Identity and Access Management(IAM)

Created by Mehrnoush

- **Provisioning**
  - Identity Proofing and Verification
    - Purpose
      - Associating online activities with a real-life individual.
      - Ensuring compliance with regulations (e.g., financial, accountability).
      - Supporting secure access to sensitive or high-risk services (e.g., government, financial).
      - Mitigating fraud through reliable identity verification.
    - Expected Outcomes
      - Identity Resolution: Unique identification within the user population.
      - Evidence Validation: Verifying the authenticity of identity evidence.
      - Attribute Validation: Confirming core attribute accuracy.
      - Identity Verification: Confirming ownership of identity evidence.
      - Identity Enrollment: Registering verified individuals with the CSP.
      - Fraud Mitigation: Detecting and responding to fraudulent attempts.
    - Privacy & Usability Principles
      - Good Usability Practices
        - Minimize burden on applicants while ensuring outcomes.
    - Identity Assurance Levels (IALs)
      - No Identity Proofing
        - No real-life association required; self-asserted attributes.
      - IAL1
        - Core attribute validation.
        - Remote or on-site proofing.
        - Protection against synthetic identities and compromised PII.

- IAL2
  - Additional evidence collection and rigorous validation.
  - Protection against evidence theft and social engineering.
- IAL3
  - Direct interaction with a trained proofing agent.
  - Collection of biometric evidence.
  - Protection against advanced falsification and repudiation.

- Identity Proofing Types
  - Remote Unattended
    - Fully automated processes.
    - Devices not controlled by CSP.
  - Remote Attended
    - Secure video interaction with a proofing agent.
    - Devices not controlled by CSP.
  - Onsite Unattended
    - Automated kiosk or workstation.
    - Devices controlled by CSP.
  - Onsite Attended
    - In-person interaction with a proofing agent.
    - Devices and location controlled by CSP.

- Identity Proofing and Enrollment Steps
  - Resolution
    - Collect identity evidence and attributes.
    - Determine uniqueness within the population served.
    - Use minimal attributes to ensure uniqueness.
    - Cross-check against CSP's records.
    - Detect potential fraud at the initial step.
  - Validation
    - Collect appropriate evidence.

- Confirm evidence is genuine, accurate, and valid.
- Use FAIR, STRONG, or SUPERIOR evidence strength requirements.
- Evidence Strength Requirements
  - **FAIR Evidence**:
    - Issued by an authoritative source.
    - Includes name and unique identifiers.
    - Contains security features to prevent forgery.
  - **STRONG Evidence**:
    - Procedures overseen by accountable institutions.
    - Includes biometric characteristics.
    - Physical or digital security features.
  - **SUPERIOR Evidence**:
    - Cryptographically protected attributes.
    - Attended enrollment with issuing source.
    - Highly resistant to copying or fraud.

- Evidence and Attribute Validation
  - Confirm authenticity, accuracy, and validity.
  - Validate against authoritative or credible sources.
  - Use approved methods (e.g., visual inspection, cryptographic checks).
  - Document validation sources and processes.
  - Validation Sources
    - **Authoritative Sources**:
      - Direct issuers (e.g., DMV, SSA).
      - Sources with verified access to issuer data.
    - **Credible Sources**:
      - Entities with access to authoritative data.
      - Subject to regulatory oversight (e.g., credit bureaus).
- Verification
  - Confirm applicant ownership of identity evidence.

- Use approved verification pathways (e.g., biometrics).
- Verification Methods
  - **Confirmation Code Verification**:
    - Demonstrate control of identity evidence by returning a confirmation code.
  - **Authentication and Federation Protocols**:
    - Demonstrate control of a digital account or signed assertion.
  - **Micro Transactions**:
    - Use micro transaction values to demonstrate control of identity evidence.
  - **Onsite Attended Visual Facial Image Comparison**:
    - Proofing agent visually compares the applicant's face with the identity evidence.
  - **Remote Visual Facial Image Comparison**:
    - Conducted via video or photograph.
    - Proofing agent may interact during or after the event.
  - **Automated Biometric Comparison**:
    - Uses automated algorithms (e.g., facial recognition, fingerprints).
    - Compares live samples with identity evidence.
- Exclusions
  - Knowledge-Based Verification (KBV) is not allowed.
- Enrollment
  - Notify applicant via validated address.
  - Establish subscriber account.
  - Bind authenticators to the proven identity.
- Identity Proofing Roles
  - Proofing Agent
    - Visually inspects evidence.
    - Makes limited risk-based decisions.
  - Trusted Referee

- Handles exceptions (e.g., missing evidence).
- Detects deception and social engineering.
- Requires extensive training.

- Applicant Reference
  - Vouches for identity or context.
  - Does not directly act in the proofing process.
- Process Assistant
  - Provides accessibility or translation support.
  - Does not make decisions or evaluate risk.

- Core Attributes
  - First Name, Middle Name, Last Name.
  - Government Identifier (e.g., SSN, Driver's License #).
  - Physical or Digital Address for communication.
  - Additional attributes as required (documented and privacy-assessed).
- Equity Considerations
  - General Principles
    - Assess user population for inequities in access or outcomes.
    - Implement proactive monitoring and mitigation strategies.
  - Identity Resolution and Validation
    - Flexible naming conventions.
    - Address inconsistencies in identity records with trusted referees.
  - Identity Verification
    - Use equitable facial recognition technologies.
    - Provide alternatives for applicants with demographic or religious barriers.
  - Accessibility
    - ADA-compliant facilities and technology.
    - Process assistants for individuals with disabilities.

- Identity Management Approaches
  - Per-Application Identity Silo

- Centralized User Repository
- Early SSO Servers
- Federated Identity and SAML 2
  - Identity Federation Concepts
    - Sharing identity information across trusted domains
    - Use cases for federation
  - Federation Protocols
    - SAML 2
    - OpenID Connect
- OpenID
  - User-centric Identity
- OAuth 2
- OpenID Connect
  - Provide information in a standard format to applications about identity of an authenticated user.
- OAuth 2.1
- Directory Services
  - LDAP (Lightweight Directory Access Protocol)
  - Active Directory
  - Importance: Foundational for storing and retrieving identity information
- Decentralized Identity

  Decentralized Identity, also known as Self-Sovereign Identity (SSI), is a framework that allows individuals to have control over their digital identities without relying on centralized authorities or intermediaries. It uses blockchain or distributed ledger technology (DLT) to enable trust, privacy, and security.

  - Key Principles of Decentralized Identity
    - 1. User Control:
      - Individuals fully own and control their digital identities.
      - Users decide who can access their data and revoke access at any time.
    - 2. Privacy:

- Data minimization through selective disclosure.
- Users share only the necessary information without exposing their entire identity.

3. Interoperability:

- Identities can work across multiple platforms and systems.
- Open standards enable seamless integration.

4. Security:

- Data is stored in a cryptographically secure manner.
- Eliminates reliance on centralized identity stores, reducing risks of breaches.

5. Decentralization:

- Identity is verified and managed using decentralized technologies like blockchain.
- No single point of failure or control.

How Decentralized Identity Works

1. Decentralized Identifiers (DIDs):

- Unique identifiers owned by the user, typically stored on a blockchain or other distributed system.
- Example: `did:example:123456789abcdefghi`.

2. Verifiable Credentials (VCs):

- Digitally signed credentials issued by trusted parties (e.g., government, university, employer).
- Users store these credentials in their digital wallets.

3. Digital Wallets:

- Secure applications that store DIDs and Verifiable Credentials.
- Enable users to manage their identity and share it selectively.

4. Blockchain as a Trust Layer:

- Used for registering DIDs and verifying public keys.
- Provides a tamper-proof and transparent record.

5. Selective Disclosure:

- Users share only specific attributes (e.g., "I am over 18") without revealing unnecessary information.

- Benefits of Decentralized Identity
  - 1. Enhanced Privacy:
    - Reduces data exposure and minimizes the risk of identity theft.
  - 2. User Empowerment:
    - Users are no longer dependent on centralized entities to manage their identities.
  - 3. Interoperable Ecosystem:
    - A single identity can be used across multiple services and platforms.
  - 4. Security and Resilience:
    - Eliminates centralized honeypots of personal data, reducing risks of breaches.
  - 5. Cost Efficiency:
    - Reduces the operational costs for organizations by offloading identity management to users.
- Challenges of Decentralized Identity
  - 1. Adoption Barriers:
    - Requires widespread adoption across organizations and governments for full effectiveness.
  - 2. Usability:
    - Managing digital wallets and private keys can be complex for non-technical users.
  - 3. Trust in Issuers:
    - Users and systems must trust that credential issuers (e.g., government or institutions) are legitimate.
  - 4. Legal and Regulatory Issues:
    - Varies across jurisdictions; lacks unified global frameworks.
  - 5. Recovery Mechanisms:
    - Losing access to private keys can result in the permanent loss of identity.
- Use Cases for Decentralized Identity

- 1. Identity Verification:
  - Verifying identity for financial services (e.g., KYC for banks).
- 2. Healthcare:
  - Sharing medical credentials securely with healthcare providers.
- 3. Education:
  - Issuing tamper-proof degrees or certifications.
- 4. Workforce and Employment:
  - Employers can verify work history or credentials directly from employees.
- 5. Travel and Border Control:
  - Simplifying visa or passport verification using verifiable credentials.
- 6. E-Government Services:
  - Citizens can use a single identity for accessing various government services.
- 7. Decentralized Finance (DeFi):
  - Establishing trust in blockchain-based financial ecosystems.

- Key Technologies Used in Decentralized Identity
  - 1. Blockchain / Distributed Ledgers:
    - Acts as a registry for DIDs and public keys.
  - 2. Zero-Knowledge Proofs (ZKPs):
    - Enables users to prove certain attributes without disclosing the actual data.
  - 3. Cryptography:
    - Public-private key pairs secure user data and communications.
  - 4. Open Standards:
    - Standards like W3C's DID and Verifiable Credentials ensure interoperability.
- Decentralized Identity Ecosystem Players
  - 1. Identity Foundations and Standards Bodies:
    - .W3C DID Specification https://www.w3.org/TR/did-core/
    - .Decentralized Identity Foundation (DIF) https://identity.foundation/

- 2. Blockchain-Based Identity Projects:
  - Sovrin
  - Hyperledger Indy
  - uPort
  - Microsoft Azure Decentralized Identity
- 3. Technology Providers:
  - Evernym
  - Civic
  - IBM Blockchain Identity
- 4. Governments:
  - Some governments are exploring decentralized identity for national ID systems.

- Comparison Between Traditional and Decentralized Identity
  - Traditional Identity
    - Ownership
      - Controlled by organizations
    - Privacy
      - Limited, data shared widely
    - Security
      - Vulnerable to breaches
    - Interoperability
      - Siloed across platforms
    - User Control
      - Minimal
  - Decentralized Identity
    - Ownership
      - Owned by individuals
    - Privacy
      - Enhanced, selective disclosure
    - Security

- Secure, no central point of failure
- Interoperability
  - Works across platforms
- User Control
  - Full
- Future of Decentralized Identity
  - Global Adoption:
    - As more organizations adopt decentralized identity, it may become the standard for identity verification.
  - Integration with IoT:
    - Decentralized identity can provide secure identity management for IoT devices.
  - AI and Privacy:
    - Decentralized identity frameworks can help users manage AI-driven systems while preserving privacy.
  - Regulatory Harmonization:
    - Governments and international organizations are expected to work on creating compatible frameworks for decentralized identity.

- Identity Provisioning
  - Provisioning Options
    - Invite-Only Registration
    - Identity Migration
      - Bulk Migration
      - Gradual Migration of Users
  - Identity Proofing and Verification
    - Document Verification
    - Knowledge-Based Verification
    - Biometric Verification
  - Administrative Account Creation
    - Manual Account Creation
    - Automated Account Creation

- Cross-Domain Account Creation
- Leverage Existing Identity Services

- Selecting an External Identity Service
  - Self-Registered Identities
  - Organizational Identities
  - Government Identities
  - Industry Consortium Identities
- Identity Provider (IdP) Selection
  - Customer Types
    - B2C
    - B2E
    - B2B

Authentication

- OpenID Connect
  - Overview
    - Purpose: Adds an identity layer to OAuth 2.0 to facilitate user authentication.
    - Problem to Solve: Standardizes how applications authenticate users and receive user information in a secure, interoperable way.
  - Key Concepts
    - Roles
      - End User: The user who needs to authenticate.
      - OpenID Provider (OP): The server that authenticates the user and provides identity information.
      - Relying Party (RP): The application that relies on the OP for authentication.
    - Client Types
      - Public Clients
      - Confidential Clients
    - Tokens

- ID Token: A JWT containing claims about the authentication event and the user.
- Access Token
- Refresh Token
- Authorization Code

- Endpoints
  - Authorization Endpoint: Where the authentication request is sent.
  - Token Endpoint: Exchanges authorization codes for tokens.
  - UserInfo Endpoint: Provides additional user information.

- How It Works
  - 1. Authentication Request: RP redirects the user's browser to the OP with an authentication request.
  - 2. User Authentication: OP authenticates the user, possibly asking for consent.
  - 3. Token Issuance: OP issues an ID Token (and optionally access and refresh tokens) back to the RP.
  - 4. UserInfo Retrieval: RP can use the access token to request additional user claims from the UserInfo endpoint.

- ID Token
  - Structure: Consists of a header, payload (claims), and signature.
  - Key Claims:
    - iss: Issuer identifier (OP).
    - sub: Subject identifier (user).
    - aud: Audience (RP's client ID).
    - exp: Expiration time.
    - iat: Issued at time.
    - auth_time: Time of user authentication.
    - nonce: Mitigates replay attacks.
    - amr: Authentication methods used.
    - acr: Authentication context class reference.

- Flows

- Authorization Code Flow
  - Suitable For: Confidential clients.
  - Process:
    - 1. RP redirects user to OP for authentication.
    - 2. OP authenticates user and returns an authorization code.
    - 3. RP exchanges the code for tokens at the token endpoint.
  - Benefits: Tokens are obtained securely via back-channel.
- Implicit Flow
  - Suitable For: Public clients (less recommended due to security concerns).
  - Process:
    - 1. RP redirects user to OP for authentication.
    - 2. OP authenticates user and returns tokens directly in the redirect URI.
  - Considerations: Access tokens may be exposed in browser history or logs.
- Hybrid Flow
  - Combines: Features of both Authorization Code and Implicit flows.
  - Process:
    - 1. RP receives some tokens directly and others via token endpoint.
  - Use Case: Applications needing immediate ID Token but secure access token retrieval.
- UserInfo Endpoint
  - Purpose: Provides additional user attributes.
  - Access: Requires a valid access token.
  - Usage:
    - RP requests user information.
    - OP responds with user claims in a JSON object.---
- SAML 2
  - Overview
    - Cross-domain Single Sign-On (SSO)

- Identity Federation
- Enables centralized management of identities.

- Key Terminology
  - Subject
  - SAML Assertion
  - Identity Provider (IdP)
  - Service Provider (SP)
  - Trust Relationship
- How It Works
  - Metadata exchange between SP and IdP to establish trust.
  - SP-Initiated and IdP-Initiated SSO flows.
- Use Cases
  - Enterprise SSO
  - Identity Federation
  - Integrating legacy systems with modern protocols.---

- Risk-base Authentication
  - Explanation: Adjusts authentication requirements based on the assessed risk level of a login attempt.
    - Factors Considered:
      - Device recognition
      - Geolocation
      - IP address reputation
      - User behavior patterns
  - Adaptive Authentication Methods:
    - Increase authentication requirements for high-risk scenarios
    - Reduce friction for low-risk scenarios

- Stronger Authentication
  - Multi-Factor Authentication (MFA)
    - Methods:
      - SMS codes

- Authenticator apps
- Hardware tokens
- Biometrics (Fingerprint, Facial Recognition)

- Importance: Adds extra security by requiring additional verification factors.

- Biometric and Behavioral Authentication
  - Biometric Authentication
    - Fingerprint Recognition
    - Facial Recognition
    - Iris Scanning
    - Voice Recognition
  - Behavioral Authentication
    - Typing patterns
    - Mouse movements
    - Gait analysis

- Passwordless Authentication

  Passwordless authentication is a modern authentication method that eliminates the need for traditional passwords. Instead of passwords, it relies on alternative methods such as biometrics, one-time codes, or cryptographic keys to verify user identity. This approach enhances security, reduces friction, and mitigates issues associated with password management.

  - Key Features of Passwordless Authentication
    - 1. No Passwords Required:
      - Users do not need to create, remember, or manage passwords.
    - 2. Enhanced Security:
      - Removes vulnerabilities associated with weak or reused passwords.
      - Reduces risks of phishing, brute force attacks, and credential stuffing.
    - 3. Convenience:
      - Simplifies login processes, improving user experience.
      - Eliminates password reset requirements.
    - 4. Multi-Factor Compatibility:

- Often integrated with additional factors for increased security.

- How Passwordless Authentication Works
  - 1. User Registration:
    - The user sets up an authentication method (e.g., biometrics, email-based verification, or a security key).
    - The system links this method to the user's identity.
  - 2. Authentication Process:
    - When logging in, the user is prompted to verify their identity using a pre-configured method.
    - A secure token or session is created upon successful authentication.
  - 3. Token-Based Verification:
    - Tokens (e.g., JWT or SAML) authenticate users across sessions or services.

- Common Methods of Passwordless Authentication
  - 1. Biometric Authentication:
    - Uses unique physical or behavioral characteristics, such as:
      - Fingerprint Scanning.
      - Facial Recognition.
      - Voice Recognition.
      - Iris or Retinal Scanning.
  - 2. One-Time Passwords (OTPs):
    - Temporary codes sent via:
      - Email.
      - SMS.
      - Authenticator Apps.
  - 3. Magic Links:
    - A unique, time-limited link sent to the user's email for authentication.
  - 4. Push Notifications:
    - A notification sent to a trusted device for user approval.
  - 5. Security Keys:

- Physical devices like YubiKeys or FIDO2-compliant keys used for authentication.
- Often based on public-key cryptography.

6. Device-Based Authentication:
   - Leverages pre-registered trusted devices (e.g., a smartphone or laptop).
7. QR Code Scanning:
   - Users scan a QR code with a trusted device to authenticate.

Benefits of Passwordless Authentication

1. Increased Security:
   - Eliminates risks related to password compromise (e.g., phishing, brute force attacks).
   - Uses methods like cryptographic keys that are inherently more secure.
2. Improved User Experience:
   - Reduces friction by removing the need to remember passwords.
   - Simplifies the login process.
3. Reduced Costs:
   - Decreases IT support costs associated with password resets and account recovery.
4. Scalability:
   - Works well in both consumer-facing applications and enterprise environments.

Challenges of Passwordless Authentication

1. Implementation Complexity:
   - Requires updates to existing authentication systems.
   - Organizations must ensure compatibility with diverse platforms and devices.
2. Device Dependency:
   - Users may lose access if their registered device is lost or damaged.
3. Initial User Enrollment:
   - Migrating users to a passwordless system can be time-consuming.
4. User Trust:

- Users may hesitate to adopt new authentication methods without proper education.

- 5. Regulatory Compliance:
  - Some industries may have regulations that require additional measures for authentication.

- Passwordless Authentication Standards and Protocols
  - 1. FIDO2:
    - A standard developed by the FIDO Alliance for secure, passwordless authentication.
    - Combines WebAuthn (web standard for authentication) with CTAP (Client to Authenticator Protocol).
  - 2. WebAuthn:
    - W3C standard enabling passwordless authentication using public-key cryptography.
  - 3. OAuth 2.0 and OpenID Connect (OIDC):
    - Used for integrating passwordless methods into applications.
  - 4. PKI (Public Key Infrastructure):
    - Leverages certificates and keys for identity verification.
- Passwordless Authentication Use Cases
  - 1. Consumer Applications:
    - Social media, e-commerce, and financial platforms use passwordless login for convenience and security.
  - 2. Enterprise Environments:
    - Businesses implement passwordless authentication for employees to improve security and streamline workflows.
  - 3. Healthcare:
    - Enables secure, quick access to patient data without risking password breaches.
  - 4. IoT Devices:
    - Provides secure access to smart devices using biometrics or trusted devices.
  - 5. Education:

- Universities and schools use passwordless authentication for secure access to online portals and learning tools.

- Best Practices for Passwordless Authentication
  - 1. Implement Multi-Factor Authentication (MFA):
    - Pair passwordless methods with additional factors for sensitive applications.
  - 2. Educate Users:
    - Train users on how passwordless authentication works and its benefits.
  - 3. Offer Backup Options:
    - Provide fallback mechanisms (e.g., recovery email or customer support) in case users lose access to their primary method.
  - 4. Monitor and Audit:
    - Continuously monitor authentication activity to detect anomalies.
  - 5. Follow Open Standards:
    - Use established protocols like FIDO2 and WebAuthn to ensure security and interoperability.
- Popular Passwordless Authentication Providers
  - 1. Okta:
    - Offers passwordless solutions for enterprises using biometrics and push notifications.
  - 2. Auth0:
    - Provides passwordless login options such as magic links and SMS-based OTPs.
  - 3. Microsoft Azure Active Directory:
    - Supports passwordless authentication with FIDO2 security keys and biometrics.
  - 4. Ping Identity:
    - Enterprise-level passwordless authentication solutions.
  - 5. Yubico:
    - Manufactures FIDO-compliant security keys for passwordless login.
  - 6. Google Identity Platform:

- Integrates passwordless login options using magic links and device-based authentication.

- Future of Passwordless Authentication
  - Wider Adoption:
    - As security threats evolve, passwordless authentication is becoming a necessity.
  - Integration with Emerging Technologies:
    - Passwordless methods will likely integrate with AI, IoT, and blockchain.
  - Enhanced Biometric Systems:
    - Improved biometrics will make authentication faster, more secure, and accessible.
  - Regulatory Support:
    - Governments and industry bodies may introduce mandates or incentives for passwordless systems.

    Passwordless authentication represents a paradigm shift in securing digital identities. By eliminating the vulnerabilities of passwords, it enhances both security and usability, paving the way for a safer digital future.

- Other Authentication Protocols
  - Kerberos
    - Network authentication using secret-key cryptography
    - Use in enterprise environments
  - RADIUS (Remote Authentication Dial-In User Service)
    - Centralized authentication, authorization, and accounting
    - Use in network access control
  - TACACS+
    - Authentication for network devices
- Single Sign-On (SSO)

  Single Sign-On (SSO) is an authentication mechanism that allows users to access multiple applications or systems with a single set of credentials (username and password). It simplifies user experience by reducing the need to log in separately to each system while maintaining strong security.

  - Key Features of SSO

- Centralized Authentication:
  - Users authenticate once and gain access to multiple applications.
  - Reduces the need for repetitive logins.
- Improved User Experience:
  - Simplifies the login process by eliminating the need to remember multiple passwords.
  - Increases productivity by reducing login-related delays.
- Enhanced Security:
  - Reduces the risk of weak or reused passwords across applications.
  - Enables the implementation of stronger authentication mechanisms like Multi-Factor Authentication (MFA).
- Centralized Management:
  - Administrators manage authentication from a single platform.
  - Simplifies user access control and deprovisioning.

- How SSO Works
  - 1. User Requests Access:
    - The user attempts to access a protected resource (e.g., an application or service).
  - 2. SSO Server Authentication:
    - If not already authenticated, the SSO server prompts the user for credentials.
  - 3. Token or Ticket Generation:
    - Upon successful authentication, the SSO server generates a token or ticket.
  - 4. Application Validation:
    - The token is passed to the target application, which verifies it with the SSO server.
  - 5. Access Granted:
    - If the token is valid, the user is granted access without re-entering credentials.

- Benefits of SSO
  - 1. User Convenience:

- Reduces password fatigue by requiring a single set of credentials.

- 2. Improved Security:
  - Encourages strong, unique passwords for the single authentication point.
  - Integrates easily with MFA for additional security.
- 3. Centralized Administration:
  - Simplifies onboarding and offboarding of users.
  - Administrators can revoke access across multiple systems by disabling a single account.
- 4. Cost Savings:
  - Reduces IT support costs related to password resets.
  - Increases employee productivity.

- Challenges and Risks of SSO
  - 1. Single Point of Failure:
    - If the SSO system is compromised, all connected applications are at risk.
  - 2. Implementation Complexity:
    - Integrating multiple systems into an SSO framework can be complex.
  - 3. Dependency on the SSO Provider:
    - Organizations become reliant on the SSO solution's availability and security.
  - 4. Authentication Token Risks:
    - If the token is intercepted, an attacker could potentially gain access to all systems.
- Common SSO Protocols and Standards
  - 1. SAML (Security Assertion Markup Language):
    - XML-based standard for exchanging authentication and authorization data.
    - Commonly used in enterprise environments.
  - 2. OAuth:
    - Open standard for access delegation.
    - Often used for granting third-party apps limited access (e.g., "Log in with Google").

- 3. OpenID Connect (OIDC):
  - Layer built on top of OAuth 2.0.
  - Provides authentication as well as authorization.
- 4. Kerberos:
  - Network authentication protocol using tickets.
  - Commonly used in Windows Active Directory environments.

- SSO Implementation Scenarios
  - 1. Enterprise Systems:
    - Employees access corporate systems (e.g., HR software, email, CRM) with one login.
  - 2. Cloud Services:
    - Users log in to cloud-based services like Google Workspace, Microsoft 365, or Salesforce.
  - 3. Customer-Facing Applications:
    - Customers use a single login for multiple related platforms (e.g., an e-commerce site with a partner site).
- Best Practices for SSO Implementation
  - 1. Use Multi-Factor Authentication (MFA):
    - Enhance security by requiring an additional layer of authentication.
  - 2. Encrypt Authentication Tokens:
    - Protect tokens in transit and at rest to prevent interception.
  - 3. Implement Robust Logging and Monitoring:
    - Monitor SSO activity for suspicious behavior.
  - 4. Regularly Update and Patch the SSO System:
    - Protect against vulnerabilities and exploits.
  - 5. Plan for Failover:
    - Ensure high availability to mitigate the single point of failure risk.
- Popular SSO Providers and Tools
  - 1. Okta:
    - Cloud-based identity and access management platform.

- 2. Azure Active Directory (Azure AD):
  - SSO and identity management for Microsoft services and third-party applications.
- 3. Google Workspace SSO:
  - Provides SSO for Google services and integrated applications.
- 4. Auth0:
  - Developer-friendly identity and access management platform.
- 5. Ping Identity:
  - Enterprise-level identity security solutions.

Authorization

- Access Control Models

  Access control models are critical to maintaining data security and ensuring that only authorized individuals or systems can access resources. The choice of a model depends on organizational needs, security requirements, and the complexity of the system.

  - Mandatory Access Control (MAC)

    Mandatory Access Control (MAC) is a strict access control model where the operating system or administrator defines access policies. Users cannot alter access permissions.

    - Key Features:
      - Centralized control.
      - Access is based on classifications (e.g., Top Secret, Secret, Confidential).
      - Enforces security labels and clearance levels.
    - Examples:
      - Military systems.
      - Governmental classified systems.
  - Discretionary Access Control (DAC)

    Discretionary Access Control (DAC) allows the owner of a resource to determine access permissions for others.

    - Key Features:
      - Ownership-based access control.

- Flexible but less secure due to user-level control.

- Examples:
  - File systems where file owners set permissions (e.g., chmod in Unix).

- Role-Based Access Control (RBAC)

  Role-Based Access Control (RBAC) assigns access permissions based on roles within an organization.

  - Key Features:
    - Access depends on job responsibilities.
    - Simplifies administration.
    - Scalable for large organizations.
  - Examples:
    - Assigning "Manager" or "Employee" roles in enterprise systems.

- Attribute-Based Access Control (ABAC)

  Attribute-Based Access Control (ABAC) uses attributes (user, resource, and environmental) to grant access.

  - Key Features:
    - Fine-grained access control.
    - Supports complex policies.
    - Dynamic and context-aware.
  - Examples:
    - Allowing access based on user department, time, or location.

- Rule-Based Access Control

  Rule-Based Access Control defines rules and conditions for granting access, typically applied in conjunction with other models.

  - Key Features:
    - Policies are defined based on conditions (e.g., "Deny access after 6 PM").
    - Often implemented in firewalls or network systems.
  - Examples:
    - Firewall configurations blocking IP ranges.

- Time-Based Access Control

  Time-Based Access Control restricts access based on specific time constraints.

  - Key Features:
    - Useful for temporary access or scheduled tasks.
    - Time-based conditions (e.g., work hours, specific dates).
  - Examples:
    - Employee access systems restricting after-office hours.
- Break-Glass Access Control

  Break-Glass Access Control allows emergency access under special circumstances, bypassing regular restrictions.

  - Key Features:
    - Provides override mechanisms during emergencies.
    - Tracks and audits access during the process.
  - Examples:
    - Emergency access to patient records in healthcare systems.
- Identity-Based Access Control (IBAC)

  Identity-Based Access Control (IBAC) grants access based on individual identities.

  - Key Features:
    - Direct mapping of permissions to user identities.
    - Suitable for small systems with minimal users.
- Policy-Based Access Control (PBAC)

  Policy-Based Access Control focuses on policies to determine access, often used in cloud environments.

  - Key Features:
    - Centralized policy management.
    - Compatible with multi-tenant systems.
  - Examples:
    - Cloud-based IAM systems like AWS IAM.

- Hybrid Access Control

  Hybrid Access Control combines features of multiple models to meet specific needs.

  - Key Features:
    - Offers flexibility.
    - Balances security and usability.
  - Examples:
    - Combining RBAC with ABAC in enterprise systems.

- OAuth 2.0 - 2.1
  - Key Concepts
    - Roles
      - Resource Server
      - Resource Owner
      - Client
      - Authorization Server
    - Clients
      - Confidential
      - Public
    - Client Profiles
      - Web Applications
      - Browser-based Applications
      - Native Applications
  - Tokens
    - Authorization Code
    - Access Token
      - Opaque
      - JWT (JSON Web Token)
    - Refresh Token

- Authorization Grant Types
  - Authorization Code
    - Optimized for Confidential Clients
    - PKCE for public clients
  - Client Credentials
  - Refresh Token
  - Implicit
    - Deprecated in OAuth2.1
  - Resource Owner Password
    - Deprecated in OAuth2.1
  - Extension Grants (OAuth 2.1)
  - Device Code Grant
- Token Management
  - Access Tokens
    - Short-lived duration
      - Access tokens are valid for a limited time (e.g., minutes to hours).
      - Reduces the risk window if a token is compromised.
      - Encourages regular token renewal, ensuring up-to-date permissions.
    - Scoped privileges
      - Define the specific permissions and resources the token grants access to.
      - Use of scopes to enforce the principle of least privilege.
      - Examples of scopes: read, write, delete, or API-specific scopes like email, profile.
    - Resource-specific
      - Tokens can be restricted to specific resource servers.
      - Prevents tokens from being misused across different services.
      - Enhances security by isolating token usage to intended resources.
    - Sender-constrained

- Mutual-TLS
  - Provides a method for binding tokens to clients using client certificates.
  - Enhances security by ensuring that only the client with the corresponding private key can use the token.
  - Ideal for environments where clients can manage certificates securely.

  - Enhances security by requiring both client and server to present and verify each other's certificates during the TLS handshake.
  - Access tokens are bound to the client's TLS certificate, preventing token misuse if intercepted.
  - Requirements:
    - Clients must possess a valid X.509 certificate.
    - TLS connections must be established with mutual authentication.
  - Use Cases:
    - High-security environments where clients can securely store certificates.
    - Scenarios requiring strong client authentication (e.g., financial services).
- DPoP
  - Offers a lightweight alternative to mutual-TLS for public clients.
  - Allows clients to prove possession of a cryptographic key without needing a client certificate.
  - Improves security for clients that cannot securely store secrets or certificates.

  - A mechanism that allows clients without client certificates to bind tokens to a key pair they control.
  - DPoP Proof:
    - A JWT signed with the client's private key.
    - Included in the DPoP HTTP header of token requests and protected resource requests.
  - Process:
    - Token Request: Client sends a DPoP proof with the token request.

- Token Response: Authorization server issues a DPoP-bound access token.
- Resource Request: Client includes the DPoP proof and access token in resource requests.

- Benefits:
  - Mitigates token replay attacks by ensuring tokens can only be used by the client possessing the private key.
  - Suitable for public clients that cannot store secrets securely.
- Use Cases:
  - Single-Page Applications (SPAs).
  - Mobile applications.

- Refresh Tokens
  - Rotation
    - Each time a refresh token is used, a new one is issued (rotated).
    - Mitigates the impact of a stolen refresh token by invalidating old tokens.
    - Implemented via Refresh Token Rotation mechanisms.
  - Secure storage
    - Must be stored securely to prevent unauthorized access.
    - Confidential clients store tokens on secure servers.
    - Public clients should use secure storage solutions (e.g., Keychain on iOS, Keystore on Android).
  - Longer Lifespan
    - Have a longer validity period than access tokens.
    - Allow clients to obtain new access tokens without user intervention

- Security Measures
  - TLS enforcement
  - Token Revocation
  - Secure Token Validation
  - Sender-constrained
    - Mutual-TLS Client Authentication

- Binds tokens to the client's TLS certificate.
- Ensures only the client with the corresponding private key can use the token.
- Requires client certificates during the TLS handshake.

- DPoP (Demostration of Proof of Possession)
  - Allows clients to prove possession of a private key when using access tokens.
  - Involves sending a DPoP proof (a signed JWT) in the HTTP header.
  - Suitable for public clients like SPAs and Mobile apps.

Privileged Access Management (PAM)

- Definition
  - PAM refers to the strategies and tools used to control, monitor, and secure access to critical systems and sensitive information by privileged users.
- Key Objectives
  - Access Control
  - Audit and Monitoring
  - Risk Mitigation
  - Least Privilege Principle
- Core Components of PAM
  - Privileged Account Discovery
  - Credential Vaulting
  - Session Management
  - Access Workflow
  - Just-In-Time (JIT) Access
  - Password Rotation
  - Multi-Factor Authentication (MFA)
- Types of Privileged Accounts
  - Human Privileged Accounts
  - Non-Human Privileged Accounts
  - Shared Accounts

- Domain Administrator Accounts
- Superuser Accounts

- Benefits of PAM
  - Improved Security
  - Regulatory Compliance
  - Enhanced Accountability
  - Minimized Insider Threats
  - Reduced Attack Surface
- PAM Implementation Steps
  - Assessment
  - Centralized Vaulting
  - Policy Creation
  - Monitoring and Auditing
  - Integration
  - Training and Awareness
- Challenges in PAM
  - Complexity
  - Resistance to Change
  - Shadow IT
  - Legacy Systems
  - Credential Sprawl
- PAM Tools and Technologies
  - PAM Solutions
  - Session Monitoring
  - Automated Password Management
  - Privileged Threat Analytics
  - Integration with SIEM
- PAM in Cloud and Hybrid Environments
  - Cloud Privileged Accounts

- Hybrid Architecture
- Dynamic Scaling
- Cloud-Native Tools

- Best Practices for PAM
  - Enforce Least Privilege
  - Enable MFA
  - Use Time-Limited Access
  - Monitor Continuously
  - Password Hygiene
  - Conduct Penetration Testing
  - Automate Where Possible
- PAM and Compliance
  - Key Regulations
    - - GDPR: Protect access to sensitive personal data.
    - - HIPAA: Secure privileged access to health records.
    - - PCI DSS: Manage privileged accounts for payment systems.
  - Auditable Controls
- Future Trends in PAM
  - AI and Machine Learning
  - Zero Trust Integration
  - Cloud-Native PAM
  - Decentralized PAM
  - IoT and Edge Devices

Sessions

- A user's interaction with an application over a period of time is known as a session. Sessions help applications track authenticated users and maintain state across multiple requests.
- Types
  - Application Sessions
    - Session Identifiers

- Unique IDs representing user sessions
- Stored in cookies (web apps) or client-side storage (SPAs, native apps)

- Session Data Storage
  - Server-side: memory, filesystem, database, shared services (e.g., Redis)
  - Client-side: in-memory or local persistent storage (careful with sensitive data)
- Session Timeout
  - Idle timeout
    - Session expires after a period of inactivity
    - Protects against abandoned sessions being misused
  - Maximum session duration
    - Session expires after a fixed period, regardless of activity
- Session Renewal
  - Applications may renew sessions upon user interaction
  - Proactive prompts to users before session expiration
- User Experience Considerations
  - Balance between security and usability
  - Strategies to prevent loss of user work upon session expiration

- Identity Provider Sessions
  - Purpose
    - Maintain user authentication state across multiple applications
    - Facilitate Single Sign-On (SSO)
  - Mechanism
    - Identity Provider (IdP) creates a session with session data
    - Session information stored in cookies set by the IdP
  - Session Management
    - IdP uses cookies to recognize authenticated users
    - Applications redirect users to IdP for authentication or session checks
  - Session Renewal
    - Applications can redirect users to IdP to renew tokens or sessions

- Use of parameters like prompt and max_age in OIDC to control reauthentication

- Multiple Sessions
  - Existence at Different Layers
    - Users may have sessions with applications and identity providers
    - Possible layers:
      - Application Session
      - Authentication Broker Session
      - Remote Identity Provider Session
  - Interaction Between Sessions
    - Application session expiration may not affect IdP session
    - IdP session termination may not immediately end application sessions
  - Session Termination Scenarios
    - User logout
    - Session timeout
    - Administrator-initiated termination
    - Cookie deletion or server restarts
  - Considerations
    - Define impact of session termination at each layer
    - Plan for consistent user experience
- Session Duration
  - Idle Timeout
    - Based on inactivity period
    - Resets upon user activity in the application
  - Maximum Session Duration
    - Fixed maximum time for a session
    - Enhances security by requiring periodic reauthentication
  - User Experience
    - Warnings before session expiration
    - Options to extend session

- Balancing security needs with usability
- Factors Influencing Duration
  - Sensitivity of application and data
  - Device type (desktop vs. mobile)
  - User context (consumer vs. enterprise
- Session Renewal
  - Mechanisms
    - Redirect to Identity Provider
      - Application redirects user to IdP for session renewal
      - IdP may reauthenticate user or issue new tokens based on existing session
    - Silent Authentication
      - Use of hidden iframes to check session status (limited by browser policies)
    - Use of Authentication Parameters
      - OIDC prompt Parameter
        - Controls whether to force reauthentication (prompt=login) or suppress prompts (prompt=none)
      - OIDC max_age Parameter
        - Specifies maximum time since last authentication
        - IdP may prompt user to reauthenticate if exceeded
    - Alternative Methods
      - Some IdPs offer APIs to check session status without redirects
- Token Renewal
  - Need for Renewal
    - Access tokens and ID tokens have expiration times
    - Applications may need fresh tokens to continue operations
  - Methods
    - Refresh Tokens
      - Confidential clients can use refresh tokens to obtain new access tokens

- Public clients face challenges due to secure storage limitations
- Redirect to Identity Provider
  - Applications redirect users to IdP to obtain new tokens
  - If IdP session is valid, new tokens are issued without reauthentication
- Silent Authentication Challenges
  - modern browsers implement policies like SameSite cookies and block third-party cookies, affecting iframe-based silent authentication.
  - May not be reliable for token renewal
- Considerations for SPAs
  - Avoid storing long-lived tokens in browser storage
  - Use short-lived access tokens and redirect for renewal

- Reconstituted Sessions
  - Improving User Experience
    - Allow users to restore session state after expiration
    - Reduce disruption caused by session timeouts
  - Techniques
    - Session State Retention
      - Store session data securely even after session expiration
      - Restore data upon user reauthentication
    - User Reauthentication Flow
      - Prompt user to reauthenticate when session expires
      - After successful authentication, restore session state
  - Considerations
    - Set limits on how long dormant session data is retained
    - Ensure compliance with security policies

Logout

- 1. Definition
  - The process of terminating a user's active session in a system, application, or device.
  - Ensures the user is securely logged out to prevent unauthorized access.

- 2. Types of Logout
  - 2.1 Manual Logout
    - User explicitly clicks a "Logout" button or link.
    - Commonly used for web applications, mobile apps, and shared devices.
  - 2.2 Automatic Logout
    - Session is terminated automatically after a specific event or condition, such as:
    - Timeout: Inactivity for a predefined period.
    - Idle Session: No user interaction for a specific duration.
    - System Policy: Forced logout at a scheduled time (e.g., end of a workday).
- 3. Logout Mechanisms
  - 3.1 Server-Side Session Termination
    - Server destroys the session token or invalidates it.
    - Prevents reuse of tokens by attackers.
  - 3.2 Token Expiry
    - Tokens (e.g., JWT) are set to expire after a certain period.
    - Requires re-authentication for continued access.
  - 3.3 Cookie Deletion
    - Authentication cookies are deleted from the browser.
  - 3.4 Global Logout
    - Logs the user out of all devices and sessions simultaneously.
    - Useful for security-sensitive applications.
- 4. Logout Triggers
  - 4.1 User Action
    - User manually logs out from the application.
  - 4.2 Inactivity
    - Automatic logout due to inactivity for a specified duration.
  - 4.3 System Enforcement
    - Administrator-enforced logout due to policy updates or security concerns.
  - 4.4 Session Hijack Detection

- Logout triggered when suspicious activity is detected, such as login from an untrusted device or location.

- 5. Security Considerations
  - 5.1 Session Management
    - Ensure proper handling of session tokens.
    - Use secure cookies with `HttpOnly` and `Secure` flags.
  - 5.2 Token Revocation
    - Immediately revoke tokens after logout to prevent reuse.
  - 5.3 Multi-Factor Authentication (MFA)
    - Combine logout with MFA to ensure the user securely re-authenticates when needed.
  - 5.4 Logout Confirmation
    - Provide a confirmation dialog or message upon logout to avoid accidental logouts.
  - 5.5 Logout URL Protection
    - Protect logout endpoints from being abused (e.g., CSRF attacks).
- 6. User Experience (UX)
  - 6.1 Ease of Access
    - Place logout buttons in prominent locations.
    - Use recognizable icons (e.g., power button symbol).
  - 6.2 Feedback
    - Display a confirmation or success message after logout.
  - 6.3 Redirects
    - Redirect the user to a landing page or login screen post-logout.
  - 6.4 Remember Me Option
    - Provide a "Remember Me" option to balance convenience with security.
- 7. Logout in Multi-Session Systems
  - 7.1 Single Device Logout
    - Logs out only from the current device or session.
  - 7.2 Global Logout

- Ends all active sessions across all devices.
- Useful in scenarios of compromised accounts.

- 7.3 Selective Logout
  - Allows the user to choose specific sessions to log out from.

- 8. Logout in Single Sign-On (SSO)
  - 8.1 SSO Logout
    - Logs the user out of all applications connected via SSO.
  - 8.2 Challenges
    - Requires coordination between multiple systems.
    - May lead to partial logout if not properly implemented.
    - Single Logout (SLO) protocols exist but are not universally supported and can introduce complexity.
  - 8.3 Protocols
    - Logout mechanisms in SSO rely on standards like SAML and OpenID Connect.
- 9. Best Practices
  - Use secure session management techniques.
  - Implement user-friendly logout interfaces.
  - Ensure logout endpoints are secure from vulnerabilities (e.g., CSRF).
  - Provide visual feedback for successful logout.
  - Test for proper session termination across all platforms.
- 10. Common Issues
  - 10.1 Session Persistence
    - Tokens or cookies not cleared properly, leading to session continuation.
  - 10.2 Timeout Confusion
    - Users may not realize their session has expired due to inactivity.
  - 10.3 Logout Loops
    - Errors in session handling causing users to be repeatedly logged out.
  - 10.4 Partial Logout
    - Logging out from one system but remaining logged in on others.

- 11. Future Trends
  - Context-Aware Logout: Intelligent logout mechanisms that adapt based on user behavior or risk detection.
  - Biometric-Based Session Recovery: Using biometrics to re-authenticate after logout.
  - Enhanced Security Protocols: Improvements in SSO logout mechanisms and token revocation processes.
  - Decentralized Logout: Logout functionality integrated into decentralized identity systems.

Account Management and Recovery

- User Self-Service
  - Password resets
  - Account unlocks
  - Profile updates
- Delegated Administration
  - Allowing designated users to perform certain administrative tasks
  - Use cases for department managers, regional admin

Deprovisioning

- Explanation: The process of removing access rights and deactivating identities when they are no longer needed.
- Steps:
  - Access revocation
  - Account disablement
  - Data handling and retention policies
- Considerations:
  - Compliance with regulations
  - Ensuring that deprovisioned accounts cannot be reactivated without proper authorization
  - Timely deprovisioning to reduce the risk of unauthorized access by former employees or third parties.

Identity Governance and Administration (IGA)

- Explanation: Ensures that identities and access rights are managed properly throughout their lifecycle.
- Components:
  - Access Certification
  - Role Management
  - Segregation of Duties
  - Compliance Reporting

Audit and Compliance

- Explanation: Monitoring IAM activities for compliance and security.
- Components:
  - Audit Trails
  - Logging
  - Reporting Mechanisms
  - Compliance with Regulations
    - GDPR
    - HIPAA
    - SOX
    - PCI DSS