wall Model - Sanitization

- For flexibility, the write model was refined as follows: A subject could write an object to a target Data Set if the object could be read from the source Data Set according to the Read Rule and the object has been “sanitized”.
- What is Sanitization?: Sanitizing an object refers to disguising information in the object about the Data Set to which it belongs to. The information that is often hidden includes the unique identification information for the Data Set so that proprietary information about the Data Set cannot be obtained through backward inference.

- In summary, to decide on the write access for a subject on a sanitized object, the Read Rule for the subject on the object has to be merely followed. The sanitized object can freely flow across the DBMS as long as the subject can read the object to perform the write operation.
- On the other hand, unsanitized information is contained only within the native data set of the subject.

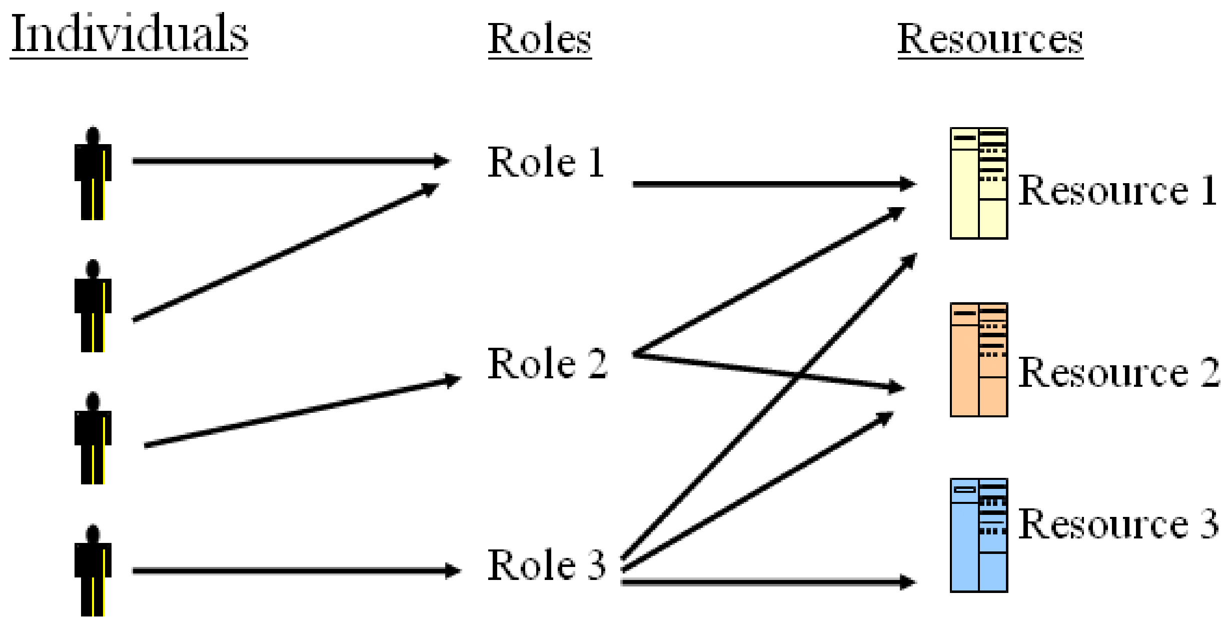
Example for the Chinese Wall Model – Refined Write Rule

- Assume a Subject S has so far accessed the following objects in the data sets of the different classes of Conflict of Interest:
  - An object in the Data Set 2-B
  - An object in the Data Set 3-A
- Question: Can the subject S read a sanitized object from and to the Data Sets in the following order:
  - An object from Data Set 1-A to Data Set 2-B?
  - An object from Data Set 2-A to Data Set 3-C?
- Answer:
  - An object in the originating source Data Set 1-A could be Read as the subject has not yet accessed any Data Set in the CoI-1. The object could be written to Data set 2-B to which the subject has "Read" access.
  - An object in the Data Set 2-A could not be Read as the subject has already read (accessed) an object belonging to Data Set 2-B, which is in the same Conflict-of-Interest class of Data Set 2-A.

Rule of Thumb: Check whether the subject has "Read" access or "could read" to both the Source Originating Data Set and the Target Destination Data Set.

Role-based Access Control (RBAC)

- A user has access to an object based on his/ her assigned role in the system. Roles are defined based on job functions.
- The motivation for the RBAC model arises from the fact that (i) as the number of subjects and/or objects increases in a DBMS, it would lead to scalability problems with the traditional DAC and MAC models for access control and (ii) the access permissions for a subject really depends on their job/function they perform on the objects and not really on the specific subject, as an individual, per se.

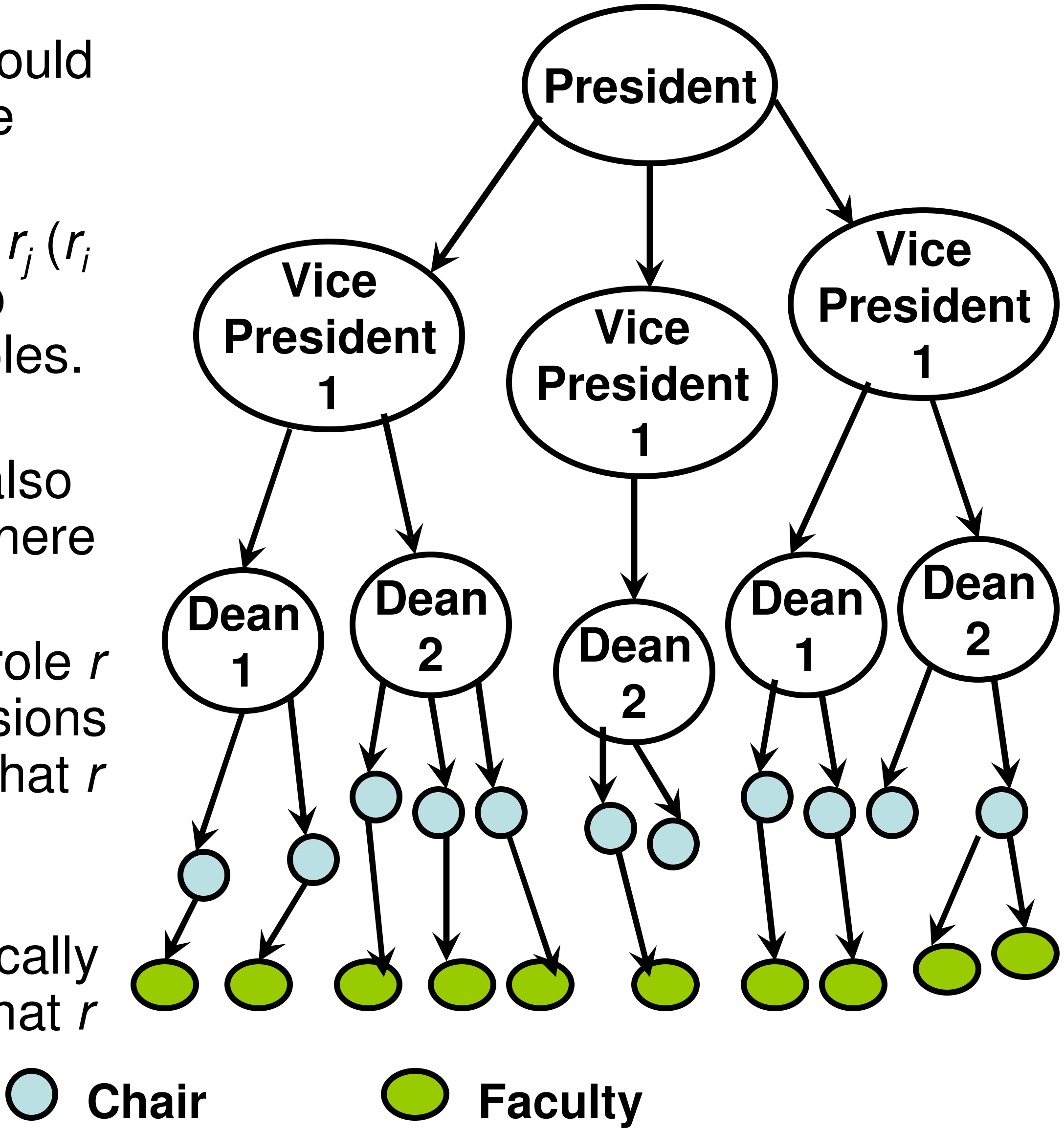


RBAC Model Sub Categories

- RBAC Session: It is a particular instance of a connection of a user to the system and defines the subset of activated roles. When a user logs in the system, he/ she establishes a session, and during this session, can request to activate a subset of the roles he/she is authorized to play.
- Core RBAC (Flat RBAC): Mainly used if a user is assigned only one role per login session.
- Hierarchical RBAC: If the user can take on multiple roles during a login session, the number of times the access permissions have to be checked can be minimized if the user is authenticated for the role that is higher up in the hierarchy among the roles the user intends to take on during the session.
- Constrained RBAC: The number of roles and/ or the combination of roles that a user can simultaneously take during a login session could be restricted according to a static or dynamic policy of Separation of Duties (SoD).
- The Static SoD constraints can be enforced upfront (i.e., before login) based on the users-role mapping and the maximum number of roles per user per session..
- The Dynamic SoD constraints can be enforced by keeping track of the user access to roles within a session. In addition to limiting the number of roles per session, constraints could be even subjective (for e.g., a user cannot be assigned role r2 if he/she already has a role r1 in the same session).

Hierarchical RBAC

- The hierarchy in a RBAC could be represented either in the form of a tree or a lattice.
- We say a role r_i dominates r_j ($r_i \geq r_j$), if r_i appears higher up than r_j in the hierarchy of roles.
- User Inheritance: All users authorized for a role *r* are also authorized for any role *r'* where $r \geq r'$.
- Permission Inheritance: A role *r* is authorized for all permissions for which any role *r'*, such that $r \geq r'$, is authorized.
- Activation Inheritance: Activating a role *r* automatically activates all roles *r'*, such that $r \geq r'$.



RBAC Separation of Duties

- The idea behind enforcing the principle of Separation of Duties (SoD) is that it prevents users from exceeding a reasonable level of authority from their roles/positions.
- With Separation of Duties, we ensure that any failure of the DBMS can be caused only as a result of collusion among the individuals.

- <u>Definitions for Separation of Duties</u>
- ANSI: “Dividing responsibility for sensitive information so that no individual acting alone can compromise the security of the data processing system.”
- The U. S. Office of Management and Budget’s Circular A-123: “Key duties and responsibilities in authorizing, processing, recording, and reviewing official agency transactions should be separated among individuals.”

Mandatory Access Control (MAC)

- Used in multi-level security systems.
- Access permissions are decided by the operating system and not by the subject (i.e., owner of an object).
- Each subject as well as object is identified with a security label.
- The most widely known system of classification of information is that implemented by the U.S. government: top secret (extensively grave damage to the national security if publicly available), secret (grave damage) and confidential (damage), restricted (undesirable effects) and unclassified (if it does not fit in the above 4 categories).

Security Models

- The security model plays a crucial role in implementing the security policies framed by an organization.
- <u>Confidentiality Model – Bell-LaPadula Model</u>
- Confidentiality – Data should not be disclosed to unauthorized individuals.
- A confidentiality model cannot however guarantee that unauthorized individuals can modify or delete data (this is a functionality of an integrity model, and not a confidentiality model).
- The Bell-LaPadula model is made up of two rules:
  - No read up rule
  - No write down rule
- The “no-read-up” rule states that no subject (such as a user or a program) can read information from an object (such as a file) with a security classification higher than that possessed by the subject itself.
- The “no-write-down” rule states that a subject can write to an object only if the subject’s security classification is lower than or equal to the object’s security classification.

Bell-LaPadula Confidentiality Model

- By itself, the “no-read-up” rule make sense because a subject can read only objects that are of the same security classification or below. Allowing a subject to read objects that are of relatively higher level of security classification would lead to unauthorized disclosure.
- However, it may be confusing for someone to look at the “no-read-up” and “no-write-down” models together.
- According to the “no-write-down” model, a subject can only write/modify objects that are of equal or higher level security classification; however, such higher-level objects that can be modified or written by a subject cannot be read.
- The “no-write-down” model does not allow write access on objects with a relatively lower level of security classification and this could be attributed to avoid either accidental or deliberate security disclosures.
- For example, if it were possible for a user with a “Top Secret” clearance to either deliberately or accidentally write Top Secret information and place it in a file marked Secret, a user with only a Secret security clearance could then access this file and view the Top Secrete information. Thus, data would have been disclosed to an individual not authorized to view it. This is what the system should protect against and is the reason for the “no-write-down” rule.

Biba Integrity Model

- In the Biba model integrity levels are used instead of security classifications.
- A principle of integrity levels is that data with a higher integrity level is believed to be more accurate or reliable than data with a lower integrity level.
- Integrity levels indicate the level of “trust” that can be placed on information at different levels.
- The Biba model for integrity is made up of the following three rules:
  - No read down rule; No write up rule; No execute up rule
- The “no-write-up” rule is more straightforward as it prevents subjects from writing to objects of a higher integrity level, as expected maintain integrity of the object.
- The “no-read-down” rule is meant to avoid a situation that when a subject reads an object of lower integrity level and creates a new object based on the original read, the newer object will have an integrity level same as that of the subject that created the object. This would lead to a situation where there are two objects with the same data – but they are of different integrity levels. To avoid such situation, the “no-read-down” rule is enforced.
- The “no-execute-up” rule states that a subject can execute a program only if the program’s integrity level is equal to or less than the integrity level of the subject. This is to ensure that neither the program nor the subject can access an object that has a relatively higher priority level (than either of them).

Multi-Level Security (MLS) Database

- With the use of Multi-Level Security (MLS), a DBMS can allow subjects with different security clearances to simultaneously access objects with different security levels.
- The Security clearances and security levels typically considered are: Top Secret (TS), Secret (S), Confidential (C) and Unclassified (U).
- MLS allows subjects with higher security clearance to easily allow access objects with equal or lower security level.
- The ordering among these clearances and security levels is as follows:
  - TS > S > C > U
- <u>The MLS approach is a classical example for the Mandatory Access Control (MAC) model</u> as the security clearance and security level for the subjects and objects are uniformly adopted enterprise-wide.
- Typically, each field (element object) in the table is assigned a security level and the security level for a tuple is the highest of the security levels of its constituent element objects.

Example for MLS Database

- Consider the following Storage database wherein, for simplicity, each tuple is assigned a security level.

Room No.	Compartment	Description	Security Level
450	X	Textbooks	Unclassified
450	Y	Computers	Unclassified
451	X	Bank Records	Confidential
451	Y	Furniture	Unclassified
452	X	Food	Unclassified
452	Y	Research Equipment	Unclassified

- Each room contains two compartments (X and Y). The combination of the room number and the compartment forms the Primary Key.
- There has to be “two” views of the above table. The DBA and other managerial-level users have to be able to see all the six tuples of the above table; whereas, a regular user should not be able to see the third tuple that has a Confidential information (Bank Records!!)

Example for MLS Database

- If a regular user with security clearance ‘Unclassified’ runs the following SQL Query
  - Select * from Storage
- The results should be:

Room No.	Compartment	Description	Security Level
450	X	Textbooks	Unclassified
450	Y	Computers	Unclassified
451	Y	Furniture	Unclassified
452	X	Food	Unclassified
452	Y	Research Equipment	Unclassified

- However, the above display of records leads the Unclassified user to believe that there is nothing in the Compartment X of Room no. 451.
- If the user executes an Insert SQL query like “INSERT INTO Storage VALUES (‘451’, ‘X’, ‘Fruits’, ‘Unclassified’); then the DBMS would return an Access denied feedback.

Example for MLS Database

- The ‘Access Denied’ feedback could mean to the Unclassified user that there is already a tuple with the primary key matching with what he is trying to insert and that the security level of the tuple is greater than his security clearance. This is referred to as an “Indirect Channel.”
- If the employee comes to know about the presence of something that is more than “Unclassified” in Compartment X of Room no. 451, then he/she could potentially do something harmful to the entity being stored over there and cause damage to the organization.
- Polyinstantiation
- To avoid such a dangerous inference, the organization could allow the employee to insert a tuple at the Unclassified level in the database. However, this would lead to a situation wherein there are two tuples with the same value for the primary key: Room no. 451 and Compartment X.
- The presence of two or more rows (tuples) with the same value(s) for the primary key, obviously with different security levels for each tuple is called “Polyinstantiation.”

Example for MLS Database Polyinstantiation

Room No.	Compartment	Description	Security Level
450	X	Textbooks	Unclassified
450	Y	Computers	Unclassified
451	X	Bank Records	Confidential
451	X	Fruits	Unclassified
451	Y	Furniture	Unclassified
452	X	Food	Unclassified
452	Y	Research Equipment	Unclassified

Polyinstantiated Rows

- When user with security classification "Confidential" or above executes the SQL query "Select * from Storage," all the above 7 tuples would be returned. The user has to however discard the polyinstantiated tuples with security level below to that of the user.

Example for MLS Database Polyinstantiation

- When user with security classification “Unclassified” executes the SQL query “Select * from Storage,” the tuples that are of security level “Unclassified” would only be returned. The user is made to believe that there are Fruits in Compartment X of Room No. 451. This is better than displaying “NULL” for the sensitive attribute and raising suspicion for the Unclassified that something sensitive is over there.
- For the organization, this will appear like a “**<u>Cover Story</u>**” that it spreads among users who are not authorized to know the actual information pertaining to that tuple.

Room No.	Compartment	Description	Security Level
450	X	Textbooks	Unclassified
450	Y	Computers	Unclassified
451	X	Fruits	Unclassified
451	Y	Furniture	Unclassified
452	X	Food	Unclassified
452	Y	Research Equipment	Unclassified

Visible and Invisible Polyinstantiation

- Visible Polyinstantiation occurs when a user with a higher security clearance attempts to insert a tuple into a database that already has a tuple, with the same primary key, at a lower security level.

- Invisible Polyinstantiation occurs when a user with a lower security clearance attempts to insert a tuple into a database that already has a tuple, with the same primary key, at a higher security level.
- The Polyinstantiation that we created in the previous slides (Storage database Example) is an example for Invisible Polyinstantiation.
- <u>Consequence of Polyinstantiation:</u>
- An after-effect of allowing polyinstantiation is that there will be an explosion of tuples in a database. If the number of security levels is 4, for every tuple, that could be entered with the highest security level in the database table, there would be at most three “Cover Story” tuples that could be recorded.
- Polyinstantiation is inevitable in Multi-level world!!