EDUSERV IT · IAM PLAYBOOK

IAM UNLOCKED

Authentication Playbook

Authentication in IAM — 8 Sections for Beginners & Professionals

WHAT'S INSIDE — AT A GLANCE

This cheat sheet covers 8 essential sections on Authentication in IAM — from first principles to real systems, common mistakes, and interview readiness.

01	**Definition**	What authentication is — and why it matters
02	**Why It Exists**	The real security problem it solves
03	**Real-World Flow**	Step-by-step: how it works in a live system
04	**Key Components**	Every building block involved in authentication
05	**Real-World Scenario**	A banking app login, explained behind the scenes
06	**Common Mistakes**	High-value lessons engineers get wrong
07	**Architecture View**	Where authentication sits in your system
08	**Interview Power Points**	Key facts expected in IAM interviews

- **AuthN = proving WHO you are**
- **MFA = 2+ factors combined**
- **Token = your digital pass after login**
- **IdP = the system that verifies you**

01 DEFINITION

"What is Authentication?" — The foundation of every security decision

DEFINITION

Authentication is verifying that someone is who they claim to be before granting system access.

KEY CONCEPTS

- **Authentication** — Verifying identity before any access is granted — WHO are you, REALLY?
- **Identity Claim** — What the user asserts ("I am Alice") — unverified until authentication confirms it.
- **AuthN vs AuthZ** — AuthN = proving identity. AuthZ = deciding permissions. AuthN always comes FIRST.
- **Verification** — The process of checking the claim using evidence the user provides.
- **Trust** — After successful AuthN, the system trusts the identity for the session duration.

REAL-WORLD EXAMPLE

When you type your username, you are making a CLAIM. The system does not believe you yet. Authentication is the process of the system VERIFYING that claim using evidence you provide — password, OTP, or fingerprint. Only after verification does any other security decision happen.

WHY IT MATTERS IN INDUSTRY

— Authentication is the #1 line of defence — before firewalls, before encryption.

— Stolen or weak credentials are the leading cause of data breaches worldwide.

— Modern systems use centralised IdPs (Okta, Azure AD) so auth logic is not per-app.

— Machine identities (APIs, services, CI/CD) also require authentication — not just humans.

02 WHY AUTHENTICATION EXISTS

The real security problem it solves

DEFINITION

Without authentication, any system is wide open — anyone can claim any identity and access everything.

KEY CONCEPTS

- **The Problem** — Digital systems cannot physically see who is at the keyboard. Anyone can type any username.
- **The Risk** — No authentication = no security boundary. Any user could access admin data, billing, PII.
- **The Solution** — Force every entity to PROVE their identity before any access is granted.
- **Scope** — Applies to humans, machines, apps, APIs, microservices — every entity needs it.
- **Foundation** — All other IAM pillars (authorization, audit, SSO) require authentication to work first.

REAL-WORLD EXAMPLE

Imagine a hospital with no door locks or ID checks. Any person who walks in can access patient records, medication dispensers, or surgery schedules. Authentication is the digital equivalent of checking ID at every door — before anyone can do anything.

WHY IT MATTERS IN INDUSTRY

— 80%+ of data breaches involve compromised credentials — authentication failure is the root cause.

— Cloud platforms (AWS, Azure, GCP) enforce authentication on every API call — no exceptions.

— Zero Trust security model: "Never trust, always verify" — authentication on every single request.

— Regulations (HIPAA, PCI-DSS, SOC2) explicitly mandate authentication controls for sensitive systems.

03 REAL-WORLD AUTHENTICATION FLOW

Step-by-step — How it actually happens in a live system

DEFINITION

Authentication follows a structured flow: user provides evidence → IdP verifies → token issued → access granted.

KEY CONCEPTS

- **Step 1 — User Action** — User opens app/portal and enters username + password on the login form.
- **Step 2 — Forwarded** — App sends credentials securely (HTTPS) to the Identity Provider (IdP).
- **Step 3 — IdP Verifies** — IdP checks hashed password against directory (Active Directory / LDAP / DB).
- **Step 4 — MFA Challenge** — IdP sends OTP to phone or requests biometric. User proves the second factor.
- **Step 5 — Token Issued** — Signed JWT Access Token + Refresh Token issued. User is now authenticated.

REAL-WORLD FLOW

1	**User Initiates Login** *User*	Opens app or portal. Enters username + password on the login form.
2	**Credentials Sent to IdP** *App → IdP*	App forwards credentials securely over HTTPS to the Identity Provider auth endpoint.
3	**IdP Verifies Identity** *IdP / LDAP*	IdP hashes the password and compares to stored hash in directory. Plaintext is never transmitted.
4	**MFA Challenge Issued** *IdP → User*	IdP sends a 6-digit OTP to phone (or biometric prompt). User approves — second factor confirmed.
5	**JWT Token Issued** *IdP → App*	Signed JWT Access Token + Refresh Token returned. User session begins. Token is the proof.

KEY INSIGHT	The application **never stores your password**. It delegates verification to the IdP. The signed JWT token returned IS the proof of authentication — apps verify this token on every subsequent request.

04 KEY COMPONENTS INVOLVED

The building blocks of every authentication system

DEFINITION

Authentication involves multiple components working together — not just a username and password field.

KEY CONCEPTS

- **User / Principal** — The entity requesting access — human, service, bot, or device.
- **Application / Client** — The system the user wants to access — web app, API, or mobile app.
- **Identity Provider (IdP)** — The authority that stores and verifies identities (Okta, Azure AD, Ping Identity).
- **Factor — Know** — Something you know: password, PIN, or security question.
- **Factor — Have** — Something you have: phone OTP, hardware key (YubiKey), or smart card.
- **Factor — Are** — Something you are: fingerprint, face scan, or iris scan (biometrics).
- **Token / Session** — Temporary digital pass issued after successful auth — JWT or session cookie.
- **Directory / DB** — Backend identity store the IdP queries — Active Directory, LDAP, or user DB.

REAL-WORLD EXAMPLE

You sit at a bank's website. You (User) open the banking app (Application). You enter credentials — the app calls Okta (IdP). Okta checks your password against its directory, then texts an OTP (MFA). On success, Okta issues a JWT (token). The bank's API validates that token on every request you make.

WHY IT MATTERS IN INDUSTRY

— IdPs like Okta and Azure AD centralise auth for hundreds of apps — one source of truth.

— Tokens (JWTs) are stateless — APIs can validate them without calling the IdP every time.

— Biometric factors are the most secure — they cannot be phished or guessed.

— Machine identities (service accounts, API keys) follow the same component model.

05 REAL-WORLD SCENARIO

Logging into a Banking App — Behind the scenes, step by step

SCENARIO

1	**Open the banking app** *Browser / App*	Login page loads. No identity check yet — you are anonymous at this stage.
2	**Enter username + password** *User Action*	You type credentials. The app does NOT verify them locally — it forwards them to the IdP.
3	**App calls the IdP auth endpoint** *App → IdP*	POST /oauth/token — credentials sent securely over HTTPS to the Identity Provider.
4	**IdP hashes and compares** *IdP / Directory*	Server hashes your password with bcrypt and compares to stored hash. Plaintext is never stored.
5	**MFA challenge issued** *IdP → User*	Bank texts a 6-digit OTP (something you HAVE). You enter it. Two factors are now confirmed.
6	**JWT Access Token issued** *IdP → App*	{"user":"you@bank.com","role":"customer","exp":3600} — signed with IdP private key.
7	**App validates token each call** *App / API Gateway*	Every API request sends the token in the Authorization header. API verifies signature + expiry.
8	**Token expires → Refresh** *Refresh Token flow*	After 60 min, access token expires. Refresh token silently fetches a new one — or re-login.

06 COMMON MISTAKES

High-value lessons — what beginners and engineers get wrong

×	**Confusing AuthN with AuthZ**	Authentication = WHO you are. Authorization = WHAT you can do. They are always separate steps.
×	**Password alone is secure**	Passwords are stolen constantly. MFA eliminates 99.9% of account compromise attacks.
×	**Ignoring token expiry**	Long-lived tokens are a massive breach risk. Keep access tokens short — minutes to hours.
×	**Not validating the token**	Apps MUST verify token signature, expiry, and issuer on every single request — not just once.
×	**Storing passwords in plaintext**	Always hash with bcrypt or Argon2. Never store or log raw passwords under any circumstance.
×	**Skipping machine authentication**	APIs, CI/CD pipelines, and microservices all need authentication too — service accounts and certs.

07 ARCHITECTURE VIEW

Where authentication sits in your system

◆ **Entry Point**	Auth sits BEFORE any API or resource access. Nothing passes without it.
◆ **IdP — Trust Anchor**	The Identity Provider is the centralized authority. Apps delegate all auth to it.
◆ **Protocol Layer**	App ↔ IdP via OAuth 2.0 / SAML / OIDC — standard, open protocols.
◆ **Token Checkpoint**	Token returned → app checks it on EVERY request. No server-side session needed.
◆ **Directory Backend**	LDAP / Active Directory is the identity store the IdP queries to validate credentials.
◆ **Audit Trail**	Every auth event is logged: who, when, success/fail — essential for compliance.

08 INTERVIEW POWER POINTS

Key facts — expect these in IAM interviews

Q1	**Authentication (AuthN) = verifying identity. Authorization (AuthZ) = verifying permissions. Never confuse them.**
Q2	**Three authentication factors: Something you KNOW (password), HAVE (OTP / hardware key), ARE (biometrics).**
Q3	**MFA combining 2+ factors eliminates 99.9% of account compromise attacks per Microsoft research.**
Q4	**JWT = JSON Web Token. Format: Header.Payload.Signature — self-contained and signed by the IdP.**
Q5	**OAuth 2.0 is an authorization framework. OIDC (OpenID Connect) adds the authentication layer on top.**
Q6	**Access tokens: short-lived (minutes to hours). Refresh tokens: long-lived, used to silently renew access.**
Q7	**Passwordless auth (passkeys, magic links) removes phishing risk entirely — the future of enterprise auth.**
Q8	**SAML 2.0 = enterprise SSO using XML. OIDC = modern SSO using JSON/JWT — preferred for mobile and APIs.**

MEMORY HOOK

Authentication = The Security Gate Check before entering the building.
Show your ID (username) → Guard confirms your face (password/biometric) → Stamp on your hand (token issued) → You're in.