What is broken authentication in APIs?

Broken authentication is the group of API flaws where the part that checks who is calling fails. It covers how login tokens are made, checked, stored, expired, and recovered. It sits at number two on the OWASP API Top 10, just below BOLA. When the 'who are you' check breaks, every 'are you allowed' check that depends on it breaks too, which is why one broken-authentication flaw often lets an attacker take over many accounts at once.

HOW IT WORKS

01 What broken-authentication patterns show up in production?

From the API engagements we ran in the last year:

- No or weak rate limit on login. Lets an attacker run a stolen password list against the live API at full speed.
- Username give-aways on login. A different error for 'wrong password' versus 'no such user' lets an attacker confirm real usernames before brute-forcing.
- JWT misconfiguration. Tokens that accept the wrong algorithm, carry forgeable claims, or are signed with a weak shared secret.
- Password-reset poisoning. The reset link goes to an address the attacker controls, because the app trusted an attacker-supplied host header.
- OAuth and SSO flaws. Open redirects in the OAuth flow, missing PKCE, weak state checks, or accepting a token meant for a different audience.
- Leaked API keys. Keys committed to public repos, baked into mobile apps, or returned in error messages.
- Logout that does not log out. A logout endpoint that does not actually kill the token. Common with JWTs, because their stateless design makes them harder to revoke than session tokens.

02 What do attackers do when authentication breaks?

Most broken-authentication chains end in account takeover. The forms:

HOW TO DEFEND

- Use a well-maintained auth library or provider. Hand-rolled authentication is where most of these bugs come from.
- Rate-limit login, signup, and recovery hard. Limit per IP, per account, and per credential, with growing delays and a lockout after too many tries.
- Require MFA on powerful accounts. At least for admins, ideally for anyone who can reach sensitive data.
- Lock down how tokens are made and checked. Pin the algorithm. Use short-lived access tokens with refresh tokens. Check every claim. See JWT Security.
- Test the recovery flows on purpose. Password reset and account recovery hide the highest-impact bugs.
- Watch for credential stuffing. Flag login bursts from new IPs, odd user-agent patterns, and post-login behavior that does not match the account's history.

SOURCES

[1] OWASP API2:2023 Broken Authentication

[2] OWASP Authentication Cheat Sheet

[3] NIST SP 800-63B Digital Identity Guidelines

[4] CWE-287 Improper Authentication

- Targeted takeover. Take over one known account (a customer, an admin, an internal user) using a weakness specific to it.
- Credential stuffing. Run a stolen password list against the live API until accounts that reuse passwords fall.
- Mass takeover. Find one flaw that takes over many accounts at once (a token you can forge for anyone, or a reset that accepts any user ID).
- Privilege escalation. Use the weakness to grab credentials for powerful accounts: admins, service accounts, or internal service-to-service tokens.

03 How does SecureLayer7 test broken authentication?

Every API engagement runs the broken-authentication matrix.

- Login flow. Rate limit, username give-aways, MFA bypass, a credential-stuffing run, and timing leaks.
- Tokens. The JWT attack set (alg=none, algorithm confusion, weak secrets), session-token randomness, and OAuth flow checks.
- Password reset and recovery. Host-header poisoning, reusing a reset link, takeover through parallel reset requests, and OTP brute-forcing.
- Logout and revocation. We check that logout really kills the token, and that a password change ends other sessions.
- Recovery beyond passwords. Email change, phone change, security-question reset, and social-login linking attacks.

The report maps findings to OWASP API2:2023, with the exact code or config change for each.

Test your API authentication end to end.
securelayer7.net/learn/api-security/broken-authentication

Open online