JWT Security Research Report

Prepared by: Dayasri
Project: JWT Security – Research, PoC Attack & Secure API
Date: August 2025

Submitted as part of Security Research Project

Research Report: JWT Vulnerabilities

Introduction

JSON Web Tokens (JWT) are widely used for securing web applications, enabling stateless authentication and authorization. However, improper implementation can lead to severe vulnerabilities. This report highlights common JWT vulnerabilities, their implications, and real-world references from OWASP and NIST CVEs.

JWT Structure

[Header] . [Payload] . [Signature]

Common JWT Vulnerabilities

alg=none Vulnerability

Some JWT libraries incorrectly allow tokens with alg=none, effectively bypassing signature verification. Attackers can craft unsigned tokens to impersonate legitimate users.

Mitigation: Always enforce strong algorithms (HS256, RS256).

Weak or Guessable Secrets

JWTs signed with weak secrets (e.g., 123456, password) are vulnerable to brute-force attacks. Attackers can forge valid tokens if the secret is predictable.

Mitigation: Use long, random, high-entropy keys.

Key Confusion Attacks

Occurs when the server mistakenly accepts a public key as an HMAC secret. An attacker may provide a forged token signed with their private key, but verified using the server's public key.

Mitigation: Enforce strict key usage (HS256 for symmetric, RS256 for asymmetric).

Replay Attacks

A stolen JWT can be reused until expiration, allowing unauthorized access.

Mitigation: Implement short token lifetimes, refresh tokens, and token revocation strategies.

Lack of Expiry Validation

JWTs without proper exp (expiry) claims may remain valid indefinitely.

Mitigation: Always enforce token expiration and validate it on the server.

Real-World Vulnerabilities

CVE-2015-9235: Node-jsonwebtoken package vulnerable to alg=none attack.

CVE-2018-0114: Auth0 JWT library susceptible to signature verification bypass.

CVE-2020-26160: Go-JWT library with weak default settings allowing forgery.

These cases demonstrate the impact of insecure JWT implementations in real-world systems.

Conclusion

JWTs are powerful but dangerous if misconfigured. Developers must enforce secure algorithms, use strong secrets, validate token expiry, and adopt defense-in-depth strategies such as refresh tokens and revocation lists. Following OWASP recommendations and monitoring NIST CVEs is essential for secure implementations.

References

OWASP JWT Cheat Sheet: https://cheatsheetseries.owasp.org

NIST CVE Database: https://cve.mitre.org

Auth0 Security Blog on JWT: https://auth0.com/blog/a-look-at-the-latest-draft-for-jwt-bcp