# Securing Your REST API: Authentication and Authorisation Done Right

**Author:** Kamakshaiah Musunuru | **Published:** 11 September 2026 | **Category:** Tech Update | **Read time:** 1 min read

**Abstract**

Authentication and authorisation remain the most error-prone parts of API design. This article provides a comprehensive analysis of proven schemes, token lifecycle management, and the principle of least privilege for production REST APIs.

**Keywords:** REST API, authentication, authorization, OAuth 2.0, OpenID Connect, JWT, least privilege, token

**Tags:** REST API, authentication, authorization, security, OAuth

Secure APIs rely on a clear separation between authentication (who you are) and authorisation (what you can do). This article examines the proven patterns for getting both right in production systems.

Core Concepts

1. Authentication Schemes Choose a battle-tested scheme rather than building your own: - OAuth 2.0 and OpenID Connect for delegated access - API keys for server-to-server machine clients - Session cookies for browser-based applications - JWT for stateless distributed systems when used with care

2. Token Lifecycle Management Tokens are only as safe as their handling: - Issue short-lived access tokens and rotate refresh tokens - Store secrets and tokens outside client-side source code - Validate issuer, audience, expiry, and signature on every request - Revoke sessions promptly on logout, compromise, or role change

3. Authorisation and Least Privilege Identity is not permission. Enforce granular, server-side access control: - Define scopes and roles that map to specific capabilities - Deny by default and allow only what is required - Validate resource ownership on every object access - Audit authorisation decisions for anomalies

4. Practical Hardening - Rate limit endpoints and return 429 with Retry-After on spikes - Validate and schema-check all request payloads - Force TLS everywhere and disable insecure cipher suites - Log auth failures and alert on brute-force patterns

Recommended flow for a typical service: user authenticates via OIDC, receives a short-lived access token, the API validates it and enforces scopes per route, and refresh tokens are rotated on renewal and stored securely.

Authentication and authorisation failures are a leading cause of data breaches. Teams should treat them as first-class engineering concerns, covered by automated tests, penetration testing, and regular review.

## *References*

OAuth 2.0 Authorization Framework (RFC 6749); OpenID Connect Core 1.0; OWASP Authentication Cheat Sheet; OWASP API Security Top 10.