JWT authentication bypass via flawed signature verification

Executive Summary

This report details the discovery of a critical vulnerability in the target web application's authentication mechanism, specifically involving JSON Web Tokens (JWT). Due to insecure server-side configuration, the application accepts unsigned JWTs by setting the algorithm to `"none"`. Exploiting this flaw allowed full administrative access, including deletion of sensitive user accounts. The risk level of this vulnerability is **high**.

Introduction

The goal of this penetration test was to evaluate the robustness of the authentication and session management mechanisms employed by the web application. The focus was placed on inspecting JWT implementation and configuration for common security missteps.

Methodology

1. Logged in as a regular user and captured session traffic using Burp Suite.

2. Analyzed the JWT token structure and claims.

3. Modified token claims and headers to simulate privilege escalation.

4. Removed JWT signature after setting `alg` to `none` to test server verification behavior

5. Confirmed administrative access and conducted an authorized action (user deletion) to verify impact.

Vulnerability Findings

- **Type**: JWT Authentication Bypass via Flawed Signature Verification

- **Location**: Session token in `Cookie` header

- **Severity**: High

Description

The web application used JSON Web Tokens (JWTs) to maintain user sessions. The JWT included a `sub` claim indicating the logged-in username. However, the server accepts tokens with the `alg` header set to `none` and does not verify the token's signature. This allows attackers to craft their own JWTs and impersonate arbitrary users, including the administrator.

Proof of Concept (PoC)

1. Logged in as user.
2. Captured the JWT from the `GET /my-account` request:

   css

   CopyEdit

   `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ3aWVuZXIifQ.[signature]`
3. Decoded and modified payload:

   json

   CopyEdit

   `{  "sub": "administrator" }`
4. Modified header:

   json

   CopyEdit

   `{  "alg": "none" }`
5. Removed the signature, resulting in a JWT of the format:

   php-template

   CopyEdit

   `<header>.<payload>.`
6. Sent a request to `GET /admin` with this tampered JWT as the session cookie.
7. Successfully accessed the admin panel and visited:

   pgsql

   CopyEdit

   `GET /admin/delete?username=user`

   to delete the target user.

Impact Assessment

This vulnerability enabled:

- Complete account takeover, including privileged users.

- Unauthorized access to restricted admin functionalities.

- Potential loss of data integrity and user trust.

If exploited in a real-world environment, attackers could impersonate any user, access or manipulate sensitive resources, and cause severe business disruptions.

Recommendations

1. Reject unsigned JWTs – configure the application to disallow `alg: none`.

2. Enforce strong signature verification – use only secure algorithms like `HS256` or `RS256` with secret or public key validation.

3. Use libraries that do not support insecure configurations by default.

4. Rotate JWT secrets regularly and invalidate old tokens when detected.

5. Conduct code reviews and implement security testing pipelines to identify such flaws early in development.

Conclusion

The application's acceptance of unsigned JWTs critically undermines its authentication model. Immediate remediation is essential to prevent unauthorized access and ensure trust in session integrity. Adopting secure JWT handling practices is crucial for the long-term security of the application.