JWT Vulnerabilities - Short Research Report

Overview:

JSON Web Tokens (JWTs) are commonly used for stateless authentication.
If implemented incorrectly, JWTs expose multiple vulnerabilities.

Common vulnerabilities:

1) alg=none / Signature stripping - Some verifiers accept 'alg=none' and mark tokens valid without verificat
2) Weak secrets - Short or guessable HMAC secrets let attackers forge valid tokens by signing payloads.
3) Key confusion - Allowing algorithm switching (e.g., RS256 <-> HS256) may let attackers use public keys
4) Replay attacks - Long-lived tokens can be reused; lack of nonce/exp checks enables replay.
5) Claim tampering and missing validation - Not validating 'exp', 'nbf', 'iss', 'aud' allows unauthorized acces

Mitigations:

- Always validate the algorithm and restrict to specific algorithms (e.g., HS256 or RS256), never accept 'no
- Use strong, high-entropy secrets for HMAC (>= 256 bits recommended) and protect private keys.
- Prefer asymmetric signatures (RS256/ES256) for third-party token issuance; carefully handle key usage.
- Validate token claims ('exp', 'nbf', 'iat', 'aud', 'iss') and implement short token lifetimes with refresh tokens
- Use token revocation lists or sliding windows and implement nonce/UUID to prevent replay when needed
- Use libraries that perform safe checks (constant-time compare, disallow dynamic alg selection).

References:

- RFC 7519 (JSON Web Token) - IETF.
- OWASP JSON Web Token Cheat Sheet - OWASP Foundation.

Conclusion:

JWTs are convenient but dangerous if validators are naive. Follow the mitigations above
and use vetted libraries and secure key management.