System Design — Authentication vs. Authorization

Written By: *Saikat Goswami*

---

1. Introduction

In system design, **authentication** and **authorization** are two critical concepts that play a pivotal role in securing systems and controlling access to sensitive resources. Although they are closely related and often implemented together, they serve distinct purposes.

- **Authentication** answers the question, "Who are you?"
- **Authorization** answers the question, "What are you allowed to do?"

This article explores the differences, mechanisms, and best practices for implementing authentication and authorization in system design, highlighting their importance in modern distributed systems.

---

2. What is Authentication?

Definition

Authentication is the process of verifying the identity of a user, system, or entity attempting to access a resource. It ensures that only legitimate users are granted access.

Types of Authentication

1. **Password-Based Authentication**:
    - Users provide a username and password to verify their identity.
    - Examples: Login forms, SSH access.
2. **Biometric Authentication**:
    - Uses biological traits like fingerprints, facial recognition, or retina scans.
    - Examples: Smartphone fingerprint unlock, airport biometric verification.
3. **Token-Based Authentication**:
    - Involves generating a secure token (e.g., JWT) after successful login.
    - Tokens are sent with subsequent requests to verify identity.
4. **Multi-Factor Authentication (MFA)**:
    - Combines two or more authentication factors (e.g., password + OTP).
5. **Certificate-Based Authentication**:
    - Uses digital certificates to authenticate users or systems.
    - Common in enterprise systems and secure APIs.

---

3. What is Authorization?

Definition

Authorization is the process of determining what actions or resources a user is permitted to access after they have been authenticated. It enforces policies to control access.

Types of Authorization

1. **Role-Based Access Control (RBAC)**:
   - Permissions are assigned based on user roles.
   - Example: An admin role has more privileges than a regular user.
2. **Attribute-Based Access Control (ABAC)**:
   - Access is granted based on user attributes (e.g., department, location).
   - Example: Employees in "HR" can access payroll systems.
3. **Policy-Based Access Control (PBAC)**:
   - Centralized policies define access rules, often using external policy engines.
4. **Discretionary Access Control (DAC)**:
   - Resource owners control access.
   - Example: File permissions on a Linux server.
5. **Mandatory Access Control (MAC)**:
   - Access is strictly controlled by the system, not by resource owners.

---

4. Key Differences Between Authentication and Authorization

Aspect	Authentication	Authorization
Purpose	Verifies identity.	Determines access rights.
Question Answered	"Who are you?"	"What can you do?"
Process	First step in access control.	Second step after authentication.
Focus	User identity.	User permissions.
Examples	Password login, biometric scan.	Accessing admin dashboard, editing a file.

Dependencies	Independent of authorization.	Dependent on successful authentication.

---

5. Importance of Authentication and Authorization in System Design

1. **Security**:
   - Prevents unauthorized access to sensitive data.
   - Protects against attacks like credential stuffing or privilege escalation.
2. **Compliance**:
   - Meets regulatory requirements (e.g., GDPR, HIPAA).
3. **Scalability**:
   - Ensures secure access as systems scale to support more users.
4. **User Experience**:
   - Properly implemented authentication and authorization provide seamless and secure user interactions.

---

6. Authentication Mechanisms in System Design

1. Password-Based Authentication

- Users provide credentials stored securely (e.g., hashed with bcrypt).
- Risks: Susceptible to brute-force attacks, weak passwords.

2. Token-Based Authentication

- After login, a token (e.g., JSON Web Token) is issued to the user.
- Tokens are sent with subsequent requests for verification.
- Benefits:
  - Stateless.
  - Ideal for distributed systems.
- Examples: OAuth2, OpenID Connect.

---

3. Multi-Factor Authentication (MFA)

- Combines:
    1. **Knowledge** (e.g., password).
    2. **Possession** (e.g., phone for OTP).
    3. **Inherence** (e.g., fingerprint).
- Significantly enhances security.

---

4. Biometric Authentication

- Uses unique physical traits.
- Benefits: Difficult to forge.
- Examples: Apple Face ID, fingerprint scanners.

---

7. Authorization Mechanisms in System Design

1. Role-Based Access Control (RBAC)

- Assigns permissions based on roles.
- Example:
    - Admin: Full access.
    - Editor: Create and edit.
    - Viewer: Read-only.

---

2. Attribute-Based Access Control (ABAC)

- Uses dynamic attributes to determine access.
- Example: A manager in "Sales" can access sales reports for their region.

---

3. Policy-Based Access Control (PBAC)

- Centralized access policies stored in external engines.
- Example: AWS IAM policies.

---

8. Best Practices for Designing Authentication and Authorization

For Authentication

1. **Use Secure Password Storage**:

    - Hash passwords using algorithms like bcrypt or Argon2.
2. **Implement MFA**:
    - Add an additional layer of security.
3. **Token Expiry**:
    - Set expiration times for session tokens.

For Authorization

1. **Follow the Principle of Least Privilege**:
    - Grant only the necessary permissions.
2. **Audit Permissions Regularly**:
    - Remove unused roles or excessive privileges.
3. **Externalize Authorization Logic**:
    - Use dedicated policy engines for scalability.

---

9. Challenges in Implementing Authentication and Authorization

1. **Scalability**:
    - Handling millions of authentication requests in distributed systems.
2. **Security Risks**:
    - Protecting against attacks like session hijacking or privilege escalation.
3. **User Experience**:
    - Balancing security with ease of use.
4. **Integration Complexity**:
    - Integrating authentication and authorization mechanisms across services.

---

10. Conclusion

Authentication and authorization are foundational components of system security. While authentication ensures only legitimate users access the system, authorization determines their privileges. Together, they safeguard sensitive resources and ensure seamless operation.

Understanding the differences, mechanisms, and best practices for implementing these processes is crucial for designing secure, scalable, and user-friendly systems. As systems grow increasingly distributed and complex, robust authentication and authorization mechanisms become indispensable.
