USER MANGEMENT

TITLE	User Management		
REVISION HISTORY			
Revision Number	**Date**	**Author(s)**	**Description**
0.1	11 Feb 2025	Rajashekar Tuppada	Initial Draft
0.2	19 Feb 2025	Rajashekar Tuppada	Updated Version
1.0	28 Feb 2025	Rajashekar Tuppada	Final Version

Note: Always refer to documents that are on-line. Using a hard copy or a local copy of this document is not recommended. When copying to the local disk or using a hard copy is unavoidable, ensure these are destroyed at the earliest. Always use only latest and approved copies.

A. ESTORE

1. ACCOUNT / USER PROFILE MANAGEMENT

- User management enables administrators to group users, define access policies, and secure digital assets
- It supports account operations such as registration, login, password reset, SSO, social login, and credential storage
- It cuts administrative costs significantly by reducing password reset issues and streamlining access management
- Modern systems integrate or utilize IAM with directory services to authenticate, authorize, and audit users
- On-premise identity providers historically managed user accounts, but cloud-based IAM offers greater scalability, flexibility, and remote access support
- Essential user management functions include on-boarding / off-boarding, role assignment, profile updates, audit trails, workflows, system integrations, notifications, and password policies
- User management has grown in complexity with multi-tenant architectures, multiple identity providers, SSO, social login, standard authorization models, and multi-tenancy
- Third-party and cloud tools are now essential due to the complexity of modern user management systems

User management is an organizational function that enables users to access and control digital assets, such as applications, devices, networks, and cloud services. Organizations are now exploring even more advanced solutions. Modern user management services provide end-to-end management of user accounts, including user registration, login and authentication, single sign-on (SSO), and permissions management.

1.1. AUTHENTICATION

- Authentication verifies a user's identity before granting access to systems or data
- It differs from authorization, which governs what a user can do after login
- Common methods include passwords, biometrics, MFA, and passwordless options
- Strong authentication reduces risk of breaches and account hijacking

User authentication is the foundation of secure access in every digital environment, from enterprise SaaS platforms to customer-facing mobile apps. At its core, it's the process of verifying that a user is who they claim to be before granting access to sensitive data, systems, or services.

* *SaaS - Software as a Service*

In 2025, authentication is more than just logging in with a password. With rising threats like phishing-as-a-service and identity sprawl, organizations need smarter ways to authenticate users, methods that balance a layer of security, usability, and developer flexibility. From MFA to passwordless logins and biometric scans, modern authentication systems must adapt to complex use cases without compromising user experience.

1.1.1. What is user authentication?

User authentication is the process of verifying a person's identity before granting access to a system, application, or resource. It's how systems confirm that a user is who they claim to be, typically by checking a combination of credentials like user IDs, passwords, biometrics, or authentication tokens.

Authentication is not the same as authorization. Authentication determines *who* the user is. Authorization determines *what* they're allowed to do once they've been authenticated, like viewing sensitive data, editing settings, or accessing admin-only dashboards.

Strong authentication is critical for modern access control. It protects user accounts, prevents unauthorized access, and helps organizations meet compliance requirements tied to privacy, data protection, and auditability. As part of a broader identity and access management (IAM) strategy, authentication verifies a user's identity, while authorization ensures users can only access applications and data they have permission to use.

1.1.2. Why does user authentication matter in 2025?

In 2025, user authentication is a critical defense against increasingly sophisticated threats.

Attackers are no longer relying on brute force alone. Tactics like phishing-as-a-service (where cybercriminals sell ready-made phishing kits) and credential stuffing (using stolen username-password pairs to gain access) are making traditional authentication methods less effective and more frustrating for users.

At the same time, enterprises are dealing with identity sprawl across cloud infrastructure, SaaS tools, mobile apps, and legacy on-prem systems. Every new tool introduces another layer of complexity—more endpoints to secure, more users to manage, and more policies to maintain.

This complexity creates tension. Security teams push for stronger controls. Product teams fight to preserve a seamless user experience. Developers are caught in the middle, expected to build systems that check every compliance box without slowing things down.

Authentication should empower teams, not delay them. When done right, modern auth not only protects sensitive data, it accelerates access, simplifies workflows, and keeps users (and developers) moving forward.

1.1.3. How is user authentication is evolving?

User authentication is shifting away from static credentials toward adaptive, context-aware methods.

In 2025, traditional authentication methods aren't enough. With SaaS sprawl, remote work, and mobile-first access becoming the norm, IT teams face rising pressure to secure a growing number of apps, devices, and APIs. Meanwhile, attackers are getting smarter: Phishing-as-a-service, brute force bots, and MFA fatigue are exposing the limits of outdated login systems.

Modern authentication must do more than verify credentials. It needs to provide secure access, support frictionless user experiences, and give developers tools that scale with complexity, not against it.

Legacy login methods typically rely on static passwords or basic two-factor authentication, making them vulnerable to phishing, credential stuffing, and reuse attacks. In contrast, modern approaches use adaptive, risk-based authentication, biometrics, and passkeys to strengthen security while streamlining the user experience.

1.1.4. How does user authentication work?

The user provides credentials, like a password, biometric scan, or security token, and the system verifies those credentials against a trusted source before granting access.

At its core, user authentication is a handshake between a person and a system. Here's an oversimplified version of an authentication flow:

1. User inputs credentials (e.g., email and password)
2. The system sends a request to an authentication service or identity provider (IdP)
3. The IdP checks the credentials against a user directory (like LDAP or Active Directory)
4. If the credentials are valid, the IdP returns a token or session confirmation

5. The user gains access to the application or resource

** LDAP - Light-weight Directory Access Protocol*
** idP - Identity Provider*

Modern authentication often replaces session-based logins with token-based authentication, where an encrypted token (such as a JWT) is issued once and used across services until it expires. In many systems, this token is stored in the user's browser or app and sent with each request to verify identity.

A token-based authentication flow typically goes like this:

1. User logs in with credentials
2. The system verifies the credentials with the IdP
3. If verified, the IdP issues a signed token (e.g., JWT)
4. The token is stored locally (e.g., in localStorage or a secure cookie)
5. With each new request, the token is sent in the HTTP header
6. The server validates the token before granting access to the requested resource

** JWT - JSON Web Token*
** HTTP - Hyper Text Transfer Protocol*

Passwordless authentication and biometric authentication are also growing in adoption. These approaches verify users based on something they are (like a fingerprint) or something they have (like a device), improving security while reducing friction.

Here's a simplified version of a passwordless authentication flow:

1. User enters their email or username
2. The system sends a one-time code, magic link, or push notification to a verified device or account
3. The user confirms their identity by entering the code, clicking the link, or approving the push notification
4. The system validates the action and grants access without requiring a password

All of this sits within the larger framework of IAM, which governs how users are authenticated, authorized, and monitored across systems. Authentication is the entry point; it's how secure access begins.

1.1.5. What are the different types of authentication factors

Modern authentication relies on a combination of factors to verify a user's identity. These factors fall into three main categories: something you know, something you have, and something you are. Combining multiple factors increases the level of security and reduces the risk of unauthorized access.

a. Knowledge Factors

These are credentials the user knows and can recall.

- **Examples:** Passwords, PINs, security questions
- **Vulnerabilities:** Susceptible to phishing, brute force attacks, and credential stuffing. Users often reuse weak passwords or choose easily guessed answers to security questions

b. Possession Factors

These are physical or digital items that the user has in their control.

- **Examples:** Mobile devices, hardware tokens, smart cards, one-time passwords (OTPs)
- **Vulnerabilities:** Can be lost, stolen, or compromised through SIM-swapping or token interception. However, they are significantly harder to exploit than knowledge factors alone.

c. Inherence factors

These are characteristics unique to the user.

- **Examples:** Fingerprint scans, facial recognition, voice patterns, typing behavior
- **Vulnerabilities:** Biometric data, once compromised, cannot be changed. Privacy concerns also arise when storing or processing biometric authentication data

Using a single factor, especially a knowledge-based one, is no longer considered sufficient for sensitive systems. Stronger security starts with MFA, combining at least two distinct categories to validate the authorized user.

1.1.6. User authentication methods

Authentication isn't one-size-fits-all. The right method depends on the level of security needed, the type of user, and the experience you want to deliver. Most teams now combine multiple authentication methods and give users flexibility in how they sign in—it's not always a matter of choosing one over another.

Below are the most common user authentication mechanisms used today, each with its own strengths, weaknesses, and use cases.

a. Password based authentication

The most familiar method. Users verify their identity by entering a username and a password. While easy to implement, this method is vulnerable to brute force attacks, phishing, and password reuse. Strong password policies (such as the requirement for numbers and special characters) and hashing help, but passwords alone no longer provide sufficient protection.

b. Token based authentication

Instead of entering credentials repeatedly, users authenticate once and receive a token, typically a JWT (JSON Web Token), that proves their identity on subsequent requests. Tokens are signed and time-bound, supporting scalable, secure sessions across APIs and web apps. Often used with OAuth 2.0 and OpenID Connect (OIDC).

c. Biometric authentication

Verifies identity using biometric data like fingerprint scans, facial recognition, or retina patterns. The inherence factor is highly secure and user-friendly, but it requires device-level support and raises privacy considerations.

d. Certificate-based authentication

Uses digital certificates to confirm identity, typically issued by a trusted certificate authority (CA). This method is difficult to forge and is often used in enterprise environments for device, user, or service account authentication. Supports secure access for internal networks and high-sensitivity systems.

e. Multi-factor authentication (MFA)

Combines two or more authentication factors: something you know (password), something you have (security token), and something you are (biometric). MFA dramatically reduces the risk of unauthorized access, especially when layered with device verification or location awareness.

f. Passwordless authentication

This option allows users to log in without entering a password. Instead, they use methods like biometric scans, passkeys, smart cards, or one-time codes sent to a trusted device. Powered by standards like WebAuthn, passwordless login reduces friction while improving security.

g. Single sign-on

SSO lets users access multiple applications with one login by sharing a token between a trusted IdP and the service provider. This simplifies the login experience while centralizing control. Often implemented using SAML, OIDC, or OAuth.

Authentication method comparison

Method	Security Level	UX Impact	Best Use Case
Password Based	Low	Moderate	Legacy systems, low-sensitivity apps
Token-based (OAuth/JWT)	High	High	APIs, SPAs, mobile apps
Biometric	High	Very high	Mobile apps, consumer logins
Certificate-based	Very high	Low (admin side)	Enterprise, internal tools
MFA	Very high	Moderate	Admin portals, finance, HR systems, SaaS
Passwordless	High	Very high	Modern SaaS, end-user platforms
SSO	High	Very high	SaaS suites, enterprise ecosystems

1.1.7. Authentication protocols developers should know?

Authentication relies on well-defined protocols to ensure secure, standardized identity verification across apps, devices, and networks. Here are the most common protocols every developer working with IAM should understand.

a. OIDC (OpenID Connect)

OpenID Connect (OIDC) is an identity layer built on top of OAuth 2.0 that enables applications to verify a user's identity and receive user profile information. It uses JSON Web Tokens (JWTs) and supports features like single sign-on (SSO), session management, and identity federation.

- **Best for:** SaaS apps, mobile apps, modern web apps
- **Why it matters:** OIDC is widely adopted and simple to integrate across a wide range of identity providers

b. SAML (Security Assertion Markup Language)

SAML is an XML-based open standard used primarily for SSO in enterprise environments. It allows identity providers to securely transmit authentication and authorization data to service providers.

- **Best for:** Enterprise SSO, HR and IT systems, B2B authentication
- **Why it matters:** SAML is battle-tested and highly secure, though more complex than newer protocols like OIDC

c. LADP (Lightweight Directory Access Protocol)

LDAP is a directory protocol used to store and query user credentials and organizational information. When integrated with systems like Active Directory, it supports centralized user management and legacy authentication processes.

- **Best for:** On-prem enterprise environments, internal tools
- **Why it matters:** Still common in hybrid IT ecosystems and is often used for initial user lookup during authentication

d. EAP (Extensible Authentication Protocol)

EAP is a flexible framework used in wireless and point-to-point network authentication. It supports multiple methods like OTPs, certificates, and smart cards, and is commonly deployed in secure wireless environments.

- **Best for:** Wi-Fi networks, VPNs, enterprise mobility
- **Why it matters:** EAP supports mutual authentication and encryption, essential for secure network access.

e. JWT (Jason Web Token)

JWTs are compact, URL-safe tokens that carry identity claims between parties. Often used with OIDC and OAuth 2.0, they allow stateless authentication across distributed systems.

- **Best for:** SPAs, micro-services, stateless auth flows
- **Why it matters:** JWTs are easy to parse, verify, and pass between systems without maintaining session state

f. CHAP vs PAP

PAP (Password Authentication Protocol) is basic and insecure. It transmits usernames and passwords in plaintext.

CHAP (Challenge-Handshake Authentication Protocol) is more secure, using a three-way cryptographic handshake to verify users.

- **Best for:** Understanding legacy systems or low-level protocol design
- **Why it matters:** These protocols laid the groundwork for more secure modern standards

1.1.8. Authentication strategies for APIs and micro-services

Modern application architectures often rely on APIs and micro-services to deliver flexibility and scalability, but they also introduce new authentication challenges. With dozens (or hundreds) of services communicating with each other, securing user and service identities becomes more complex than in traditional monolithic systems.

Key challenges include:

- **Service-to-service authentication:** Ensuring that internal services can securely verify each other without exposing sensitive credentials
- **Decentralized identity management:** Avoiding duplication of user data across micro-services
- **API security:** Protecting endpoints from unauthorized access and ensuring every request is validated

To address these, development teams typically choose from three architectural strategies:

1. Auth at each micro-service

   Each service handles authentication independently by validating tokens or credentials.

   - **Pros:** Simple to implement in isolated services
   - **Cons:** Not scalable. Auth logic is duplicated, security policies drift, and identity data must be shared across services

2. Central authentication service

   A dedicated micro-service handles all authentication. Other services forward auth requests to it.

   - **Pros:** Centralizes identity logic, reduces duplication
   - **Cons:** Adds latency, introduces coupling, and can become a bottleneck.

Authorization logic still needs to be distributed or centralized elsewhere

3. API gateway approach (recommended)

   A centralized API gateway handles user authentication at the edge. Authenticated requests are passed to internal services along with a validated token or security context.

   - **Pros:** Centralizes authentication, decentralizes authorization. Services can remain lightweight while trusting the gateway for identity enforcement
   - **Cons:** Requires investment in gateway infrastructure and secure token propagation

Regardless of approach, developers should follow best practices for secure token handling, including:

- Validating token integrity and expiration
- Using short-lived tokens for sensitive operations
- Monitoring token usage for suspicious activity
- Ensuring observability through request logging and identity tracing

For teams building and scaling micro-services, a gateway-first authentication strategy offers the best balance of security, performance, and developer control.

1.1.9. Best practices for stronger authentication

As authentication threats evolve, securing user access requires more than just checking a password. Below are the best practices every development and security team should follow to strengthen their authentication strategy in 2025 and beyond:

- **Use password-less authentication where possible.** Eliminate weak password dependencies by adopting passwordless authentication methods like biometric scans, hardware tokens, or passkeys via WebAuthn. These methods improve both security and user experience
- **Enforce multi-factor authentication (MFA).** Protect critical systems by requiring MFA, especially for admins, finance, and engineering roles. Use inherence, possession, and knowledge factors to make account compromise far less likely
- **Monitor and log all authentication attempts.** Track every login request, successful or failed, to detect suspicious activity like brute force attempts or login anomalies. Integrate with a SIEM or centralized logging solution for real-time threat visibility
- **Use short-lived, revocable tokens.** Limit token lifespans to reduce the risk of replay or token theft. Make sure your system supports token revocation in response to anomalies or policy violations
- **Enable account lockout thresholds.** Prevent brute force attacks by temporarily locking accounts after several failed login attempts. Use progressive delays or CAPTCHA challenges to deter automated attacks while minimizing user frustration
- **Regularly audit user permissions.** As roles evolve, outdated permissions become a security liability. Implement regular IAM reviews to ensure users only retain access to what they currently need
- **Never store user credentials in plain text.** Always use strong hashing (with salting) for password storage. Better yet, offload credential storage to trusted identity providers or use federated authentication protocols like OIDC or SAML

Strong authentication is about consistently applying best practices to reduce your attack surface while supporting secure, seamless access.

1.1.10. Secure access starts with smarter authentication

Authentication serves as the foundation of every user experience. The way you authenticate users shapes everything from onboarding and retention to data protection and compliance.

As threats grow more sophisticated and digital ecosystems become more complex, your authentication system needs to do more than verify credentials. It must adapt to new risks, support multiple identity flows, and empower teams to manage access without engineering friction.

1.2. SINGLE SIGN-ON

- **Simplify user access:** SSO lets users log in once to access multiple apps, reducing password fatigue and improving user experience
- **Boost security posture:** Centralized authentication allows stronger policies like MFA, session controls, and better auditing
- **Balance convenience and risk:** SSO reduces credential sprawl but can create a single point of failure if not properly secured
- **Choose the right protocol:** Understand the differences between SAML, OAuth, and OpenID Connect to select the best fit for your use case
- **Plan for fallback:** Always implement backup access methods and logging in case your identity provider becomes unavailable

Single sign-on (SSO) isn't just for convenience. It's a critical layer of security, usability, and scalability for modern applications.

In simple terms, SSO lets users log in once and access multiple apps or services without re-authenticating for each. It allows users to access different applications with one set of login credentials. Instead of managing separate identities for each app, users sign in once through a trusted identity provider (IdP) and gain secure access across systems.

But in 2025, SSO plays a bigger role: It reduces password fatigue, supports zero trust security strategies, and helps teams manage user identities across sprawling SaaS environments.

In this guide, we'll unpack how SSO works, explore core protocols like SAML and OAuth 2.0, and highlight why SSO is foundational to identity and access management (IAM) and overall security. You'll also see how ABC simplifies SSO implementation for SaaS teams so you can ship secure, user-friendly login flows without months of engineering overhead.

1.2.1. Why does SSO matter in 2025?

Single sign-on is becoming essential for modern organizations facing rising identity and security challenges. Key reasons include:

- **Improved security and usability:** SSO offers centralized access control, reducing login friction without compromising protection
- **Growing SaaS complexity:** As companies expand their software stacks, IT teams must manage user access across more tools, while minimizing risk and maintaining a seamless experience
- **Remote and hybrid work demands:** Employees now access dozens of systems from various devices, locations, and time zones, increasing the risk of identity sprawl
- **Rising threat of credential-based attacks:** Phishing, credential stuffing, and similar attacks are growing more sophisticated, putting pressure on traditional password-based systems
- **User fatigue from MFA:** Even with more convenient multi-factor authentication methods, like biometrics, constant prompting frustrates users and slows productivity

But here's what most organizations miss: SSO isn't just a tool for IT. At ABC, we believe identity should be a shared responsibility across teams. By democratizing power over access control, product, security, and customer-facing teams gain more autonomy and SSO becomes a force multiplier. Developers can focus on building, while the rest of the org can manage users and permissions without bottlenecks.

1.2.2. How does SSO work?

SSO establishes trust between applications and a centralized IdP. When a user first tries to access an app, the IdP steps in to authenticate them and issues a secure token that can be used across other trusted services.

Here's how the SSO process typically works:

1. **User initiates access.** The user opens a web application (often, the "service provider")
2. **Redirect to IdP.** The app redirects the user's browser to the identity provider for authentication
3. **Authentication request.** The user enters their user credentials (or uses multi-factor authentication) to prove their identity to the IdP
4. **Token issued.** Upon successful authentication, the IdP issues an authentication token (e.g., a SAML assertion or OIDC ID token)
5. **Token returned.** The token is sent back to the original application
6. **Access granted.** The application validates the token and grants the user access

Once authenticated, the user can access other applications in the trusted SSO network without logging in again.

Behind the scenes, this model reduces reliance on repeated credential entry, centralizes authentication control, and helps maintain session continuity, making it easier to manage user access at scale.

a. SSO authentication artifacts

In an SSO system, once a user logs in through an IdP, the system issues an authentication artifact (such as an SSO token, assertion, or ticket) that can be passed to other applications, allowing access without requiring another login. They're a core part of the trust relationship between the IdP and the service providers in an SSO system.

Two common types of authentication artifacts used in SSO are:

Access tokens: Used with OAuth 2.0 and OIDC, these are passed along with requests and grant access to protected resources.

SAML assertions: XML-based artifacts that pass authentication and authorization data between the identity provider (IdP) and the service provider (though not all of that data is necessarily intended for the service provider itself).

The token lifecycle includes several key steps:

- **Issuance:** After the user logs in, the IdP generates a token
- **Transmission:** The token is sent to the requesting application
- **Validation:** The application checks the token with the IdP or validates things like its cryptographic signature
- **Access:** If the token is valid, access is granted to the app or resource

These tokens are encrypted and time-bound to protect against unauthorized access. If compromised, they can be revoked or expire automatically, helping reduce security risks while keeping the login experience smooth for end users.

1.2.3. What are the nine benefits of SSO?

The key benefits of single sign-on are:

1. **Reduced password fatigue:** This is achieved by reducing the need to remember different usernames and passwords
2. **Faster logins:** Users can log in faster thanks to the reduced times needed to re-enter passwords whenever the same user logs in
3. **Less third-party risk:** SSO mitigates risks associated with accessing third-party websites (i.e., federated authentication) and avoiding the external storage and management of user passwords
4. **Lowered IT costs:** There is a big drop in the number of password reset and signing in calls to the IT help desk, thus lowering costs
5. **Simple admin:** SSO processes are transparent and involve the same tools used for other admin tasks
6. **Added control:** All network management data is in one repository, providing a unified source to control user privileges. Admins can easily modify access privileges throughout the network
7. **More productivity:** Users avoid disruptions and delays due to multiple logins or managing multiple passwords. They can access protected network areas easily and get back to work immediately
8. **Better network security:** Complex password management is a common cause of security breaches because users often write down their passwords. By consolidating the network access data, admins can confidently disable user accounts
9. **Network consolidation:** Admins can connect separate networks to consolidate their management efforts and oversee all diverse, distributed environments

1.2.4. Different types of SSO and key protocols

Different SSO protocols support different environments, from cloud-native SaaS platforms to legacy enterprise systems. Understanding these standards helps technical teams choose the right fit and gives decision-makers clarity on what's powering their access control.

a. SAML (Security Assertion Markup Language)

SAML is an XML-based open standard widely used in enterprise web applications. It enables secure communication between an IdP and a service provider by sending authentication and authorization data through browser redirects. It's especially common in B2B settings and supports federated identity management across organizations

b. OIDC (OpenID Connect)

OIDC is an identity layer built on top of OAuth 2.0, designed for modern web and mobile applications. It uses lightweight JSON tokens and supports features like social login and identity verification. OIDC is ideal for customer-facing SaaS apps where performance and scalability are critical

c. OAuth 2.0

OAuth 2.0 is an open standard for token-based authorization. While not an authentication protocol on its own, it plays a central role in issuing and managing access tokens. OAuth lets users grant limited access to apps without sharing their credentials, making it a key part of the SSO process

d. Kerberos

Kerberos is a ticket-based protocol commonly used in on-premises environments like Windows domains. It allows secure mutual authentication between users and services using a centralized ticket-granting server. While older, Kerberos remains essential in legacy enterprise IT infrastructures

e. LDAP/Active Directory

LDAP (Lightweight Directory Access Protocol) is a standard for accessing and managing directory information. When paired with Microsoft Active Directory, it provides centralized user identity storage and is often used to enable SSO across internal tools and systems. It remains popular in hybrid cloud or fully on-prem environments.

f. SSO protocol comparison

The table below compares the common SSO protocols described in this section for an at-a-glance view:

Protocol	Best for	Data format	Identity provider type	Modern relevance
SAML	Enterprise web apps	XML	Internal or federated IdPs	High
OIDC	Customer-facing SaaS	JSON (JWT)	External/social IdPs	Very high
OAuth 2.0	Authorization for APIs and third parties	JSON	Token services	Essential
Kerberos	On-prem Windows systems	Binary	Local servers	Niche
LDAP/AD	Internal corporate directories	Varies	Active Directory	Moderate

1.2.5. Social login vs. enterprise SSO

While social logins, using platforms like Google, Facebook, or LinkedIn as identity providers, can streamline sign-up and access for consumers, they come with tradeoffs that limit their usefulness in enterprise environments.

Social login pros include:

- **Fast onboarding:** Users can sign in using credentials they already remember
- **Convenience:** Reduces friction for accessing consumer-facing web applications
- **Widespread familiarity:** Many end-users are already accustomed to these flows

Social login cons include:

- **Single point of failure:** If an attacker compromises a user's social account, they may gain access to multiple connected applications
- **Limited access control:** Organizations can't easily manage user roles, permissions, or revoke access
- **Compliance gaps:** Social IdPs don't offer the same security logging, policy enforcement, or compliance readiness as enterprise SSO platforms

While social login simplifies access, it centralizes identity control under third-party providers, which can introduce vulnerabilities. As Jason Polakis, a security researcher at the University of Illinois at Chicago, explains, "In general, I'm against consumer SSO schemes because they not only present a single point of failure, but because they also enable additional attacks that are not feasible with traditional password-based authentication."

By contrast, enterprise SSO gives organizations more control over user authentication, permissions, and auditability. It connects apps to a centralized, trusted identity provider, typically using protocols like SAML or OIDC, and supports security features like multi-factor authentication (MFA) and role-based access control (RBAC).

For businesses that need to manage user identities securely at scale, enterprise-grade SSO is a security and compliance essential.

1.2.6. Implement SSO (in five steps)

Implementing SSO successfully starts with preparation and ends with a secure, seamless experience for users. Whether you're building a new application or modernizing internal systems, these five steps will help your team get SSO right:

1. **Audit application access needs:** Map out which applications your users need to access and group them by role, department, or use case. Identify login patterns and prioritize the tools that will benefit most from streamlined access
2. **Choose or configure your IdP:** Decide whether you'll use an internal IdP (like Active Directory) or integrate with a third-party provider. Ensure your chosen SSO solution supports open standards like SAML or OIDC to ensure compatibility across platforms
3. **Clean up user identity data:** Verify that all user accounts in your directory are up to date. This is the time to remove inactive users, align email addresses across systems, and standardize naming conventions to avoid mismatches during authentication
4. **Define roles and permissions:** Apply the principle of least privilege. Assign user access based on defined roles, ensuring that users can only access what they need. Document and test these roles before rolling out SSO broadly
5. **Ensure security and high availability:** SSO becomes a single point of entry, so uptime and protection are critical. If hosting on-premises, build in redundancy and monitoring. For cloud-based SSO, evaluate your provider's reliability, SLAs, and available security settings like MFA enforcement and IP restrictions

Done right, SSO boosts both security and productivity without slowing down development or overburdening IT

1.2.7. Common challenges with SSO

While single sign-on offers major benefits, it's not without its friction points. Organizations implementing SSO should be aware of the potential risks and roadblocks, most of which stem from configuration issues or a lack of planning.

- **Initial setup can be complex:** Integrating multiple applications with a single identity provider can require significant development and coordination, especially in

hybrid or legacy environments

- **SSO becomes a single point of failure:** If your identity provider or SSO service goes down, users may be locked out of all connected apps. Redundancy, failover planning, and SLA-backed providers are essential
- **Misconfigured tokens or mismatched directories:** If user identities in your app don't match those in the IdP, or if token claims are incomplete, users may be denied access unexpectedly
- **Limited visibility into login activity:** Some cloud platforms abstract IP address data or session context, making it harder for security teams to track where access is coming from
- **Security depends on strong implementation:** An insecure token strategy, weak MFA policy, or improper session management can create vulnerabilities. SSO should always be implemented with careful monitoring, auditing, and role-based access control

When done right, SSO is a win for both security and usability. But without proper planning, it can introduce new risks.

1.2.8. SSO is just the start of identity modernization

In 2025, SSO is foundational to every business. It simplifies access for users, reduces overhead for IT, and tightens security in a world where login credentials remain a top attack vector.

But SSO is just one part of a larger IAM strategy. As teams adopt more SaaS tools, support hybrid workforces, and prioritize zero trust security, modern IAM needs to do more than authenticate users; it needs to empower the entire organization to manage identity without friction.

1.3. PERMISSIONS MANAGEMENT

Securing access to software applications and computer systems is essential across all industries. User roles and permissions play a pivotal role in this process by defining and controlling access levels for different users within a system. This article delves into the fundamentals of user roles and permissions, exploring their significance in ensuring data security, streamlining workflows, and providing tailored user experiences.

We'll also discuss how implementing robust role and permission management can help improve security, enhance efficiency, and allow better scalability, while keeping you compliant with privacy regulations. By understanding the importance of user roles and permissions, software engineers, DevOps teams, and product teams can better safeguard their applications against unauthorized access and foster more efficient, user-friendly environments.

1.3.1. Fundamentals of User Roles

A user role represents a specific set of tasks or responsibilities assigned to a group of users within an application. Assigning user roles enables organizations to control the actions each user can perform in the system, streamlining their workflow.

Common Examples of User Roles

- **Admin:** Users with this role usually have complete access rights, allowing them to manage all aspects of the application, such as adding new users, modifying settings, viewing and editing all content in the system
- **Supervisor:** This role might be responsible for managing content created by other users, but may not have full administrative privileges like creating new accounts or changing global settings
- **User:** This role might be able to create content and view their own content, but not view or modify content created by others

1.3.2. Understanding Permissions

While user roles that focus on group capabilities, permissions determine the specific actions individual users are allowed—or disallowed—to take when interacting with an app.

For instance, a permission could grant read-only access, enabling a user to view content but not modify it. Permissions are typically assigned to users based on their role, ensuring that each person has the appropriate access level for their job function.

Common Examples of Permissions

- **Create:** Allows users to add new content or resources
- **Read:** Permits users to view existing content without making changes
- **Update:** Grants permission for users to edit and make changes to existing content
- **Delete:** Provides authority for removing content from the system entirely

1.3.3. Significance of User Roles and Permissions

Here are some of the reasons user roles and permissions are critical for most organizations.

a. Improved Security

User roles and permissions are crucial for maintaining secure access control within an application. By assigning specific privileges to each user based on their role or function in the organization, you can limit unauthorized access to sensitive data or restricted areas. A well-designed permission model ensures that users only have access to the resources they need to perform their job duties, preventing potential security breaches caused by excessive privileges

b. Efficiency

A structured user role management system simplifies administrative tasks related to managing user access rights. Instead of individually configuring each user's permissions manually, administrators can assign predefined roles with associated privileges quickly, saving time on repetitive tasks and reducing the risk of human error during configuration changes

c. Greater Scalability

An effective user role permission model allows your application to scale smoothly as your organization grows or evolves over time. As new team members join or existing ones change positions within the company structure, adjusting their corresponding roles becomes more manageable. This flexibility enables businesses to adapt quickly without compromising security standards

d. Compliance Requirements

- **Data privacy regulations:** Compliance with data privacy regulations like GDPR, HIPAA, or CCPA often requires strict access control measures. Implementing user roles and permissions ensures that sensitive data is only accessible to authorized users
- **Audit trails:** A well-designed permission model enables comprehensive audit trails of user activity within the application. This information can be critical during internal audits or external regulatory inspections

e. Customization and Personalization

User roles and permissions enable a more personalized experience for end-users by tailoring their access based on individual needs. By granting specific privileges according to each role, you can create a customized environment where users have access to the most relevant resources, improving their user experience

1.3.4. Understanding User Role Permission Models

A user role permission model is a structure that organizes and controls user access rights within a system or application.

Key Components of a User Role Permission Model

- **User:** A person who interacts with the system, usually possessing a unique identifier (like a username) and authentication credentials (such as a password)
- **Role:** A set of predefined permissions related to specific tasks or responsibilities. Roles group users according to their job functions, streamlining permission management
- **Permission:** The authorization to perform specific actions within the system, such as viewing data, modifying records, or executing commands. Permissions are generally defined at different levels, ranging from broad categories to granular control over individual resources

Users, Roles, and Permissions: How They Relate

In a user role permission model, users are assigned one or more roles that determine their access level within the system. These roles contain sets of permissions that define what actions users can take. Assigning roles, rather than individual permissions, allows administrators to manage access rights efficiently across large groups.

To illustrate this concept, consider an online project management tool with distinct roles for Project Managers, Team Members, and Clients. In this scenario:

- Project Managers create and manage projects, assign tasks to Team Members, and communicate with Clients
- Team Members view project details, update task statuses, and collaborate with Project Managers
- Clients access only relevant projects without editing capabilities

By creating roles for each user type, administrators can assign appropriate permissions based on job functions. This approach simplifies permission management and ensures users have access to necessary features for their specific role.

Designing a User Role Permission Model

To design an effective user role permission model, follow these steps:

1. **Identify Application Resources**

   List all resources in your application that require access control, such as pages, APIs, data entities, or other components. Document each resource, its purpose, and associated actions (e.g., read, write, delete)

2. **Define User Roles**

   Group users into distinct roles based on their responsibilities and access needs. Common user roles include administrators, managers, contributors, and viewers

3. **Assign Permissions to User Roles**

   Assign permissions to each role based on required resource access. Use pre-built templates for common use cases or create custom rules based on unique requirements. Maintain detailed records of permissions granted to each role for developer reference.

   Use the principle of least privilege (POLP) to ensure that every user or system receives only the minimal privileges it needs to carry out its role

4. **Manage User Roles & Permissions**

   Update roles and permissions as needed to accommodate new features or changing business requirements. Regularly review and update your access control policies to ensure long-term security and efficiency

5. Security Auditing and Monitoring Access

Conduct routine security audits to verify the effectiveness of your role permission model. Auditing can highlight areas where permissions are overly broad or too restrictive, enabling you to fine-tune your model for optimal security and functionality.

Implement a monitoring system to track user activities in real-time, including when and how users access resources. These logs should record every successful and failed attempt to access a resource and any changes made to user roles or permissions. In case of security breaches, this will provide a traceable record of actions.

User management systems allow administrators to manage users' access to devices, software, and services. This includes managing permissions, monitoring usage, and providing authenticated access. User management is a core part of Identity and Access Management (IAM).

1.4. IDENTITY AND ACCESS MANAGEMENT

Identity and Access Management (IAM) is a framework of policies and technologies that ensures the right people have appropriate access to technology resources. IAM systems are designed to provide a means of identifying users in a system (e.g., a country, a network, or an enterprise) and controlling their access to resources within that system by associating user rights and restrictions with the established identity.

IAM is not just about controlling access. It's about ensuring that access is granted swiftly, efficiently, and securely. When implemented correctly, IAM systems provide a seamless user experience, enhance security, streamline administrative processes, and ensure compliance with regulatory mandates.

1.4.1. Why Is IAM Important?

a. Enhancing Security by Enforcing Authorized Access

As businesses become more digital, they also become more vulnerable to cyber threats. IAM plays a crucial role in protecting against these threats. By ensuring that only authorized users have access to systems and data, IAM helps to prevent unauthorized access and potential data breaches. IAM utilizes technologies such as multi-factor authentication and biometric verification to ensure that the person accessing the system is who they claim to be.

b. Improving User Experience by Easing Access

In the modern business environment, users expect seamless access to systems and data. They don't want to remember multiple passwords or go through complicated procedures to access the information they need.

IAM systems can provide single sign-on (SSO) solutions that allow users to log in once and have access to all the systems and applications they need. This not only improves the user experience but also enhances security by reducing the risk of weak or reused passwords.

c. Automating Administrative Processes

Managing user identities and access rights can be a complex and time-consuming task. It involves creating user profiles, assigning access rights, monitoring user activities, and deactivating user profiles when they are no longer needed.

IAM systems can automate these administrative processes, making them more efficient and less prone to errors. This not only saves time and resources but also ensures that access rights are managed consistently across the organization.

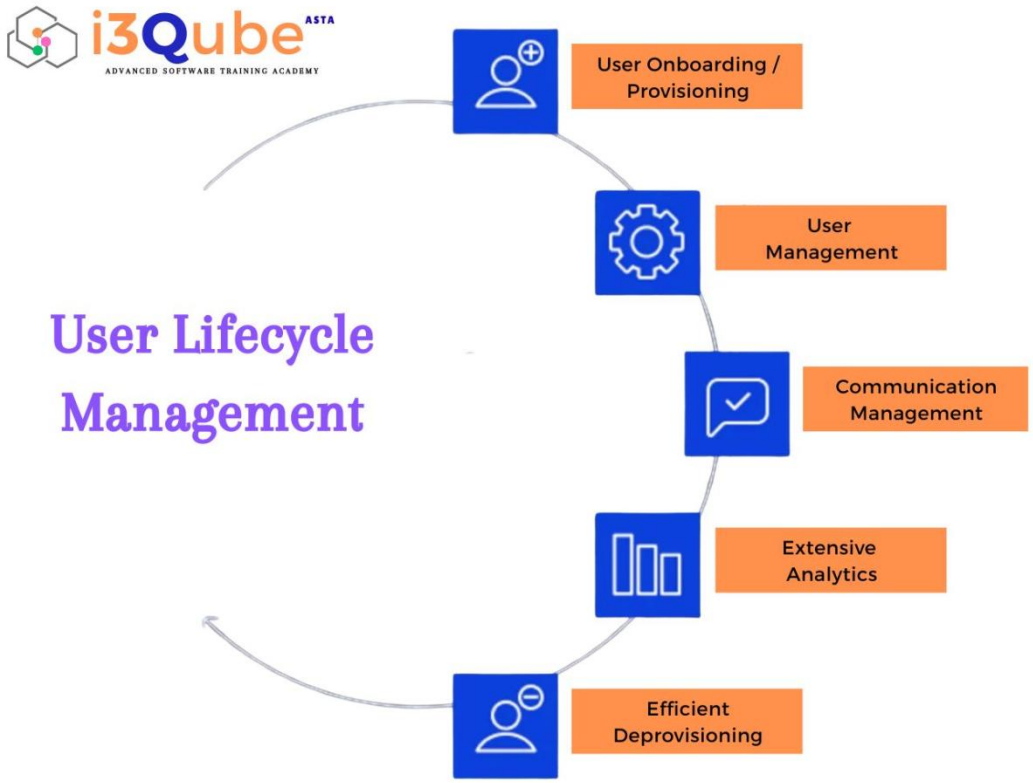
d. Supporting Compliance Efforts

Many industries, such as healthcare and finance, are subject to strict regulatory mandates regarding data security and privacy. Non-compliance can result in hefty fines and damage to the organization's reputation.

IAM systems can help organizations comply with these regulatory mandates by providing a transparent and auditable record of who has access to what, when, and why. This makes it easier for organizations to demonstrate compliance during audits and inspections.

1.4.2. IAM Lifecycle

The IAM is a structured process that starts when users initially receive access to resources, and ends when access is eventually revoked. Here are the main stages involved:



a. Provisioning

Provisioning is the process of creating and managing user identities and access rights. This involves creating user profiles, assigning access rights based on the user's role in the organization, and ensuring that these rights are updated as the user's role changes.

IAM systems can automate the provisioning process, ensuring that access rights are granted quickly and accurately. They can also integrate with HR systems to automatically update access rights when a user's role changes.

b. Authentication

Authentication is the process of verifying the user's identity.

Modern IAM systems use multi-factor authentication, which is done through a combination of something the user knows (e.g., a password), something the user has (e.g., a security token), and something the user has (e.g., a fingerprint). Combining two or more authentication factors ensures the person accessing the system is who they claim to be, and is a strong defense against social engineering attacks.

c. Authorization

Authorization is the process of verifying that the authenticated user has the appropriate access rights. This involves checking the user's access rights against the resources they are trying to access.

IAM systems can enforce fine-grained access control policies, ensuring that users have access only to the resources they need to perform their job. This helps to minimize the risk of unauthorized access, insider threats, and resulting data breaches.

d. Audits and Compliance

IAM makes it possible to monitor user activities and ensure compliance with regulatory mandates. This includes logging and analyzing user activities, detecting and responding to suspicious activities, and providing auditable records of who has access to what, when, and why.

IAM systems can provide comprehensive audit and compliance tools, making it easier for organizations to detect and respond to security incidents and demonstrate compliance during audits and inspections.

e. Deprovisioning

Deprovisioning is the process of revoking user identities and access rights when they are no longer needed. This is critical to prevent unauthorized access.

IAM systems can automate the deprovisioning process, ensuring that access rights are revoked promptly and accurately. They can also integrate with HR systems to automatically deprovision access rights when a user leaves the organization.

1.4.3. Challenges of IAM Implementation

Although IAM offers compelling benefits, it can be complex to implement IAM in a large organization. Some of the key challenges include:

a. Managing Growing Numbers of User Identities

As organizations expand and evolve, so does the number of user identities that require management. With every new employee, partner, or customer, comes an additional identity that needs to be created, managed, and eventually deactivated. This growing volume of identities can quickly become overwhelming, creating a significant administrative burden and increasing the risk of errors.

Moreover, each user identity represents a potential point of vulnerability. If not managed effectively, inactive or redundant accounts can provide an easy entry point for cybercriminals. Therefore, it's not just about managing the sheer number of identities, but also ensuring that each one is securely managed throughout its lifecycle.

b. Keeping up with Evolving Security Threats

Cybercriminals are constantly devising new methods of attack, and vulnerabilities can emerge from any number of directions.

For example, social engineering attacks such as phishing or spear-phishing can be particularly difficult to defend against, as they exploit human weaknesses rather than technical ones. Similarly, insider threats, whether malicious or accidental, can pose a significant risk to an organization's IAM strategy.

IAM systems must provide innovative security measures to address the latest threat vectors, and teams must activate them and learn to effectively use them.

c. Balancing User Convenience and Security

Another key challenge in IAM is striking the right balance between user convenience and security. On the one hand, strict security measures can help to minimize the risk of unauthorized access and data breaches. On the other hand, if these measures are too cumbersome or intrusive, they can impede user productivity and satisfaction.

For example, requiring users to remember complex, unique passwords for each system or application they use can lead to frustration and potentially risky behaviors, such as writing down passwords or using the same password across multiple systems. Similarly, overly strict access controls can prevent users from accessing the resources they need to do their jobs effectively.

d. Integrating IAM Solutions with Legacy Systems

Finally, integrating IAM solutions with legacy systems can pose a significant challenge. Many organizations are still reliant on older, monolithic systems that were not designed with modern IAM principles in mind. These systems can be difficult to integrate with newer IAM technologies and may lack the flexibility to support dynamic, policy-based access controls.

In addition, legacy systems often have their own, separate identity repositories, leading to a proliferation of identity silos across the organization. This not only increases the complexity of IAM but also the risk of inconsistent or outdated identity data.

1.4.4. IAM in the Cloud

Despite these challenges, the advent of cloud-based IAM solutions offers new opportunities for organizations to deploy and make use of IAM technology. Cloud-based IAM (also known as IAM-as-a-Service or IDaaS), can provide a scalable and cost-effective solution for managing and securing user identities.

One of the key advantages of cloud-based IAM is that it can help to simplify the management of user identities. Rather than having to manage separate identity repositories for each system or application, organizations can centralize their identity data in the cloud. This not only reduces the administrative burden but also improves the consistency and accuracy of identity data.

In addition, cloud-based IAM can provide advanced security features such as multi-factor authentication (MFA), risk-based authentication, and anomaly detection, all delivered as a service with no need to deploy software on-premises.

1.4.5. Key IAM Technologies and Tools

Here are some of the primary building blocks of an IAM solution. Depending on your use case, you might use some or all of these technologies.

a. Directory Services and Identity Providers (IdP)

Directory services, such as Microsoft Active Directory, provide a central repository for storing and managing user identity data. They play a crucial role in authenticating users and authorizing their access to resources.

In addition to storing user identities, directory services can also store information about resources, such as servers, printers, and network devices, and the relationships between them. This information can be used to enforce access control policies and to provide a comprehensive view of the organization's IT environment.

In recent years, directory services have evolved into Identity Providers (IdPs), systems that authenticate and authorize users by providing and managing digital identities. Many businesses use third-party IdPs, many of them cloud-based, which integrate with numerous applications, both cloud-based and on-premises. Modern IdPs support modern authentication capabilities such as SSO and MFA (see below).

b. Single Sign-On (SSO) Solutions

Single sign-on (SSO) allows users to log in once and gain access to multiple systems or applications, without having to re-enter their credentials. This not only improves user convenience but also reduces the risk of password-related security issues. There are various types of SSO solutions, including:

- **Web-based SSO:** Enables access to multiple web applications with a one-time login, primarily used for cloud-based or externally hosted services

- **Enterprise SSO:** Facilitates internal access to various on-premises systems using a single set of credentials, improving convenience for internal users
- **Federated SSO:** Extends single sign-on capabilities to external networks and domains, promoting secure and efficient inter-organizational collaboration

c. Multi-Factor Authentication (MFA)

Multi-factor authentication (MFA) is a security measure that requires users to provide more than one form of verification to prove their identity. This could involve something they know (like a password), something they have (like a security token), and something they are (like a fingerprint).

By requiring multiple forms of verification, MFA provides an added layer of security, making it more difficult for unauthorized users to gain access to resources. At the same time, advancements in MFA technology mean that it can be implemented in a way that is convenient for users, helping to maintain the balance between security and usability.

d. Privileged Access Management (PAM)

Privileged access management (PAM) helps control and monitor privileged users, who have administrative access to critical systems and data. PAM tools provide a secure way to authorize and monitor all privileged user activities. They also manage passwords for privileged accounts, automatically rotating and encrypting them to prevent unauthorized access.

Implementing PAM is important for sensitive and mission critical applications. It reduces the risk of data breaches caused by compromised privileged credentials, ensures compliance with regulatory requirements, and provides complete visibility into privileged activities.

e. Identity Governance and Administration (IGA)

Identity governance and administration (IGA) tools are responsible for managing digital identities, access rights, and compliance policies across the organization. They automate and streamline IAM processes, reducing manual effort and error.

IGA tools can automatically create, modify, and deactivate user accounts based on predefined policies. They help enforce the principle of least privilege (PoLP), ensuring users have only the access they need to perform their jobs. Also, they provide comprehensive reports and audits, making it easier to demonstrate compliance with regulations.

f. Consumer Identity and Access Management (CIAM)

Customer identity and access management (CIAM) is a subfield of IAM that focuses on managing and securing customer identities. It combines traditional IAM practices with customer relationship management (CRM) to deliver a secure and seamless user experience.

CIAM tools help businesses securely capture and manage customer identity and profile data, and control customer access to applications and services. They support social login, multi-factor authentication, and consent management, providing a secure and personalized customer experience.

Implementing a CIAM solution can improve customer engagement, increase conversion rates, and enhance security. It also helps businesses comply with privacy regulations like GDPR and CCPA.

g. Entitlement Management Software

Entitlement management software helps organizations manage user permissions and entitlements to ensure that users have the appropriate access to resources.

This software helps companies define, enforce, and review access policies, ensuring that only authorized individuals have access to sensitive data. It also provides robust auditing capabilities, allowing organizations to monitor access and detect any anomalies or unauthorized access.

The implementation of entitlement management software should be done in conjunction with a broader IAM strategy. This ensures that all aspects of identity and access management are covered, from the initial authentication process to the ongoing management of user permissions and entitlements.

1.4.6. Best Practices for Implementing IAM Solutions

a. Assess Needs and Capabilities

Before implementing an IAM solution, it's vital to conduct a thorough assessment of your organization's needs and capabilities. This involves understanding your existing infrastructure, applications, and data, as well as your business processes, security risks, and compliance requirements.

The assessment should identify who needs access to what, when, and why. It should also uncover any gaps in your current access controls and identity management practices. The insights gained from this assessment will guide your IAM strategy and help you choose the right IAM technologies and tools.

b. Centralized Identity Management

One of the key best practices in IAM is centralized identity management. This means managing all user identities and access rights from a single, centralized system.

Centralized identity management simplifies administration and improves security. It provides a single source of truth for user identities, reducing inconsistencies and errors. It also enables consistent enforcement of access policies across the organization.

Implementing centralized identity management requires careful planning and execution. It involves integrating all your applications and systems with the central IAM system, and migrating existing user accounts and access rights to the new system.

c. Implement Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is a method of managing access rights based on users' roles in the organization. In RBAC, access rights are assigned to roles, not individual users. Users are then assigned roles, and they inherit the access rights of their roles.

Implementing RBAC simplifies access management and improves security. It makes it easier to manage and track access rights, especially in large organizations with many users and resources. It also helps enforce the principle of least privilege, reducing the risk of unauthorized access.

d. Regularly Review and Update Access Rights

Access rights should be regularly reviewed and updated to match changes in users' roles, responsibilities, and employment status. This process should be automated as much as possible, using IAM tools that support access certification and recertification.

Regular access reviews help identify and correct excessive or inappropriate access rights. They also detect orphaned accounts, which belong to users who have left the organization but whose accounts have not been deactivated.

e. Transition to Passwordless Authentication

Transitioning to passwordless authentication can address the numerous vulnerabilities associated with password use, such as weak password creation, password reuse, and susceptibility to phishing attacks. Modern IAM systems support passwordless authentication methods, including biometrics, mobile device authentication, one time passwords (OTP), and SSO solutions.

Transitioning to these or similar authentication methods and stopping the use of passwords altogether can significantly improve security while also improving the user experience and reducing administrative overheads.

f. Educate and Train Employees

Finally, it's important to educate and train employees about IAM. They should understand the importance of IAM, the risks of poor IAM practices, and their role in maintaining good IAM.

Training should cover topics like secure password practices, phishing threats, and the importance of reporting suspicious activities. It should also explain the company's IAM policies and procedures, and the consequences of non-compliance.

Regular, ongoing training can help create a culture of security in the organization. It can also reduce the risk of insider threats, which are often caused by careless or uninformed employees.

User management allows administrators to:

- Group users according to their needs and roles
- Define flexible access policies
- Maintain the security of IT systems
- Prevent unauthorized access to infrastructure, applications, and data
- Store user details and credentials
- Provide a convenient login mechanism for end-users
- Allow users to set and reset passwords
- Allow users to create accounts
- Use social providers for authentication
- Federate with an identity provider
- Give users the freedom to choose from one or more providers
- Enable multi-factor authentication (MFA)
- Assign user rights to systems, services, and applications
- Manage user entitlements within services and applications

User management can improve security by:

- Implementing strong authentication methods, such as multi-factor authentication
- Carefully managing user access to resources and data

User management can reduce administrative costs by:

- Reducing costs related to customer password and registration issues by 75%
- Saving $30 per user password reset
- Reducing administrative costs to manage user access, requests, and policies by 80%

A solution commonly used to implement user management is IAM. IAM enables administrators to define access to IT resources, both for internal and external users. IAM either includes or integrates with a user directory service, which contains credentials and other details of all users. The directory service enforces access controls by authenticating, authorizing, and auditing user access.

Traditionally, organizations managed user management and authentication via on-premise identity providers (IdP) such as Microsoft Active Directory. The on-premise IdP server handled user management, authentication, and authorization for the local network. In recent years, IAM has moved to the cloud. Cloud-based IAM is more scalable and flexible, gives administrators more control, and is built for secure remote access.

2. THE NEED FOR MODERN USER MANAGEMENT

User management allows administrators to manage resources and organize users according to their needs and roles while maintaining the security of IT systems. Administrators need powerful

user management capabilities that can allow them to group users and define flexible access policies.

For end-users, many parts of user management are invisible. When users are exposed to user management—for example, when they use a login box to access an application—they expect the interaction to be simple and seamless. Login is a frequently-performed, critical operation, meaning that any delay or malfunction annoys users and hurts productivity.

Many organizations recognize that on-premise IdP solutions are insufficient for the modern IT environment. Users increasingly rely on cloud services and access corporate systems remotely, often via personal devices, and traditional IdP cannot address these use cases.

Organizations must find a way to manage secure access for a distributed environment. At the same time, users demand the same simplicity of popular services like Google and Facebook in their work environment. These challenges are making user management more important and more complex than ever before.

3. USER MANAGEMENT FUNCTIONS

User management encompasses a broad range of functions designed to enhance security, improve user experience, and streamline administrative processes. Here are some essential functions of user management:

- **User onboarding and offboarding:** efficient processes for adding new users and deactivating departing ones ensure that only current employees have access to critical assets
- **Role-based access control (RBAC):** assigns user roles (e.g., administrator, user, guest) and provides access to resources based on these roles. This minimizes the chance of unauthorized access
- **Profile management:** allows users to update their personal details, settings, and preferences, enhancing the user experience
- **Audit trails and monitoring:** keeps track of user activities, offering insights into who accessed what resources and when. This helps in detecting and preventing unauthorized activities
- **Automated workflows:** streamlines administrative tasks like approval processes for granting specific permissions
- **Integration with other systems:** modern user management tools can be integrated with other software, such as CRM systems, HR platforms, or cloud services, allowing for consistent data and reduced administrative overhead
- **Password policies:** enforces strong password requirements and periodic changes to enhance security

Notifications and alerts: informs users or administrators about potential security threats or suspicious activities.

4. EVOLUTION OF USER MANAGEMENT in 2023 AND BEYOND

Below are several trends driving the evolution of user management and the development of new technological solutions.

4.1. ZERO TRUST

The zero trust model is a security framework that enhances security in a modern, distributed IT environment. Zero trust calls for strict authentication for all connections, internal and external, aiming to eliminate implicit trust. Systems should only grant users the minimal privileges they need to perform their roles.

A core part of zero-trust security strategies is strict implementation of user access and identity verification. Strict access management means that if attackers compromise a regular user account, they cannot do anything beyond the account's authorized

privileges—preventing privilege escalation. Even if attackers compromise an admin account or device, zero trust access systems can identify anomalous use of the account and block malicious activity.

Zero trust requires continuous monitoring of user behavior, even after authentication, enabling rapid detection and response to compromised accounts.

IAM is a critical part of any zero trust implementation, because it provides a convenient mechanism for managing individual user access, concurrent connections, and access by third parties. IAM should closely integrate with network segmentation. This integration allows the organization to enforce network boundaries—known as "micro perimeters"—according to the organization's user access policies.

4.2. PASSWORDLESS

When it comes to user access control, security and user experience are often at odds. However, to be successful, user access must address both—it must be secure, and at the same time, must be convenient for users. Password-based authentication is a case in point—users find it more convenient to use short, simple passwords and reuse them across multiple services. However, this makes it very easy for attackers to compromise passwords. Organizations try to enforce strong passwords that cannot be easily guessed but are met with resistance from users.

Another challenge of password-based systems is password reset. Passwords are commonly lost or forgotten, requiring users to reset their passwords. In some cases, this is a lengthy and inefficient process, creating a nuisance for users. Password reset processes also open another door for password theft.

According to a Verizon report, 81% of breaches involving external hackers involved password compromise. Organizations are waking up to the threat posed by password-based authentication and are increasingly adopting passwordless authentication solutions. A passwordless authentication system identifies a user via multiple authentication methods without using a password the user must remember.

For example, the system might authenticate a user via biometric readings and one-time passwords (OTP). Passwordless systems provide more secure authentication which is also more convenient for users.

4.3. PRODUCT-LED GROWTH (PLG)

PLG is a new marketing approach in which the product itself drives customer acquisition, conversion, and retention. This aligns the entire organization, including product teams, sales and marketing, around improving the product and building it to promote and sell itself.

Product-led growth requires close consideration and optimization of every customer interaction with the product. Every interaction is an opportunity to create satisfied, loyal customers. User management can impact PLG in several ways:

- **User experience** – Ineffective user management can be a barrier to product-led growth. Reliance on inconvenient or insecure user management mechanisms hurts productivity and makes users unhappy. Conversely, an excellent sign-in process promotes customer satisfaction
- **First impression** – registration and login are the user's first point of interaction with a product. A smooth, seamless process creates trust and encourages registrants to become active users
- **Improved conversion** – a primary goal for product marketers is to get users to sign up and start using a product. A convenient registration and account creation process can dramatically improve conversion from a website visit or social referral to an active user

4.4. INCREASED COMPLEXITY

Traditionally, when organizations developed software, they would develop the authentication and user management components in-house. However, in today's technology ecosystem this is no longer feasible. Modern authentication and authorization systems are very complex, involving:

- The use of multiple identity providers, both internal and external
- The requirement for single sign on (SSO)
- Delegation of authentication and authorization to other applications, as in the case of social login
- Standardized methods for authorization including ACL, RBAC, and ABAC
- Multi-tenancy, as in the case of SaaS applications that must manage groups of users belonging to different organizations

To implement these complex requirements, and ensure user management is robust and secure, organizations must use third-party tools. This gave rise to an entire market of cloud services and applications that can help organizations implement user management and access control.

4.5. USER MANAGEMENT CONCEPTS

Here are a few key concepts that form the building blocks of a user management system.

4.5.1. User Profiles

A user profile is a collection of information related to a user and settings defined by that user. It contains sensitive information used to identify an individual, such as name, age, photo, and personal characteristics such as knowledge and expertise.

Most modern applications have user profiles, which allow users to manage their identity, control their preferences, and present themselves to others within the application. User profiles have achieved prominence as part of social networks—user profiles on services like Facebook, Twitter, and Instagram are becoming a critical part of users' online identity.

4.5.2. User Roles

A user role defines the functional role of a user. Users can then be granted permissions based on their role. There are many ways to define a role, including:

- **Organizational definition**—based on a specific employee's job or the team they are part of. For example, a user might have a "finance" role or an "engineering" role
- **Functional definition**—based on the specific functions a user needs to perform in a system. For example, an engineer who sometimes writes blogs can have the role of "author" on a company's website

Every system or application can define its own roles, and there are access control standards like RBAC and ABAC that provide a framework for allocating users to roles.

Here are a few common roles used by many systems:

- **Guest**—a user who is allowed to access the system, but does not have special privileges
- **Read-only access**—a user who needs to view information in the system but should not be allowed to modify it
- **Read/write access**—a user who needs to view information and also modify it (possibly with some restrictions)
- **Manager**—a user who has ownership over a project or other scope within the application. They will typically be able to read/write all data in the project and

set permissions for other users

- **Admin**—an administrator role typically grants a user the ability to modify configuration and assign permissions to other users, including managers (if applicable)
- **Super Admin**—a super admin, also known as "root user", has complete access to all system functions

4.5.3. User Permissions

User permissions are granted to users to allow access to specific resources, such as devices, files or folders, applications, and specific functions within an application. User permissions also specify the type of access for each object within a system.

Most systems can assign permissions to specific users, but this can be unwieldy for a large number of users. Therefore, it is common to attach user permissions to roles. This allows administrators to manage large numbers of users, allocating each user the appropriate permission based on their role. For example, in a CRM application, sales staff can have read, write, and delete permissions. Finance staff do not need to edit customer information, so they can receive read-only access.

4.5.4. User Groups

User groups are groups of users that perform similar tasks. Groups are a mechanism commonly used to manage permissions. Users can belong to multiple user groups and will receive the least restrictive permissions of all their groups.

For example, by defining a group with all the employees in a specific department, administrators can define permissions for the entire department in one place. Most applications and IT systems have a special administrator group that allows its members to modify system configuration and grant permission to other users. Many systems also use a default or "guest" group, which contains users that do not have any special permissions and do not belong to any other group.

4.5.5. Policies

User accounts, roles, permissions, and groups, are elements that can be used to construct access control policies. A policy is an organizational directive that determines how users should be allowed to access a certain system.

For example, the organization can enact a policy specifying that users should only be able to access the systems used within their department, and only within their working hours. To implement this policy, administrators can:

- Assign a role to each user specifying the department they belong to
- Define which systems belong to each department, and give each department role permissions to access the relevant systems, while denying permission to other systems
- Add a field to each user account specifying what are the employee's working hours
- Create a rule that denies access to users when they access a system outside their working hours

In reality, defining user access policies can be complex, especially in a large organization or in multi-tenant use cases. User management services can help organizations set up policies using a simple drag-and-drop interface, and easily modify these policies when business requirements change.

5. THREE GENERATIONS OF USER MANAGEMENT SOLUTIONS

To better understand the evolution of user management, let's break it down.

5.1. First Generation: On-premise Identity Provider (IdP)

An on-premise identity provider (IdPs) typically includes two components: a user management component, and a central directory service, such as Windows Active Directory or Apache Directory Services:

- **The user management component** delegates administrator privileges, tracks the roles and responsibilities of each user and group, configures user accounts, and manages passwords. Modern IdPs enable self service for some or all of these features, to reduce the burden on IT staff
- **The user directory** is a repository of user and group data, serving the entire organization. It gives administrators a unified view of users and permissions across all IT systems

5.2. Second Generation: Cloud-based Identity and Access Management

Cloud-based IAM, also known as Identity as a Service (IDaaS), is a cloud-based service hosted and managed by a third party provider. IDaaS provides all the capabilities of an on-premise IdP, but because it is cloud based, it is easier to set up, maintain and scale.

Businesses use cloud-based IAM to manage user identities and control access to corporate resources across cloud and on-premise systems. They ensure that the right individuals have access to the right resources, enabling controlled access from any device or location.

5.3. Third Generation: User Management Service

A user management service is an application that manages users from end to end. It is especially suited to SaaS applications or other use cases that involve users from multiple different organizations (known as multi-tenancy). User management services have additional functionality beyond the basic capabilities of traditional IdP or cloud-based IAM.

5.4. Key features of a user management service include:

- **End-to-end login process**—convenient, secure login interface, supporting multiple login options. These can include direct login, enterprise single sign on (SSO), and social login using existing accounts in services like Google, Facebook, and Twitter
- **User registration**—allowing users to conveniently self-register for a product or service, with post-registration workflows like transactional emails
- **User profile**—built-in UI to display user profiles within your application, and allow users to manage their profiles
- **Enhanced security**—support for multi-factor authentication (MFA), token-based authentication with strong encryption, and additional security controls such as account compromise detection
- **Multi-tenancy**—ability to support any organizational structure, from a single team, to multiple departments in a large enterprise, to thousands of customers using a product, each with their own team
- **Self-service account management**—ability for users to register for a new account on their own, customize their login settings, create user profiles, invite other team members and assign roles

6. TRANSFORMING USER MANAGEMENT

User management is an essential part of any modern SaaS application. The user journey includes the signup stage, login options, collaboration, security, and monetization, all are recurring components in all top applications today.

The experience of these users browsing through the application and working with it must be spotless. The self-served approach is what makes it happen. The approach is to transform our customers' applications into modern and frictionless ones. It has to cover everything from Signup to Checkout, allowing you to focus on the core differentiations and innovate.

7. ROLE BASED ACCESS CONTROL (RBAC)

- Role-Based Access Control (RBAC) assigns permissions based on user roles, not individuals
- Core RBAC components include roles, permissions, users, and role bindings
- RBAC improves security, simplifies audits, and supports least-privilege principles
- Implementations vary across platforms like Kubernetes, Azure, and GitHub

Role-Based Access Control (RBAC) is a method for restricting network access based on the roles of individual users. RBAC allows employees to access only the information they need to do their job. Employee roles in an organization determine the privileges granted to individuals and prevent lower-level employees from accessing sensitive information or performing higher-level tasks.

RBAC remains a widely adopted access control model across the tech industry. In fact, according to a 2025 study, 94.7% of companies have used RBAC at some point, and 86.6% say it's the model their platform uses today. This speaks to its reliability, scalability, and the familiarity engineers and security teams have with the model.

In the role-based access control data model, roles are based on several factors, including authorizations, responsibilities, and job competency. This model allows businesses to specify whether individuals are end-users, administrators, or expert users. Additionally, a user's access to computing resources may be restricted to certain operations, such as viewing, creating, or modifying files.

Network access restrictions are especially important for organizations with large numbers of employees and for organizations that allow access to third parties such as customers and suppliers, which can be difficult to monitor. Companies that rely on RBAC can better protect their sensitive data and applications.

7.1. How Does Role-Based Access Control Work?

RBAC assigns access permissions to users based on roles that define specific responsibilities and privileges.

Before implementing RBAC in an enterprise, the organization should define the permissions for each role as thoroughly as possible. This includes precisely defining permissions in the following areas:

- Permissions to modify data (e.g., read, write, full access)
- Permission to access company applications
- Permissions inside an application

The first step to getting the most out of RBAC is to model roles and permissions. This includes assigning all employee responsibilities to specific roles that determine appropriate privileges. The organization can then assign roles based on the employee's responsibilities.

Role-based access control allows organizations to assign one or more roles to each user or assign permissions individually. The goal is to define permissions that allow users to perform their tasks without further changes.
Not all teams rely solely on off-the-shelf RBAC. A 2025 survey found that 62.2% of organizations have built a custom in-house solution to manage authorization. This is done to meet specific business or security needs that basic role structures can't address.

Organizations use Identity and Access Management (IAM) systems to implement and monitor RBAC. IAM primarily supports businesses with large numbers of employees by logging, monitoring, and updating all identities and permissions. Assigning permission is

called "provisioning" and removing permission is called "deprovisioning". This type of system requires organizations to establish a unified and standardized set of roles.

7.2. What Is an RBAC Role?

An RBAC role is a predefined set of permissions assigned to users based on their job functions.

Within the RBAC framework, roles are semantics that organizations can use to build their privileges. Roles can be defined by various criteria, such as authority, responsibility, cost center, business unit, and more.

A role is a collection of user privileges. Roles are different from traditional groups, which are collections of users. In the context of RBAC, permissions are not directly associated with identities but rather with roles. Roles are more reliable than groups because they are organized around access management. In a typical organization, features and activities change less frequently than identities.

7.3. What is the RBAC Model

The RBAC model organizes permissions by associating them with roles and assigning roles to users.

There are three types of access control in the RBAC standard: core, hierarchical, and restrictive.

1. **Core RBAC**

   The core model describes the key elements of a role-based access control system. Core RBAC can serve as a standalone access control method, but it is also the basis for the hierarchical and constraint models.

   All RBAC models must adhere to the following rules:

   - **Role assignment—**a subject can only exercise privileges when the subject is assigned a role
   - **Role authorization**—the system must authorize a subject's active role
   - **Permission authorization**—a subject can only apply permissions granted to the subject's active role

2. **Hierarchical RBAC**

   Hierarchical RBAC builds on the basic RBAC model and introduces a role hierarchy. A role hierarchy is a way to structure roles to reflect a complex organizational structure and enable sharing and inheritance of permissions between roles.

   A simple example of hierarchical RBAC is a series of roles, in which each role inherits the permissions of the previous one, and adds more permissions:

   - **Guest user** – limited permissions
   - **Regular user** – same permissions as guest user and more
   - **Power user** – same permissions as regular user and more
   - **Administrator** – same permissions as a power user and more

   This is useful because any permission added to the guest user, for example, will automatically be added to all roles.

   Hierarchical RBAC supports several types of hierarchies:

   - **Tree** – bottom-up hierarchy in which the elements at the bottom of the tree grant permissions to elements higher up. For example, at the bottom is a departmental role with general permissions, which grants permissions to multiple employees
   - **Inverted tree** – top-down hierarchy in which senior roles inherit some of their permissions to junior roles under them
   - **Lattice** – a combination of bottom-up and top-down, where every role can inherit permissions from nodes below it and above it

The image below illustrates an inverted tree hierarchy for RBAC in an engineering department, where senior roles inherit permissions from junior roles. At the top, the Director oversees Project Leads, who manage Production and Quality Engineers, with Engineers at the base supporting both functions.

3. **Constrained RBAC**

Constrained RBAC adds separation of duties to the core model. There are two types of separation of duties:

- **Static Separation of Duties (SSD)**—no single user can have mutually exclusive roles (as defined by the organization). SSD prevents, for example, one person from making purchases and approving those purchases
- **Dynamic Segregation of Duties (DSD)**—users can be members of conflicting roles. However, the same user cannot perform both roles in a single session. This constraint helps control internal security threats by, for example, enforcing a two-person rule that requires two different users to approve an action

7.4. What is RBAC in Modern IT Systems?

In modern IT systems, RBAC is used to manage user access efficiently and securely across applications and services.

RBAC is used to manage authorization in many modern IT systems, including those powering the transition to the cloud. Let's see a few examples.

7.4.1. Azure RBAC

Azure RBAC is a real-world implementation of RBAC that helps administrators manage access to Azure resources and define exactly what actions can be performed within them. Azure RBAC is an authorization system based on Azure Resource Manager (ARM).

There are three key elements to assigning a role in Azure.

- **Principal**—a user, group, service principal, or managed identity that requested a resource and was granted access to the resource
- **Role definition**—a set of permissions on various Azure resources, associated with the role assigned to a principal. It defines the actions that a principal with a certain role can perform on a resource
- **Scope**—defines the resources the role provides access to and the permission level for one or more roles

In Azure, it is possible to define scopes directly at the management group, subscription, resource group, or individual resource level. Scopes have a parent-child relationship, and the child resource inherits the permissions of the parent resource.

Azure includes several built-in roles based on industry best practices and Azure resource structures. Azure also provides compatibility for creating custom RBAC roles based on an organization's requirements.

After defining roles and mapping scopes to these roles, administrators can use role assignments to grant access to a role scope for one or more security principles. They can also de-assign role assignments if the organization needs to remove access. Scopes thus make it easier to manage permissions.

7.4.2. AWS RBAC with Amazon Cognito

The Amazon cloud provides RBAC through the Amazon Cognito service. Cognito enables authentication, authorization, and user management for mobile and web applications.

Cognito uses the concept of user pools, which are managed user directories, and identity pools, which allow users to gain access to other AWS services. Identity pools grant users temporary, limited-privilege access to AWS resources. These permissions are created

through Amazon IAM roles. You can use tokens to assign roles to users of your applications, and leverage rule-based mapping to assign roles to users.

7.4.3. RBAC in Kubernetes

The RBAC model in Kubernetes includes the following components:

- **Role—**specifies permissions at the namespace level
- **ClusterRole**—specifies permissions at the cluster level or for namespaces across the environment
- **Subject**—describes a user, group, or service account
- **RoleBinding—**lists subjects and their assigned roles to bind roles to entities at the namespace level
- **ClusterRoleBinding**—lists subjects and assigned roles at the cluster level for every namespace in the cluster

The rbac.authorization.k8s.io API group determines all role authorization decisions in a Kubernetes cluster. It is best to ensure that the RBAC roles in all clusters are enabled—this helps the organization secure the project and maintain security compliance.

Additional best practices for using Kubernetes RBAC include:

- **Implementing least privilege**—with RBAC enabled, it blocks all access by default. An administrator can define permissions at a granular level, granting each user the necessary permissions. The least privilege approach prevents redundant permissions from expanding the attack surface and increasing the security risk. For example, most employees should not have cluster-admin permissions
- **Keeping the RBAC policy up to date—**an effective RBAC strategy requires continuous maintenance. Admins need to keep adjusting the roles and rules, validating and testing the RBAC. It may be useful to adopt a phased approach to RBAC management, taking the time to understand the issues and successes. Most important is regular reevaluations of the operational state and analysis of the environment for gaps like old user accounts
- **Keeping the roles simple—**Kubernetes RBAC roles are reusable, so there is no need to tailor roles to individual users. Organizations should use the fewest roles possible and avoid continuously creating new roles to enable simpler management. The more roles, the more complex the system becomes and the higher the potential risk. The focus of RBAC is on the roles, not the user

7.5. RBAC Alternatives

There are several ways to manage access control, including access control lists and attribute-based access control.

7.5.1. RBAC vs. ACL

Access control lists (ACLs) are tables that list the permissions associated with computing resources. An ACL is an object-based access control mechanism defining who can access a specific object and what actions are permitted. The operating system allows access based on entries for every user in the system, which list the permitted actions (i.e., view, create, export, etc.).

RBAC is the better option for most organizations because it offers higher security and less administrative effort than ACLs. An ACL may be useful for controlling access to low-level data. However, RBAC is more effective for controlling access.

7.5.2. RBAC vs. ABAC

Attribute-based access control (ABAC) implements a set of policies that manage access permissions according to attributes (i.e., object, user, system, and environmental

information). It uses boolean logic to determine if a user can access an object, evaluating set-valued or atomic attributes and their relationships to allow or deny access.

While RBAC relies on a predefined set of roles, ABAC is granular, making it more complex to manage. An example of the difference between RBAC and ABAC is that the first system might grant GitHub access to all users with a management role, while the second might restrict access to software engineers.

7.6. Examples of Role-Based Access Control

7.6.1. Kubernetes RBAC

Kubernetes uses RBAC to manage access to its cluster resources. Roles in Kubernetes define what actions a user or service account can perform within a namespace or cluster. Kubernetes RBAC defines roles at both the cluster and namespace levels, allowing for fine-grained access control.

For example, a "Developer" role may have permissions to deploy applications and manage pods in a specific namespace but not have access to modify critical system resources. Cluster-wide roles, on the other hand, might allow access across all namespaces, like a "Cluster Admin" role.

Role	Permissions	Scope
Cluster Admin	Full access to all resources	Cluster-wide
Developer	Create, update, delete pods	Namespace
Viewer	Read-only access to resources	Namespace
Service Account	Limited access to specific APIs	Namespace

7.6.2. Database Management Systems (DBMS) RBAC

In a Database Management System (DBMS), RBAC ensures that users only have access to the data and operations necessary for their roles. A "DB Admin" might have full access to all databases, including the ability to create, modify, or delete databases and tables, while a "Data Analyst" may only have read access to specific datasets. This structure helps maintain data integrity and enforce security by restricting sensitive operations to authorized users.

Role	Permissions	Scope
DB Admin	Full access to databases and tables	All databases
Data Analyst	Read-only access to data	Specified tables
Data Entry	Insert and update records	Specified tables
Auditor	Read and audit access logs	All databases

7.6.3. GitHub RBAC

GitHub uses RBAC to manage repository access. Users are assigned roles such as "Owner", "Maintainer", or "Contributor", with each role providing different levels of access. For example, an "Owner" can manage repository settings and collaborators, while a "Contributor" can commit code and create pull requests but cannot modify repository settings. This model helps teams collaborate efficiently while maintaining control over sensitive repository functions.

Role	Permissions	Scope
Owner	Full access, including repository settings	Repository
Maintainer	Manage pull requests and branches	Repository
Contributor	Create branches, submit pull requests	Repository
Read-only User	View repository code and issues	Repository

RBAC allows organizations to specify whether users are administrators, expert users, or end-users. Additional options include:

- Software engineering users who receive access to development tools like cloud services or source control repositories
- Marketing users who receive access to marketing tools like web analytics, content management systems (CMS), or customer relationship management (CRM)
- Finance users who receive access to billing systems or accounting software

Each role can include a separate management layer and contributor layer. Different roles have varying privilege levels within a given application.

When an end-user's job changes, the organization must manually reassign the roles to other users—or alternatively, assign the roles to a role group and use role assignment policies to add or remove members of role groups.

When users join a role group, they have access to all the roles in that group. Removing a user from a group restricts that user's access. Another option is to temporarily assign users to multiple groups, granting them access to certain data or programs and removing them when they no longer need access.

7.7. What are Role-Based Access Control Security Best Practices ?

Best practices include least privilege, regular audits, role minimization, and clear separation of duties.

RBAC can help organizations enhance their security posture and ensure compliance with security standards and regulations. Implementing company-wide RBAC is often complex, so it is important to ensure successful adoption to keep stakeholders on board. When adopting role-based access control, organizations should organize the implementation process using the following phases:

7.7.1. Determining the organization's needs

The organization should comprehensively analyze its business needs before adopting RBAC. The analysis should determine which job functions support established business processes and tools. Auditing and other regulatory requirements should also help inform the RBAC policy. It may be useful to combine RBAC with an additional access control measure in some cases.

7.7.2. Identifying the Scope of Implementation

The organization should identify the scope of the RBAC needs and align the implementation plan with these needs. The scope should be relatively narrow, focusing on apps and systems that process or store sensitive data. A limited scope makes for an easier transition.

7.7.3. Defining the roles

The organization should define roles based on the conclusions of the needs analysis, considering how users can perform their tasks. It is important to avoid the pitfalls of role design, such as an inappropriate level or granularity, excessive exceptions, and overlapping roles.

7.7.4. Rolling out the RBAC system

The organization should implement RBAC in stages to avoid disrupting business with overwhelming workloads. For example, the first stage might address a core user group, providing less granular access control and gradually providing more detailed roles. User feedback and monitoring of the RBAC implementation in the computing environment can help plan the rest of the rollout.

7.8. RBAC for SaaS Applications

User management is a crucial component of modern SaaS apps due to the multiplication of use cases and customer requirements. You need robust and self-selved capabilities to create an app that can scale up fast without causing stress on IT and dev teams. ABC allows organizations to manage roles and permissions seamlessly without losing focus on core tech development.

Our end-to-end user management platform is completely self served and comes with predefined roles and permissions for quick implementation. We also allow the creation of new ones based on your requirements. That's not all. Your customer's end-users can also unlock a smooth in-app experience without putting additional stress on your IT teams. A truly self-served option.

8. ATTRIBUTE BASED ACCESS CONTROL (ABAC)

ABAC (Attribute-Based Access Control) is an evolution of the more traditional RBAC (Role-Based Access Control) methodology. It allows the use of additional attributes for a more granular approach. The option of using user, environment, and resource attributes is now allowing SaaS businesses to address more complex use cases.

8.1. Attribute-Based Access Control (ABAC)

Attribute-based access control (ABAC) is an authorization paradigm that defines access control policies according to attributes like resource, object, environment, and user attributes. ABAC uses Boolean logic to create access rules containing if-then statements, which define the user, the request, the resource, and the action. For example, if the requester is an accountant, then allow read-write access to financial data.

ABAC enables organizations to create dynamic, context-aware access control policies using specific attributes according to unique business needs and compliance requirements. Implementation of ABAC was announced as a Priority Objective for implementation by the US Federal Government, and the National Institute of Standards and Technology (NIST) issued a set of standard guidelines that define how to implement ABAC in an enterprise environment.

8.2. How Does ABAC Security Work? Key Components

ABAC systems make access decisions by:

1. Intelligently studying how attributes interact in an environment
2. Creating rules that determine access for a set of attributes if specific conditions are met

Here are the four main categories of ABAC security attributes:

Subject/user attributes	Attributes that indicate an individual is attempting to gain access. For example, username, age, ID, job title, organization, job role, department, and security clearance
Resources/object attributes	Attributes that indicate the resource requested for access.
Action	Attributes that indicate the action the user wants to perform with a resource. For example, view, transfer, read, or delete.
Environmental attributes	Attributes that contextualize the access attempt. For example, time, device, and location.

ABAC systems establish policies that define what combination of various attributes is required to perform a specific action with a certain object or resource. The system uses these policies to grant or deny access.

Here is how the process typically works:

1. An access request is triggered
2. The ABAC tool scans attributes to determine if they match existing policies
3. If the attributes match a policy, the system grants access to the user

8.3. RBAC vs ABAC

RBAC and ABAC are access management methods. RBAC grants access to roles, whereas ABAC uses attributes-based policies to grant or deny access.

8.3.1. What is RBAC?

RBAC was formalized by the NIST in 1992 and quickly became the standard for SMBs with over 500 employees. It was implemented in user provisioning systems to streamline the joiners, movers, and leavers (JML) human resources (HR) process. It enables organizations to manage access control by roles instead of the individual user ID or each employee. It involves grouping users and entitlements according to business functions or activities, called roles.

Organizations can now use RBAC to create flat or hierarchical roles and also include inheritance. The key benefit is using roles as an extra level of abstraction. Roles act as a set of entitlements or permissions. It significantly simplifies access management, enabling organizations to assign one role to hundreds of users, a big advantage while scaling up fast.

8.3.2. How ABAC differs from RBAC

ABAC enables organizations to extend existing roles via attributes and policies. It offers the context needed to make intelligent authorization decisions. Instead of granting access only by roles, ABAC systems account for the role, the relevant actions and resources for the job, the location, time, and how the request is made.

ABAC policies are based on individual attributes, consist of natural language, and include context. It eliminates the need for hundreds of overloaded roles, enabling administrators to add, remove, and reorganize attributes without rewriting the policy. It requires significantly fewer roles. As a result, it offers simpler identity management.

8.4. ABAC: Key Use Cases and Examples

8.4.1. Healthcare

In healthcare, ABAC can enforce strict data access policies based on various attributes such as job role, location, and time of access. For example:

- A doctor may have access to a patient's medical records only during their shift and only if they are in the hospital premises
- Nurses might have read-only access to patient records relevant to their current duties
- Sensitive information like psychiatric records could be restricted to only specific authorized personnel, ensuring compliance with HIPAA and other regulatory requirements

8.4.2. Financial Services

ABAC is particularly useful in the financial sector where regulatory compliance and security are paramount. Financial institutions can use ABAC to control access to sensitive data such as customer financial information, transaction histories, and internal reports. For example:

- A financial analyst might be allowed to view and analyze transaction data during business hours but restricted from accessing the same data remotely
- Managers in a finance department might be allowed to view all transaction data in the company office
- Only designated, authorized individuals are allowed to edit the company's annual financial reports, while in the office and accessing them from specific, secure devices

8.4.3. Government and Defense

Government and defense sectors require highly granular access controls to protect classified information. ABAC allows these organizations to specify access policies based on security clearance levels, project assignments, and operational needs.

For example, a military officer may access classified documents only if they possess the necessary clearance, are currently assigned to a relevant project, and are accessing the information from a secure location.

8.4.4. Education

Educational institutions can leverage ABAC to manage access to various digital resources such as student records, research data, and administrative systems. For example, a professor may access student grades and academic records for the courses they teach, but not for other departments. Students can be granted access to course materials and their own records, but restricted from accessing other students' information. This approach helps in maintaining privacy and data integrity across the institution.

8.5. Benefits of ABAC

ABAC offers several key advantages over traditional access control models:

- **Granular control**: ABAC provides fine-grained access control by evaluating multiple attributes, enabling organizations to define highly specific access policies. This ensures that only authorized users can access particular resources under precise conditions
- **Flexibility and scalability**: ABAC policies can adapt to a wide range of scenarios without requiring constant adjustments. As attributes can be dynamically assigned and updated, ABAC scales efficiently with organizational changes, new user roles, and evolving security requirements
- **Context-awareness**: By considering environmental attributes such as time, location, and device, ABAC enhances security through context-aware decision-making. This helps in mitigating risks associated with unauthorized access attempts from unusual locations or during odd hours

- **Improved compliance**: ABAC facilitates compliance with regulatory requirements by allowing organizations to implement stringent access controls and audit trails. Policies can be crafted to ensure that access to sensitive information is restricted according to legal and organizational mandates
- **Reduced role explosion**: Unlike role-based access control (RBAC), which can suffer from "role explosion" due to the need to create numerous roles for various scenarios, ABAC reduces the complexity by leveraging attributes. This simplification leads to fewer roles and more manageable policies

8.6. ABAC Implementation Challenges

While ABAC offers significant benefits, its implementation can present several challenges:

- **Complex policy management**: Developing and maintaining ABAC policies can be complex due to the multitude of attributes and conditions that need to be considered. Organizations must carefully design policies to avoid conflicts and ensure that access decisions are both accurate and efficient
- **Attribute management**: Effective ABAC requires robust management of user, resource, action, and environmental attributes. Ensuring the accuracy and integrity of these attributes is critical, as incorrect or outdated attributes can lead to inappropriate access grants or denials
- **Performance overheads**: Evaluating multiple attributes for each access request can introduce performance overheads, particularly in environments with high volumes of access attempts. Organizations must optimize their ABAC implementations to balance security and performance
- **Interoperability and integration**: Integrating ABAC with existing systems and applications can be challenging. Organizations may face compatibility issues or need to customize their infrastructure to support ABAC policies, which can require significant time and resources
- **Training and awareness**: Implementing ABAC requires a shift in how access control policies are created and managed. Organizations must invest in training for administrators and users to ensure a smooth transition and effective use of the ABAC system

8.7. ABAC in the Cloud

ABAC is one of the access models offered by leading cloud providers. Here is how the two leading providers – Amazon Web Services (AWS) and Microsoft Azure have implemented ABAC.

8.7.1. AWS ABAC

Amazon Web Services (AWS) is a cloud computing vendor that offers ABAC as part of its identity and access management (IAM) service. Here is how AWS ABAC works:

- Use attributes as tags and attach them to IAM resources and IAM entities like roles and users
- Create one or a set of ABAC policies for IAM principals
- Configure AWS ABAC policies to allow operations when a principal's tag matches a resource tag

ABAC is ideal for rapidly-growing environments and complex policy management scenarios. AWS ABAC enables organizations to control access to AWS resources, helping teams and resources grow with little changes to AWS policies. It lets you pass session tags when assuming a role or federating a user and then define policies that use tag condition keys to grant permissions to principals.

Organizations using a SAML-based identity provider (IdP) can use SAML attributes to set up fine-grained access control within the AWS cloud. SAML attributes include user email

addresses, cost center identifiers, project assignments, and department classifications. Passing these attributes as session tags enables you to use them to control access to AWS.

8.7.2. Azure ABAC

Microsoft Azure is a cloud computing vendor that offers various access control management options, including RBAC and ABAC. Azure RBAC is an authorization system for managing access to Azure resources via role definitions and role assignments. Azure ABAC builds on Azure RBAC, adding attributes-based role assignment conditions in context to specific actions.

A role assignment condition serves as an additional check you can add to a role assignment for granular access control. A condition can filter permissions granted for a role definition and a role assignment. It lets you add a condition that requires a certain object to have a specific tag in order to read this object. However, it does not let you use conditions to explicitly deny access to specific resources.

Azure ABAC enables you to significantly reduce the number of role assignments. Currently, Azure subscriptions have a role assignment limit. In some cases, it can require thousands of role assignments that you also need to manage. You can add conditions to reduce the number of role assignments.

8.8. Template for Deploying an ABAC Solution

Here is a template of key factors to consider before deploying an ABAC solution, based on the NIST guidelines.

You can use this template to fill in your organization's unique considerations for implementing an ABAC solution.

8.8.1. Establish the Business Case for ABAC Implementation

- Define costs of developing or acquiring new capabilities and transitioning away from legacy technology
- List the benefits provided by ABAC
- Map any new risks introduced by ABAC, if any
- Understand the governance structures required to manage shared capabilities and policy documentation
- List all datasets, applications, networks, and systems that require ABAC capabilities

8.8.2. Understand the Operational Requirements and Overall Enterprise Architecture

- Define the management, monitoring, and validation processes of privileges for compliance
- Determine interfaces and objects to expose for information sharing
- Choose the relevant ACM
- Define sharing and management processes and tools for subject and object attributes
- Specify access control rules, including how to capture, evaluate, and enforce these controls

8.8.3. Establish or Refine Business Processes to Support ABAC

- Determine whether access rules and policies are fully understood and documented

- Specify how to identify and assign attributes
- Define how to handle access failures
- Specify the parties allowed to create new policies
- Define how to share and manage common policies

8.8.4. Develop and Acquire an Interoperable Set of Capabilities

- Determine how to achieve interoperability
- Define how to integrate subject attributes from identity management integrated into ABAC
- Determine how to handle unique identities
- Define how to share and manage subject attributes across enterprise entities
- Assess centralization tradeoffs in comparison to distributed authentication, attribute management, authorization, and enforcement
- Specify how to use environment conditions in access decisions
- Define how to measure confidence, quality, and accuracy for access decisions
- Specify how to map attributes between organizations

8.8.5. Evaluate Performance

- Define how to manage subject attributes for disconnected connections, limited bandwidth, or resource-limited users
- Specify how to measure and enforce the quality and timeliness of changes to attributes
- Determine measures that indicate adequate overall system and end-to-end performance

8.9. ABAC for SaaS Applications

SaaS app usage is becoming more and more unpredictable, with multiplying usage patterns and use cases. Businesses need a versatile solution that's self-served and plug-and-play in nature, enabling engineering teams to focus more on core technology development and customers to enjoy more in-app independence with less friction points.

ABC SaaS-as-a-Service infrastructure takes user management to the next level by providing users with maximum flexibility when it comes to implementation and adoption for complex use cases and requirements.

9. AUTHORIZATION

Authorization is the process of allowing an entity to have access to something or perform specific actions in a computer system.

In a multi-user system, the system administrator uses the authorization mechanism to define permissions for each user or group of users. Once a user is logged in, via a process called authentication, the system determines which resources should be available to them during their session. So the term authorization can have two related meanings:

- Allocation of privileges to users by the system administrator
- Granting access to users at runtime according to predetermined permissions

9.1. Authentication vs Authorization

Authentication is the process of verifying a user's identity—making sure they are who they say they are. Traditionally, the most common method for identity authentication was a password. If the username provided by a user matched the password credentials, the identity was determined valid and the user was granted access.

Authorization comes into action after the user's identity has been verified through authentication. It provides full or partial access to resources such as devices, files, applications, specific operations or data. To take a simple example:

- A user logs into a business application, providing their company username and password
- The application authenticates the user and verifies the password
- The application checks what permissions are allocated to that username and grants access to the relevant data and features

9.2. How Does Authorization Work?

Authorization works by determining a user's permissions and access levels after their identity has been authenticated. The process involves several key steps:

1. **Verification of identity**: Once authentication confirms a user's identity, the system consults its authorization policies to decide what resources and actions the user is allowed to access
2. **Access policies:** Authorization policies are defined based on the organization's security requirements and are typically managed through frameworks like Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), or Relationship-Based Access Control (ReBAC). These policies set the permissions for various user roles, attributes, or relationships
3. **Permission assignment:** The system assigns permissions according to the defined policies. For example, in RBAC, users are assigned roles like 'admin' or 'editor,' which come with predefined access rights
4. **Access control mechanisms:** When a user attempts to access a resource, the access control system checks the user's permissions against the requested action. If the user's permissions include the requested action, access is granted; otherwise, it is denied
5. **Audit and monitoring:** Continuous monitoring and logging of authorization actions help in detecting unauthorized access attempts and ensuring compliance with security policies

9.3. User Authorization Strategies and Techniques

9.3.1. Role-Based Access Control (RBAC)

RBAC mechanisms allow or restrict access to users according to their roles within the organization. It provides users with access to data and applications required to perform their jobs while minimizing the risk of unauthorized activities like access to sensitive data.

Organizations can use RBAC to define how each user can interact with data. For example, it lets you allow read/write or read-only access to specific roles to limit a user's ability to perform unauthorized actions, such as data deletion and commands execution.

9.3.2. Attribute-Based Access Control (ABAC)

ABAC lets you define access and privileges based on attributes (or characteristics) instead of roles. It helps protect objects such as network devices, IT resources, and data from unauthorized actions and users that do not have the approved characteristics defined by the organization's security policies.

ABAC evolved from simple access control lists to a form of logical access control. It was initially endorsed in 2011 by the Federal Chief Information Officers Council to help federal organizations improve access control architectures. The council recommended ABAC to help organizations safely share information.

9.3.3. Relationship-Based Access Control (ReBAC)

ReBAC applies access decisions according to the relationships between subjects such as users, applications, and devices. When a subject requests access to a resource, the ReBAC mechanism grants or denies access according to the relationships the subject has.

ReBAC does not allow access based on a certain role (like RBAC) or attributes (like ABAC). Instead, it allows access based on a subject's relationships with other entities. These relationships influence how the authorization policy grants or denies access to protected resources.

Social networks use ReBAC to allow access. Facebook, for example, allows friends of friends to access photos and posts. It means friends can see the user's posts and also the friends of these friends. However, this is where it stops—third connections are not allowed access.

9.3.4. JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard that defines how to securely transmit information between parties like JSON objects. It verifies information by digitally signing it. JWTs are compact and self-contained. You can sign JWTs using a public/private key pair through RSA or ECDSA, or a secret through the HMAC algorithm.

You can also encrypt JWTs to increase security. Signed tokens can verify claims' integrity, and encrypted tokens hide claims from other parties. When you use public/private key pairs to sign tokens, the signature certifies that only the party that holds the private key signed it.

You can use JWT for authorization mechanisms. After the user logs in, each subsequent request includes the JWT to allow the user to access the services, resources, and routes permitted with this token. Since JWT can easily be used for different domains and has a small overhead, it also powers many Single Sign-On (SSO) mechanisms.

9.4. Authorization Use Cases and Examples

9.4.1. Enterprise Data Access

In large organizations, data access needs vary significantly based on job functions and responsibilities. Role-Based Access Control (RBAC) is a common strategy to manage this complexity. RBAC assigns permissions to roles rather than individuals, simplifying administration and enhancing security.

Consider a scenario where the finance department needs to generate quarterly financial reports. The analysts have access to financial databases, accounting software, and reporting tools, all governed by their role permissions. However, these analysts cannot access HR records or confidential executive communications, as their role does not include these permissions. This segmentation reduces the risk of unauthorized data access and ensures compliance with data protection regulations.

9.4.2. Cloud Resource Management

Cloud providers like AWS, Azure, and Google Cloud implement Attribute-Based Access Control (ABAC) to handle the dynamic and complex access requirements of modern enterprises. ABAC policies consider attributes such as user roles, department, project involvement, and security clearance levels. This allows for more granular control compared to RBAC.

For example, a company using AWS can set policies where only members of the DevOps group have access to deploy new applications, while developers in a

specific project team can only access the development and testing environments specific to their project. Additionally, ABAC can enforce time-based access controls, allowing access during work hours and restricting it outside of these times.

9.4.3. Healthcare Systems

In healthcare, protecting patient data is of utmost importance due to legal and ethical considerations. RBAC is commonly used to ensure that only authorized medical staff can access patient records, adhering to principles like the Health Insurance Portability and Accountability Act (HIPAA) in the U.S.

For example, a nurse might have access to basic patient information such as contact details and medication schedules, essential for daily care. In contrast, doctors and specialists have comprehensive access to a patient's medical history, diagnosis, and treatment plans. This hierarchy ensures that sensitive information is only accessible to those with a legitimate need, thus protecting patient confidentiality and maintaining compliance with regulatory standards.

9.4.4. eCommerce Platforms

eCommerce platforms face the challenge of securing user sessions and authorizing actions seamlessly across various services. JSON Web Token (JWT) is a popular solution for this purpose. JWT tokens are generated upon user authentication and included in subsequent requests to authorize actions like adding items to the cart, placing orders, or accessing order history.

Consider an online retail site where a user logs in to make a purchase. After authentication, a JWT is created and sent to the user's browser. This token is included in each request to the server, ensuring that the user remains authenticated without re-entering credentials. The JWT contains encoded information about the user's session, ensuring secure and efficient authorization across the platform.

JWTs can also be easily integrated with Single Sign-On (SSO) systems, allowing users to access multiple related services with a single login, enhancing user experience and security.

9.4.5. Social Media Networks

Social networks like Facebook utilize ReBAC to manage access to user-generated content based on relationships. Users control who can view their posts, photos, and other content by defining their relationships with other users (e.g., friends, friends of friends).

For instance, a Facebook user can set a photo album's privacy settings to 'Friends,' ensuring only direct friends can view the photos. They can also extend access to 'Friends of Friends,' allowing a broader audience while still maintaining a level of control. This approach helps users protect their privacy and manage their social interactions effectively. ReBAC's flexibility allows social networks to offer nuanced privacy controls that cater to various user preferences and social dynamics.

9.5. Common Authorization Challenges

9.5.1. Complexity in Policy Management

As organizations grow and their IT infrastructure becomes more sophisticated, managing access control policies can become overwhelmingly complex. This complexity arises from the need to balance varying access requirements across different departments, roles, and projects. For example, different teams within an

organization may require access to different sets of resources, and each team may have different levels of access permissions.

In addition, regulatory requirements often necessitate fine-grained access controls, adding another layer of intricacy to policy management. Ensuring that all these policies are correctly implemented and consistently enforced is a significant challenge, especially when using frameworks like Attribute-Based Access Control (ABAC), where multiple attributes and conditions must be managed.

9.5.2. Scalability

Authorization systems must be capable of scaling efficiently as the organization grows. This means handling an increasing number of users, resources, and access requests without compromising performance or security.

Scalability issues can arise in various forms, such as delays in access decision-making processes, increased load on servers, and difficulties in maintaining up-to-date access control lists. In large enterprises and cloud environments, the dynamic and often unpredictable nature of scaling requirements further complicates the implementation of effective authorization mechanisms. This challenge is particularly pronounced in environments where thousands of access decisions must be made in real-time.

9.5.3. Integration with Existing Systems

Integrating new authorization mechanisms with legacy systems presents a formidable challenge. Legacy systems were often not designed with modern access control requirements in mind, leading to potential compatibility issues. These older systems might use outdated or incompatible protocols, making it difficult to implement contemporary authorization solutions without significant modifications.

In addition, the integration process must ensure that the existing functionality remains uninterrupted, which often requires extensive testing and potentially custom development. This process can be time-consuming and resource-intensive, posing a significant barrier to updating authorization infrastructures.

9.5.4. User Experience

Balancing robust security measures with a seamless user experience is a complex task. Overly stringent authorization controls can frustrate users, leading to reduced productivity and potential workarounds that compromise security.

For instance, if users frequently encounter access denials or cumbersome multi-factor authentication processes, they may seek less secure methods to achieve their goals. Striking the right balance involves not only implementing effective security controls but also designing intuitive and user-friendly interfaces and processes that do not hinder the user's ability to perform their tasks efficiently.

9.6. Authorization Best Practices

9.6.1. Design and Implement Authorization Early in the Software Development Lifecycle

Authorization functionality is a critical security component that should be considered during the early phases of the software development lifecycle. Building your application with security in mind helps ensure the design itself is secure, reducing complex and costly fixes when the product is already ready for release. When designing authorization, consider all requirements for roles and privileges and use the most suitable web framework supporting this type of authorization.

9.6.2. Implement Authorization Middleware

Authorization middleware enables you to configure short-lived authorization grants and set up the server to perform continuous authorization on each request. It means the authorization middleware handles the requests, eliminating the risks associated with API endpoints and unprotected URL paths.

9.6.3. Implement Authorization Policies

Authorization policies enable you to configure and enforce fine-grained access controls. You can use various types of policies, such as mandatory access control (MAC), RBAC, ABAC, or discretionary access control (DAC).

ABAC is preferable over RBAC, according to the OWASP foundation. However, you should choose the method that suits the application's requirements, and in some scenarios, ABAC may not be the optimal choice. Ideally, the authorization type should include at least one compulsory deny-by-default policy to block unauthorized access.

9.6.4. Implement Scope Bounds

A scope bound defines a blast radius for functions and features. It reduces the impact of threats that manage to bypass security controls. Limiting functionality and features to a specific scope can help avoid various vulnerabilities, such as local file inclusion (LFI), remote file inclusion (RFI), and directory traversal. You can implement scope bounds by allowlisting specific local directories, program binaries, and URL endpoints.

9.6.5. Validate and Sanitize Dynamic Values

Validation processes include data verification, and sanitization processes attempt to clean malicious inputs. Proper sanitization and validation of all values that compute authorization, including user inputs, can help prevent the exploitation of many vulnerabilities. Here are key best practices:

- Validation and sanitization must include user inputs and all session data required to compute authorization
- The sanitization and validation of inputs must be performed on the server side

Using input validation and sanitization with scope bounds can help you avoid security risks associated with parameter tampering and vulnerabilities like path traversal, RFI, and LFI.

9.6.6. Implement Authentication and SSO in Internal Applications

Applications should always include access controls. You can use standards like Oauth2, SAML, or OpenID Connect to implement single sign-on (SSO). SSO is an authentication scheme that enables you to defer authentication and authorization policy management to an external identity provider. SSO is considered a standard in application security, but you can use other authentication methods.

10. SINGLE USER ID

A single user ID is a unique name, number, or string that identifies one specific person across software systems, websites, or apps. It helps track user activity, manage account access, and link data safely.

10.1. Key Characteristics

- **Uniqueness**: No two users share the same ID
- **Permanence**: It usually stays the same over time

- **Privacy**: It links data without exposing secret personal details

11. REGISTRATION

User registration is the process where a person creates an account on a website or app by providing details like a username, email address, and password.

11.1. Core Steps

- **Sign Up Form**: Users fill out fields with personal or login details
- **Verification**: The system checks the data and often sends a confirmation email link
- **Account Access**: Users log in with their credentials to use special site features

12. LOGIN

A login is the process of gaining access to a computer system, website, or app by providing identification, such as a username and password. It serves as a digital key, ensuring only authorized users can access personal accounts, sensitive information, or private networks.

-

12.1. Key Login Concepts

Understanding the login process involves a few specific terms and mechanisms:

- **Credentials**
  These are the unique pieces of information you provide to prove your identity-most commonly a username (or email) and a password
- **Authentication**
  This is the underlying security process the system uses to verify that your credentials match their records
- **Two-Factor Authentication (2FA)**
  Many modern systems add an extra layer of security. After entering your password, you might be prompted to enter a code sent to your phone or email to ensure it is actually you

12.2. Common Variations

- **Log in (Action)**
  Written as two words, this is a verb that describes the act of accessing a system
- **Login (Noun/Adjective)**
  Written as one word, it refers to your credentials or the screen/form where you enter them (e.g., "Go to the login page to log in.")
- **Social Login**
  A feature that allows you to sign into a new website by using an existing account from another service, such as Google, Apple, or Facebook

13. FORGOT LOGIN CREDENTIAL

"Missing credentials" means the system or application you are trying to access cannot find the required login information—such as your username, password, or an authentication token. This usually happens if you haven't entered them, the system failed to transmit them, or your saved login files are corrupted.

You can break down and resolve this issue depending on how and where it is happening:

1. **Logging into a Website or App**
   - **The Cause**
     You left the username or password fields blank, or the login page failed to send your details
   - **The Fix**
     Re-enter your username and password carefully. If you are using a saved password in your browser, try typing it manually instead of using autofill
2. **Digital Portfolios or Profiles**
   - **The Cause**
     If you are referring to a professional or educational profile (like Quora), missing credentials mean you haven't filled out or have removed that specific background or expertise section. If it is an app where you expected a digital certificate or badge to show up, it may have been issued to a different email address
   - **The Fix**
     Update your profile section to include your credentials, or contact the issuer to ensure your digital badge was sent to the correct email address
3. **API, Coding, or Software Development**
   - **The Cause**
     Software and APIs will throw this error when an authentication token, API key, or authorization header is left out of the request
   - **The Fix**
     Verify your API keys and secrets. Ensure that your code's authorization headers are formatted correctly and that the application has permissions to access the targeted server

13.1. FORGOT LOGIN CREDENTIALS

To recover forgotten login credentials, use the **Forgot Password** or **Forgot Username** link on the login page, verify your identity via a registered recovery email, phone OTP, or security questions, and check browser or password manager vaults

13.1.1. Standard Recovery Procedures

- **Email Recovery Link**
  Enter your username to receive a temporary reset link sent to your saved email address
- **Phone / SMS OTP**
  Confirm your identity using a text message code sent to your trusted mobile number
- **Security Questions**
  Answer personal verification questions set up during account creation
- **Saved Credentials Check**
  Inspect your device's browser settings or password manager for stored auto-fill entries