**IAM/IGA Fundamentals**

What is IAM? (identity Access Management)

IAM is a framework of policies that ensures the right individuals in an organization have access to the tools they need while keeping unauthorized users out.

Key Concepts:

1. Authentication: Verifying who you are
2. Authorization: Determining what you are allowed to do
3. Users, Roles, and Groups: Represents entities and their access levels
4. Access Control: Granting or restricting access based on defined policies

Example scenario: Imagine you work at a bank. IAM ensures that

* Tellers can access account information but not server configurations.
* IT Administrators can configure servers but cannot see customer bank balances.

What is IGA? (Identity Governance and Administration)

IGA is a subset of IAM that focuses on

* + ensuring the right people have the right access
  + providing compliance with regulations
  + Managing the lifecycle of identities (e.g. when employees join, move roles, or leave)

Key functions:

1. Identity Life Cycle Management: Handles Joiners, Movers, and Leavers
2. Access Reviews/Attestation: Regular checks to confirm users have appropriate access
3. Segregation of Duties (SOD): Prevents conflicts in access (e.g. no single user should approve and pay invoice)

Example scenario: In a retail company,

* When an employee moves from Sales to HR, IGA ensures their old access is revoked (Sales) and new access is granted (HR)
* During an audit, the system generates a report showing no single employee has both "create purchase order" and "approve purchase order" permissions.

**Core IAM Concepts**

1. Authentication Mechanisms (e.g. MFA, SSO)
2. User and Role Management
3. Access Controls (e.g. RBAC, ABAC)
4. Authentication vs Authorization

Authentication: Proving an identity using

* Something you know – password
* Something you have – token
* Something you are – biometric

Example: Logging into a corporate system with username and password.

Authorization: What actions are you allowed after authentication.

Example: A manager can approve the leave request, an employee cannot.

***Activity 1: Authentication vs Authorization***

Imagine you are tasked with designing access to a corporate system for two types of users: **employees** and **contractors**. The goal is to ensure proper security while keeping the system easy to use.

**Scenario:**

1. Employees: Full-time staff who need to access email, shared drives, and the internal HR portal daily.
2. Contractors: Temporary workers hired for a short period who only need access to a project management tool and must not access sensitive employee data.

**Task:**

1. Authentication
2. **Employees**: What type of authentication method(s) would you implement for a full-time employee?
3. **Contractors**: How would you authenticate contractors differently, given their temporary nature?
4. Authorization
   1. Employees: What actions or resources should they be authorized to access after logging in?
   2. Contractors: What actions or resources should they be authorized to access, and how would you ensure that they cannot access sensitive data?

***Solution Breakdown***

1. **Authentication**

To verify the identity of employees and contractors:

* **Employees:**
* Use multi-factor authentication (MFA) (e.g., password + mobile app token)
* Justification: Employees access sensitive internal resources, so strong security is required.
* **Contractors**

Use Temporary Credentials:

* Auto-expire after the project duration.
* Password-only or password + email verification (less stringent than employees)

1. **Group Assignment**

Define user groups to simplify access management:

* Group 1: Full-time employees
* Permissions: Email, shared drives. HR portal
* Role: “Employee User”
* Group 2: Contractors
* Permissions: Only project management tools.
* Role: “Contractor User”

1. **Role-Based Access Control (RBAC)**

Assign roles to these groups:

* Employees Role:
* Actions: View/edit HR portal, access shared drives.
* Constraints: Cannot modify HR systems configurations.
* Contractors Role:
* Actions: Access project management tool only.
* Constraints: Cannot access shared drives or HR portal.

We can also apply Attribute-Based Access Control (ABAC) rules:

For example: “Deny access to HR portal if userType = Contractor.”

We can also limit the access scope by grating access only during work hours using time-based policies.

**2. User and Role Management**

**User Management**

User Management is all about creating, updating, and managing user accounts within the system.

Key concepts of User Management:

1. Attributes:

* Each user has specific attributes like name, email, department, job title, etc.

Example: name: John Doe, email: [john.doe@company.com](mailto:john.doe@company.com), department: IT

1. Identity Life Cycle:

* Joiner: When a user joins, they are created in the system.
* Mover: When a user switches roles or departments, their access is updated.
* Leaver: When a user leaves, their access is revoked.

1. Provisioning and Deprovisioning:

* Provisioning: Granting users the right access when they join or move.
* Deprovisioning: Removing access when they leave.

*Example scenario for User management:*

1. **Joiner:** John joins as a software engineer in the IT department.

* Provision him access to development tools and email.

1. **Mover:** John gets promoted to IT manager.

* Update his access to include team management tools and remove access to individual developer tools.

1. **Leaver:** John leaves the company.

* Deactivate his account and revoke all access.

**Role Management**

Roles simplify managing permissions for users by grouping access rights.

**Key concepts:**

1. **Role-Based Access Control (RBAC)**

* Assign roles based on job functions:
* Example roles:
* **HR Admin**: Access to payroll and employee records
* **IT Support**: Access to server logs and support tickets

1. **Hierarchy**

* Roles can have a hierarchy for inherited permissions.
* Example:
* **Manager Role** inherits permissions from an Employee Role and adds extra management rights.

1. **Role Engineering**

* Design roles based on the **least privilege principles** to minimize risks.

Examples scenario for Role Management:

1. A software engineer needs:

* Access to Git repositories.
* Access to testing tools.

1. A Project manager needs:

* Access to project dashboards.
* Access to team reports.
* Does not need direct access to code repositories.

***Activity 2: User and Role Management***:

1. User Lifecycle:

* John is hired as a Finance Analyst. What attributes would you assign to him and what access does he need?
* John moves to the Compliance team. What changes would you make to his access?
* John resigns. What access would you take to secure the system?

1. Role-Based Access

* Create roles for the following:
* HR Manager
* Software developer
* Customer support agent

**Solution Breakdown:**

1. **User Lifecycle Management**

* **Attributes for John when he joins:**
* Username, email, userId, job role/title, department, managerId, status
* **When John moves:**
* Update department and job role, managerId, and re-certification to ensure that John no longer has any access from his previous finance analyst role.
* **When John resigns:**
* Revoke all access and permissions.
* Add account archival: before deletion, archive the account for audit and compliance purposes.

1. **Role-Based Access Control**

* **HR Manager**
  + Selecting new employees (Access to recruitment portal)
  + Onboarding new hires (Access to employee records and payroll systems)
* **Software Developer**
* Access to code management/edit (e.g., Git, Jenkins)
* Access to testing environment.
* **Customer Support Agent**
* Access to ticketing system (e.g., Zendesk, ServiceNow)
* Access to virtual customer support (e.g., chatbot tools, live chat)

**3.Access Controls In IAM**

Access control ensures that users can only access the resources they are authorized for, based on policies and roles.

Types of Access Control

1. **Discretionary Access Control (DAC)**

* Resource owners decide who can access their resources
* Example: A file owner sharing a document with specific colleagues.

1. **Mandatory Access Control (MAC)**
   * Access is granted based on policies set by administrators.
   * Example: Classified documents are accessible only to users with “Top Secret” clearance.
2. **Role-Based Access Control (RBAC)**

* Access is assigned based on roles, simplifying management.
* Example: All members of the HR group can access payroll data.

1. **Attribute-Based Access Control (ABAC)**

* Access is granted based on user and resource attributes.
* Examples: “Allow access to the marketing portal if user.department = marketing and time < 6 pm .”

**Example Scenario:**

You work for a hospital:

* Doctors need access to patient records but not billing systems.
* Nurses can access patient records but cannot update them.
* Admin Staff can access billing but not medical records.

Using RBAC

* Create roles: “Doctor”, “Nurse”, and “Admin Staff”
* Assign permissions:
  + Doctor: Read/Write access to records
  + Nurse: Read-only access to patient records
  + Admin Staff: Access to billing systems.

Using RBAC

* Add constraints
  + Nurses can access records **only for their assigned patients.**
  + Doctors can update records **only during work hours.**

***Activity 3:***

Using the Hospital example above,

1. Define attributes needed to implement ABAC (e.g., department time)
2. Propose an RBAC structure with roles and permissions.

***Solution Breakdown:***

1. **Attributes for ABAC**
   * userRole
   * department
   * AssignedPatient—to restrict access to specific patient records (for nurses)
   * TimeOfAccess—to enforce work hour policies for doctors and nurses.
2. **RBAC Structure**

|  |  |  |
| --- | --- | --- |
| **Role** | **Permissions on Patient Records** | **Permissions on Billing** |
| **Doctor** | Read/Write | None |
| **Nurse** | Read-Only | None |
| **Admin Staff** | None | Read/Write |

**Justification:**

* Doctors handle diagnosis and updates (hence R/W on patient records)
* Nurses need to view but not modify records
* Admin staff only deal with billing and do not need access to patient records.

**Adding ABAC Rules**

Now, layer ABAC on top of RBAC for finer control:

1. **Doctors:**
   * Add timeOfAccess < 6 pm (to limit record updates during working hours)
2. **Nurses:**
   * Add: assignedPatient = True ( to ensure access is restricted to their assigned patients)
3. **Admin Staff:**
   * No additional attributes are required; access is based on their RBAC role.