**Chapter No 4: Access Control**

In **Discretionary Access Control (DAC)**, the owner of a resource (like a file or folder) has the authority to decide who can access it and what permissions others have.

**Example:**

Imagine you're working on your computer and you have a **document** called project.docx. Since you created the document, you are the **owner**. As the owner, you have control over who else can access or edit this file.

1. **You (the owner)** create the file project.docx.
2. You decide that your friend **Alice** can **read** the file, but you don't want her to **edit** it.
3. You also give your colleague **Bob** permission to **both read and edit** the file.
4. Alice, because she only has read access, can view the file but cannot make changes.
5. Bob, with both read and write access, can view the file and also edit its contents.

In this case, **you** (the file owner) have the **discretion** to grant or revoke access to others, making this **Discretionary Access Control**.

**Key Points of DAC:**

* The **owner** controls access to the resource.
* Permissions (like read, write, or execute) are granted by the owner to other users.
* It's flexible because the owner can change the permissions at any time.

DAC is commonly seen in operating systems like Windows and Linux, where file owners can decide who gets access to their files and what actions they can perform.

**Role-Based Access Control (RBAC) - Simple Example**

In **Role-Based Access Control (RBAC)**, access to resources is granted based on the **role** a user has within an organization. Instead of assigning permissions directly to each user, permissions are assigned to **roles**, and users are then assigned to those roles. This makes managing access easier, especially in large organizations.

**Example:**

Imagine you work at a **hospital**, where employees have different roles. Each role comes with specific permissions to access patient records or other hospital resources.

**Roles and Permissions:**

1. **Doctor**: Can view and edit patient medical records.
2. **Nurse**: Can view patient records but cannot edit them.
3. **Receptionist**: Can only view patient appointment schedules, but cannot access or edit medical records.

**Example Scenario:**

* **Dr. Smith** is a doctor, so he is assigned the "Doctor" role. With this role, he can both **view** and **edit** patient medical records.
* **Nurse Jane** is assigned the "Nurse" role. She can **view** patient medical records but cannot make any changes.
* **John** works as a receptionist and is assigned the "Receptionist" role. He can only **view** the appointment schedules and does not have access to medical records at all.

**Key Points of RBAC:**

* Users are assigned **roles**, not individual permissions.
* Each **role** has a predefined set of permissions.
* If a person’s job changes (e.g., from Nurse to Doctor), they simply switch to a new **role** with the corresponding permissions.

RBAC simplifies managing access, especially in environments with many users. For example, instead of manually updating permissions for every doctor, you just assign the "Doctor" role to all doctors, and they automatically get the right permissions.

**Attribute-Based Access Control (ABAC) - Simple Example**

In **Attribute-Based Access Control (ABAC)**, access to resources is granted based on attributes (characteristics) of the user, the resource, or the environment. These attributes can be anything, like a user's role, department, time of day, location, or the classification of the resource. The access rules are defined using these attributes.

**Example:**

Imagine a **university system** where different users need access to a **student database**. Access is controlled based on **attributes** of the users, like their **job title**, **department**, or even **time of access**.

**Attributes in Use:**

1. **User attributes**: Job title (Professor, Student, Admin), Department (Science, Arts).
2. **Resource attributes**: Resource type (Exam Grades, Lecture Notes, Student Info).
3. **Environmental attributes**: Time of day, Location (on-campus, off-campus).

**Example Scenario:**

* **Professor Alice**: She has the attribute "Job title: Professor" and "Department: Science." Alice can access **exam grades** and **lecture notes** for the Science department, but only during working hours (8 AM - 5 PM).
* **Student Bob**: He has the attribute "Job title: Student" and "Department: Arts." Bob can only view **lecture notes** for the Arts department, and cannot access exam grades. Bob can access these resources from both on-campus and off-campus.
* **Admin Clara**: Clara, with the attribute "Job title: Admin," has broader access to **student info** and **exam grades** but is restricted by the rule that says she can only access the database from within the university’s office network (on-campus).

**How ABAC Works:**

* **Professor Alice** tries to access the student database at 9 AM from her office. Her attributes (Professor, Science) and the resource’s attributes (Exam Grades, Lecture Notes) are checked. Since she meets the conditions, she is granted access to Science exam grades and notes.
* **Student Bob** tries to view exam grades. However, his attributes (Student, Arts) do not match the condition that allows access to exam grades (only Professors can access them), so his access is denied.
* **Admin Clara** tries to access student info from home (off-campus), but because her location attribute is off-campus, her access is denied.

**Key Points of ABAC:**

* **Access decisions** are based on multiple attributes (user, resource, environment).
* **Flexible** and can accommodate complex access rules.
* Provides **fine-grained control** because access can be tailored using many different conditions.

ABAC is more dynamic than RBAC because it doesn’t just depend on a user's role, but on various conditions and attributes, allowing for more complex and specific access controls.

**Mandatory Access Control (MAC) - Simple Example**

In **Mandatory Access Control (MAC)**, access to resources is controlled by a central authority and **cannot be altered** by individual users. In this model, both users and resources (files, documents, etc.) are assigned security classifications, and access is granted based on these classifications. Users cannot change the access rules themselves.

**Example:**

Imagine you're working in a **government agency** that deals with sensitive information. The agency has a strict classification system for both documents and employees based on security levels.

**Security Levels:**

1. **Top Secret**: Highest classification.
2. **Secret**: Mid-level classification.
3. **Confidential**: Lower-level classification.

**Example Scenario:**

* **John** is a government employee with a **Secret** clearance.
* **Mary** is another employee with a **Top Secret** clearance.
* There are three types of documents:
  + **Document A** is classified as **Top Secret**.
  + **Document B** is classified as **Secret**.
  + **Document C** is classified as **Confidential**.

**How MAC Works:**

* **John** (Secret clearance) can access **Document B** (Secret) and **Document C** (Confidential), but he **cannot** access **Document A** (Top Secret) because his clearance is not high enough.
* **Mary** (Top Secret clearance) can access **all three documents** (Top Secret, Secret, and Confidential) because her clearance level is the highest.
* **No one** can change these access levels. For example, John cannot decide to share Document B with someone else. Access is strictly controlled by the central authority and based on the clearance levels.

**Key Points of MAC:**

* Access is based on **predefined security classifications**.
* **Users cannot modify** the access rules.
* Access is enforced by a **central authority**, often used in environments that require strict security (e.g., military, government).

MAC is used in highly sensitive environments because it provides strict and rigid control over who can access what, minimizing the risk of unauthorized access.