# JWT Security: Common Vulnerabilities, Real-World Cases, and Defenses

## Overview

JSON Web Tokens (JWTs) are widely used for stateless authentication. Their security depends on co

## Key Vulnerabilities

1) Weak or Guessable Secrets (HS256)
   - Risk: Attackers brute-force HMAC secrets and forge tokens (e.g., role=admin).
   - Defense: Use long, random secrets (>= 256 bits) or asymmetric RS256; rotate keys regularly.

2) 'alg=none' Acceptance
   - Risk: If servers accept 'alg=none', unsigned tokens bypass signature verification.
   - Defense: Explicitly allowlist algorithms; never accept 'none'.

3) Algorithm Confusion / Key Confusion
   - Risk: Treating an RSA public key as an HMAC secret or accepting algorithm swaps can allow signa
   - Defense: Pin expected algorithm per application; separate HS/RS code paths; validate key types.

4) Replay Attacks
   - Risk: Stolen but otherwise valid tokens are replayed until expiry.
   - Defense: Short expiries (e.g., 15 min), refresh tokens, TLS everywhere, jti-based blacklists, device-

5) Claim Validation Gaps
   - Risk: Missing checks for exp/nbf/iat, iss/aud, or accepting unsigned critical claims.
   - Defense: Verify exp/nbf/iat with clock skew, enforce iss/aud, and sign all sensitive claims.

6) Insecure Storage/Transport
   - Risk: Tokens in localStorage are vulnerable to XSS; tokens over HTTP are sniffable.
   - Defense: Use HttpOnly, Secure cookies; strict Content Security Policy; HTTPS-only; avoid exposin

## Real-World References

- OWASP JWT Cheat Sheet – configuration pitfalls and best practices.
- NIST NVD (CVE) – examples of libraries and services impacted by algorithm confusion and verificati

## Proof-of-Concept Summary

- Included script 'jwt_attack_demo.py' shows:
  a) Brute-forcing a weak HS256 secret from a small wordlist, then forging an admin token.
  b) Constructing an 'alg=none' token (PoC only).

## Defensive Checklist

- Use RS256 or strong HS256 secrets (>= 32 bytes random).
- Allowlist algorithms; reject 'none' and disallow algorithm switching.
- Validate exp/nbf/iat/iss/aud; enforce short lifetimes and rotation.
- Protect tokens in transport (TLS) and storage (HttpOnly cookies).
- Implement logout / jti revocation.
- Monitor and log failed verification attempts.

Citations
- OWASP: JSON Web Token (JWT) Cheat Sheet
- NIST NVD: https://nvd.nist.gov/