D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Subject Name: Information System

Unit No:02 Unit Name: Access Control Models

Faculty: Mrs. Bhavana Alte
Mr. Prathmesh Gunjgur

Unit No: 2 **Unit Name: Access Control Models**

Role-Based and Task-Based Access Control

Role-Based Access Control (RBAC)

- **Role-Based Access Control (RBAC)** is a security model where access to resources is based on the **roles** that users hold in an organization.
- In RBAC, access rights (permissions) are **assigned to roles**, and users are **assigned to roles** based on their job responsibilities.
- Each role comes with a set of **permissions**, which define what the users in that role can **access and perform**.

Key Concepts:

•**Roles**: These represent job functions within the organization. Examples include **Admin**, **Manager**, **Employee**, or **Guest**.

•**Permissions**: These define what actions users can perform on resources. For example, **read**, **write**, **delete**, or **update**.

•**Users**: These are the individuals or entities that are assigned to roles, such as employees, customers, or clients.

•**Resources**: These are the objects users want to access (e.g., databases, files, applications, etc.).

How Does RBAC Work?

•A user is **assigned a role** based on their job function.

•Each role has **specific permissions** associated with it. Permissions define the level of access the role has on certain resources.

•A user inherits the permissions of the **role(s)** they are assigned to.

•The system **enforces the role-based rules**, granting access to resources based on the user's role.

RBAC Example:

- Let's say we have a company with three roles: **Admin**, **Manager**, and **Employee**.
- **Admin Role**:
  - Permissions: **Create**, **Read**, **Update**, **Delete** files, **Manage Users**.
- **Manager Role**:
  - Permissions: **Read**, **Update** files, **Approve Reports**.
- **Employee Role**:
  - Permissions: **Read** files only.
- **Now, let's assign users to roles:**
- **Alice** is an **Admin**. She has full control over all resources.
- **Bob** is a **Manager**. He can read and update files, but cannot delete them or manage users.
- **Charlie** is an **Employee**. He can only read files but cannot modify them.

Advantages of RBAC:

•**Simple Management**:

•Roles are predefined, so it is easier to assign and revoke access.

•Instead of managing individual permissions for each user, roles can be created, and users can be grouped under them.

•**Scalability**:

•As organizations grow, RBAC helps to efficiently manage permissions for large numbers of users. You just have to manage the roles rather than permissions for each individual user.

•**Security**:

•By defining roles with specific permissions, organizations can follow the **Principle of Least Privilege**, ensuring that users only have the necessary permissions for their job.

Task-Based Access Control (TBAC)

- **Task-Based Access Control (TBAC)** is a security model where access to resources is determined by the **tasks** a user needs to perform, rather than their role.
- In TBAC, the system assigns permissions based on specific tasks or activities that a user must complete, and the **access rights are linked directly to the task** at hand.

Key Concepts:

•**Tasks**: A task represents a specific job or function a user is assigned to complete. Tasks are time or event-based actions, such as generating a report, reviewing a document, or approving a purchase request.

•**Permissions**: Permissions define what actions a user can perform on resources to complete the assigned task.

•**Users**: Users are assigned to perform specific tasks and are granted the permissions needed to complete those tasks.

How Does TBAC Work?

•Users are assigned tasks, and based on these tasks, they are granted permissions to perform actions on certain resources.

•The system grants **task-specific permissions** that can be activated based on the task at hand. For example, a user may be granted access to modify a document only when assigned the task of **reviewing** it.

TBAC Example:

- Imagine a **document review system** in a **company** where users are assigned tasks related to reviewing and editing documents.
- **Task 1**: **Review Report**:
  - Permissions: **Read**, **Edit** report documents.
- **Task 2**: **Approve Report**:
  - Permissions: **Approve** the report but **cannot edit**.
- **Task 3**: **Generate Report**:
  - Permissions: **Create**, **Save**, and **Share** the report.

- **Now, let's assign users to specific tasks:**
- **Alice** is assigned to **Task 1** (Review Report). She has permissions to **read and edit** the report.
- **Bob** is assigned to **Task 2** (Approve Report). He can **approve** the report but cannot modify its content.
- **Charlie** is assigned to **Task 3** (Generate Report). He has permissions to **create and save** reports but cannot approve or edit them.

Advantages of TBAC:

- **Dynamic Access Control**:
  - TBAC allows more flexibility because permissions can be granted based on the specific task a user is performing, rather than their role in the organization.
- **Context-Specific Access**:
  - It ensures that users only have access to the resources they need to perform a specific task, which reduces the chance of unauthorized access or privilege escalation.
- **Fine-Grained Control**:
  - TBAC offers more granular control over **what** users can do. For example, a user might be granted access to only **edit reports** when assigned the **task** of reviewing them but not when they're doing other tasks.

Real-World Example of TBAC:

- Let's consider a **Project Management System** in a company:
- **Task 1: Assigning Tasks**:
  - Permissions: **Assign tasks** to employees.
  - A **Manager** is responsible for assigning tasks, and they are granted the **Assign Task** permission only when they're actively managing tasks.
- **Task 2: Reviewing Task Progress**:
  - Permissions: **View task progress**, **Update task status**.
  - A **Team Lead** is granted the permission to review the progress of tasks but cannot assign them.
- **Task 3: Approving Completed Tasks**:
  - Permissions: **Approve** or **reject** tasks that are completed.
  - **Managers** or **Project Managers** can approve the completion of tasks.

Unified Access Control Models:

- **Unified Access Control Models** combine elements of different access control models to provide a **comprehensive and flexible approach** to access control.
- In the real world, different organizations may have varied needs, and no single access control model (such as **Role-Based Access Control (RBAC)** or **Task-Based Access Control (TBAC)**) may meet all their security requirements.
- **Unified models** aim to blend the strengths of multiple models, addressing the **varied security needs** of an organization while **simplifying management** and ensuring **consistent enforcement** of policies across different levels of access.

Key Concepts of Unified Models

- A **Unified Access Control Model** typically:
- **Combines** multiple models, such as **RBAC**, **ABAC (Attribute-Based Access Control)**, and **MAC (Mandatory Access Control)**, to cover a wide range of use cases.
- **Supports complex and flexible policies**, combining the static nature of **roles** with the **dynamic nature of attributes** (like time, location, and environment).
- **Centralizes the management** of access policies while still considering the various permissions required at different organizational levels.

Types of Unified Access Control Models

- Role-Based and Attribute-Based (RBAC + ABAC)
- Role-Based and Task-Based (RBAC + TBAC)
- Discretionary and Mandatory Access Control (DAC + MAC)
- Combining RBAC, ABAC, and MAC

1. Role-Based and Attribute-Based Access Control (RBAC + ABAC)

- **ABAC** (Attribute-Based Access Control) assigns permissions based on **attributes** (e.g., department, time of day, location) of the **user**, **resource**, and **environment**. For example, a user may be granted access to a resource if their role is "Manager" and the current time is within working hours.

- In a **Unified RBAC + ABAC Model**, access control is determined based on both:
- The **user's role** (e.g., Admin, Employee).
- **Attributes** such as time of day, location, or other environmental factors.

- **How it Works:**
- **Role**: A user is assigned a role (Admin, Manager, Employee).
- **Attributes**: Additional attributes (such as time or location) determine the **context** of access.

Example:

- Consider a company where:
- **Admin** has the role to **create**, **read**, **update**, and **delete** files.
- **Manager** has the role to **read** and **update** files but cannot delete them.
- **Employee** has the role to **read** files but cannot modify them.

- In this **RBAC + ABAC** model, an additional **attribute** might be added:
- **Time of Day Attribute**: Access to certain resources (e.g., sensitive financial reports) may only be allowed during working hours (9 AM to 5 PM).
- **Location Attribute**: Employees accessing resources from **outside the company network** may be restricted to read-only access, while those on the company network may have full access.

Benefits of this Unified Model:

•**Flexibility**: Combines **role-based security** with the flexibility of **contextual attributes**, such as time, location, or even device type.

•**Granularity**: Provides fine-grained access control based on dynamic factors, making it suitable for modern, dynamic environments.

2. Role-Based and Task-Based Access Control (RBAC + TBAC)

- In a **Unified RBAC + TBAC Model**, access control is granted based on both:
- The **role** of the user (e.g., Admin, Manager).
- The **specific task** they are performing (e.g., reviewing a report, approving a request).

- **How it Works:**
- **Role**: A user is assigned a role (Admin, Employee, etc.), which determines the **general permissions** they have.
- **Task**: For each specific task the user performs (e.g., reviewing a report, approving a request), they are granted additional permissions.

Example:

- Consider a **Document Management System** where employees have different roles and need to perform specific tasks:
- **Admin Role**:
  - Permissions: Full access to **all tasks** (create, read, update, delete documents).
- **Manager Role**:
  - Permissions: Can only **review** and **approve** reports.
- **Employee Role**:
  - Permissions: Can **create** and **read** reports but cannot approve them.
- In this **RBAC + TBAC** model, access permissions depend on:
- **Role**: The **Admin** has full control over the document system, while the **Employee** has limited access.
- **Task**: The **Manager** can review and approve reports, but cannot create new ones.

Benefits of this Unified Model:

- **Fine-Grained Control**: Combines the **role-based structure** with **task-specific access**, providing more flexible and precise control over what users can do.
- **Task-Oriented Security**: Access is determined not just by the role but by **what the user is trying to do**, adding an extra layer of security.

3. Discretionary and Mandatory Access Control (DAC + MAC)

- **Example:**
- **DAC**: A file owner grants access to specific users (e.g., “Employee A” can access file X).
- **ABAC**: The access might depend on the **time of day.** For example, "Employee A" might only be able to access file X **during office hours** but not outside of those hours.

Scenario 1: Healthcare System (RBAC + ABAC)

- In a **healthcare system**, a unified model combining **RBAC** and **ABAC** might be used:
- **Roles**: Doctors, Nurses, Administrative Staff.
- **Attributes**: Patient’s condition (critical, non-critical), Time of Day (emergency hours, regular hours), Department (Cardiology, Neurology).

- **Example Access Decision**:
- A **Doctor** in the **Neurology Department** can access **patient records** based on their role, but access might be restricted during **non-emergency hours** unless the patient’s condition is **critical**.

Scenario 2: Government Agency (RBAC + MAC)

- A **government agency** might implement a unified model using **RBAC** and **MAC**:
- **Roles**: Admin, Analyst, Employee.
- **Security Levels**: Top Secret, Secret, Confidential, Public.

- **Example Access Decision**:
- An **Admin** role might have the clearance to modify resources, but their access to **Top Secret** data is restricted by **MAC policies** unless they have the appropriate **security clearance**.

Scenario 3: Financial System (DAC + ABAC)

- A **financial system** might combine **DAC** and **ABAC** for access control:
- **Resource Owners**: Finance department members (who own financial reports).
- **Attributes**: User department, location (office or remote), time of day.
- **Example Access Decision**:
- **DAC**: A finance department employee might grant access to a financial report to a colleague.
- **ABAC**: However, access might be restricted based on attributes like time of day (e.g., reports can only be accessed during office hours), location (e.g., employees working remotely may have restricted access), and departmental roles (e.g., only those in the finance department).

Access Control Algebra

- **Access Control Algebra** is a formal approach to modeling and managing access control policies. It involves using **logical operations** and **set theory** to combine and define access control rules, policies, and permissions.
- The goal is to provide a **structured** and **mathematical** way to combine different access control decisions to ensure **secure access** to resources.
- Access control algebra allows for the **combination of multiple policies** in a logical and consistent way to define how permissions are granted or denied.

Key Concepts in Access Control Algebra:

•**Permissions**: These are the actions users can perform on resources (e.g., **read**, **write**, **execute**).

•**Subjects**: The entities trying to access the resources (e.g., **users**, **processes**, **devices**).

•**Objects**: The resources to be protected (e.g., **files**, **databases**, **documents**).

•**Access Control Policies**: Rules that define what permissions are granted to subjects for accessing objects (e.g., **who can access what resources and how**).

•**Logical Operations**: Set-based operations such as **union**, **intersection**, and **complement** are used to combine access control policies.

Basic Operations in Access Control Algebra:

- Access Control Algebra uses logical set operations to manipulate access policies.
- The key operations include:
- **Union (OR):** Combining two policies to grant access if either of the policies allow it. This operation means that a subject can access a resource if at least one policy allows the access.
- **Symbol:** $A \cup B$
- **A∪BMeaning**: Either policy A or policy B (or both) allow the access.
- Example: A user can read a file if they are either a Manager or have read permission granted by the resource owner.

Basic Operations in Access Control Algebra….:

- **Intersection (AND)**: Combining two policies to grant access only if **both policies** allow it. This operation means that a subject can access a resource only if **both policies** allow the access.
- **Symbol**: A∩B
- **Meaning**: Access is granted only if both policies A and B allow the access.
- **Example**: A user can read a document only if they are part of the **Admin role** and the document's **sensitive level** is **public**.

- ❑ **Complement (NOT)**: This operation denies access if a policy **does not allow** it.

•It represents the negation of a policy.

•**Symbol**: $\overline{A}$

•**Meaning**: Access is denied if policy $\bar{A}$ does not allow access.

•**Example**: A user cannot access a resource if they **do not** have the **Admin** role.

- ❑ **Difference (Subtract)**: This operation allows access to a resource by removing permissions granted by one policy from the permissions granted by another policy.

•**Symbol**: A−B

•**Meaning**: Access is granted based on policy A, but **denied** for permissions in policy B.

•**Example**: A user has **read** permission (policy A) but is denied the ability to **edit** the document (policy B).

Access Control Algebra with Example:

- **Scenario:**
- **User A** is a **Manager** and wants access to a **Sensitive File**.
- The **Sensitive File** has the following access policies:
  - Policy 1: Only **Managers** can access it (Role-based access).
  - Policy 2: Only users with a **Top Secret** clearance can read the file (Clearance-based access).
  - Policy 3: The file should only be accessed during **business hours** (Time-based access).

Step 1: Defining the Policies

- **Policy 1** (Role-based access):
  - If the user is a **Manager**, they are granted **read** permission.
  - Let's denote this as:
    - P1=Role(Manager)
- **Policy 2** (Clearance-based access):
  - If the user has **Top Secret** clearance, they are granted **read** permission.
  - Let's denote this as:
    - P2=Clearance(Top Secret)
- **Policy 3** (Time-based access):
  - The file can only be accessed during **business hours** (9 AM to 5 PM).
  - Let's denote this as:
    - P3=Time(Business Hours)

Step 2: Combining Policies Using Algebra

- Combine these policies using **set operations** to determine if **User A** can access the **Sensitive File**.
- **Union of Policy 1 and Policy 2** (Role OR Clearance):
  - User A is a **Manager**, so they meet Policy 1. User A also has **Top Secret** clearance, so they meet Policy 2.
  - Combined policies: $P1 \cup P2$
  - This means **User A** can access the file if they are a **Manager** or have **Top Secret clearance**.

- **Intersection with Policy 3** (Time AND):
  - The access is allowed only during **business hours**. So, the combined policy from Step 1 will be further restricted by Policy 3.
  - Final access decision: $(P1 \cup P2) \cap P3$
  - This means **User A** can access the file only if they are either a **Manager** or have **Top Secret clearance** AND it is **business hours.**

Final Access Decision

- **User A** is a **Manager** with **Top Secret clearance**, and it's **business hours**.
- According to the combined policy (P1∪P2)∩P3, User A meets the access criteria, so they are **granted access** to the **Sensitive File**.

Benefits of Access Control Algebra

•**Formal and Precise**: Access Control Algebra uses **logical operations** and **set theory**, which provides a formal and precise way to define access control policies.

•**Flexibility**: By combining policies using union, intersection, and complement, you can easily adapt access control systems to complex scenarios.

•**Scalable**: It's easy to scale up by adding more policies and combining them using algebraic operations.

Use Cases for Access Control Algebra:

•**Complex Organizational Systems**: In large organizations with multiple policies, Access Control Algebra allows for **fine-grained access control** by combining various conditions (e.g., roles, clearance levels, time restrictions).

•**Security for Sensitive Data**: For systems that handle sensitive or classified data, Access Control Algebra allows for **dynamic policies** based on different conditions.

•**Contextual Access Control**: For systems where access should depend on factors like the **time of day**, **user role**, and **resource sensitivity**, Access Control Algebra provides a flexible framework.

Temporal and Spatio-Temporal Access Control Models

- **Temporal Access Control Models** and **Spatio-Temporal Access Control Models**, which are advanced models used in **access control systems** that take into account **time** and **location** factors to make more dynamic and flexible access control decisions.
- These models can be used to enhance the security and efficiency of access control policies in various systems, including corporate environments, security systems, and other resource management systems.

Temporal Access Control Model

- A **Temporal Access Control Model** manages access rights based on **time-related factors**. In this model, access permissions are determined not only by the identity of the user or their role but also by **when** the user tries to access a resource. Temporal access control allows for access decisions based on conditions like:
- **Time of day**: Only allowing access during working hours (e.g., 9 AM to 5 PM).
- **Date-based restrictions**: Restricting access to certain days or certain time periods, such as weekends, holidays, or specific time slots.
- **Time duration**: Specifying how long a user can access a resource.

How Temporal Access Control Works:

•**Time Windows**: Permissions can be granted or denied depending on whether the access request falls within a specific time window (e.g., only accessible during office hours).

•**Time-based Policies**: Time-based access policies allow administrators to define rules such as "Only allow access to sensitive data from 8 AM to 6 PM on weekdays."

Example of Temporal Access Control:

- Imagine a **Corporate Office** that has a **confidential database** containing sensitive employee information. The company wants to ensure that only **HR personnel** can access this database during **office hours** (9 AM to 5 PM) on weekdays.
- **Policy**: Only users with **HR roles** can access the database between **9 AM and 5 PM** on weekdays.
- **Example**:
  - **Alice**, an HR manager, tries to access the database at 10 AM on a **Monday**: **Access Granted** (since it's within the allowed time window).
  - **Bob**, an HR employee, tries to access the database at **6 PM** on a **Monday**: **Access Denied** (since it's after office hours).
  - **Charlie**, a non-HR employee, tries to access the database at 10 AM on a **Monday**: **Access Denied** (since Charlie doesn't have HR clearance).

Applications of Temporal Access Control:

•**Workplace Access**: Restrict access to resources such as company databases or files only during specific working hours.

•**Secure Systems**: Allow access to security-sensitive data or systems only during authorized time periods (e.g., during a shift).

•**System Maintenance**: Limit access to a system during scheduled maintenance windows, ensuring no unauthorized access during updates.

Spatio-Temporal Access Control Model

- A **Spatio-Temporal Access Control Model** adds a **location-based** dimension to the Temporal model. It combines both **time** and **space** (location) factors to enforce access policies. This means that access to resources is not only dependent on **when** access is requested (as in Temporal models) but also on **where** the user is located at the time of the request.
- In a spatio-temporal model, access permissions are granted based on:
- **Time of access** (similar to Temporal Access Control).
- **Geographical location** (e.g., based on IP addresses, GPS coordinates, or network locations).

How Spatio-Temporal Access Control Works:

- **Time and Location Conditions**: The system evaluates both the **time** and the **location** of the user making the request. If both conditions match the allowed criteria, access is granted.
- **Geofencing**: Geofencing is a technique used in spatio-temporal models, where access to a resource is allowed only if the user is within a specific **geographical area**.

Example of Spatio-Temporal Access Control:

- Imagine a **corporate network** where **employee access** to sensitive resources is **restricted** based on both **time** and **location**.
- **Policy**: Only employees with a **Manager** role can access the **secure financial records** during **office hours** (9 AM to 5 PM) but only if they are **within the company's campus**.
- **Example**:
  - **Alice**, a **Manager**, tries to access the records at **10 AM** while on the **company campus**: **Access Granted** (both time and location match the policy).
  - **Bob**, a **Manager**, tries to access the records at **7 PM** while on the **company campus**: **Access Denied** (time is outside the allowed range).
  - **Charlie**, a **Manager**, tries to access the records at **10 AM**, but he is **outside the company campus** (e.g., in a different city): **Access Denied** (location is outside the allowed area).

Applications of Spatio-Temporal Access Control:

•**Physical Security**: In facilities where access to physical resources (e.g., rooms, labs) is based on both the **time** (e.g., during working hours) and the **location** (e.g., only within the building).

•**Mobile Device Security**: For apps or services that require users to be at a specific **geographical location** (e.g., access to a secure service only when a user is on company premises).

•**Government and Military**: Systems that require both time and location-based access to highly sensitive resources or data, like military bases or government intelligence systems.

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Thank You