RESEARCH REPORT ON JWT HACKING

What is JWT Hacking?

JWT, or JSON Web Token, is a compact and self-contained way to securely transmit information between parties as a JSON object. It's commonly used for authentication and information exchange in web applications.

Key Components of a JWT

A JWT consists of three parts:

1. Header: Typically contains two parts— the type of the token (JWT) and the signing algorithm (e.g., HMAC SHA256).

2. Payload: Contains claims, which are statements about an entity (usually the user) and additional data. These can be standard claims (like `sub` for subject, `iat` for issued at, and `exp` for expiration) or custom claims.

3. Signature: Created by encoding the header and payload, then signing them with a secret key using the specified algorithm. This ensures the token's integrity and authenticity.

How It Works

1. User Authentication: A user logs in and the server creates a JWT containing the user's information.

2. Token Issuance: The JWT is sent back to the client.

3. Subsequent Requests: The client includes the JWT in the HTTP headers of requests to access protected resources.

4. Verification: The server verifies the token's signature and extracts the user information from the payload.

Benefits

1. **Compact:** Easy to transmit in URLs, HTTP headers, or POST parameters.
2. **Self-contained:** Contains all the necessary information, reducing the need for database lookups.
3. **Cross-platform:** Works with various programming languages and frameworks.

Security Considerations

- Use HTTPS to protect tokens during transmission.

- Set expiration times to limit token lifespan.

- Use strong signing algorithms and keep your secret keys secure.

In summary, JWT is a widely used method for managing authentication and securely transmitting information in web applications.

What is the structure of JWT token?

The structure of a JSON Web Token (JWT) consists of three main parts, each encoded in Base64Url format and separated by dots (`.`). Here's a breakdown of its components:

1. Header

The header typically consists of two parts:

- `alg`: The signing algorithm used (e.g., `HS256` for HMAC SHA-256).

- `typ`: The type of token, which is usually "JWT".

Example:

```json
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2. Payload

The payload contains claims. Claims are statements about the user or entity and additional data. There are three types of claims:

- **Registered claims:** Standard claims like `sub` (subject), `iat` (issued at), and `exp` (expiration).

- **Public claims:** Custom claims that can be defined by users to share information.

- **Private claims:** Claims created to share information between parties that agree on them.

Example:

```json
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}
```

3. Signature

The signature is created by taking the encoded header, the encoded payload, a secret key, and the algorithm specified in the header. This ensures that the token hasn't been altered.

Example:

- To create the signature, you would:

  1. Encode the header and payload in Base64Url.

  2. Concatenate them with a period (`.`): `header.payload`.

3. Apply the signing algorithm with your secret key.

Final JWT Structure

A complete JWT looks like this:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIi
wibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6y
JV_adQssw5c
```

Summary

Header: Contains the token type and signing algorithm.

Payload: Contains claims about the user or entity.

Signature: Verifies the integrity of the token.

Each part is Base64Url encoded, and they are concatenated with periods to form the final JWT.

What is JWT token hacking and what are the types of attacks?

JWT token hacking refers to various methods and techniques used by attackers to exploit vulnerabilities in systems that use JSON Web Tokens (JWTs) for authentication and authorization. Here are some common types of attacks related to JWTs:

1. Token Forgery

Description: Attackers create a fake JWT by manipulating the token's header and payload.

How It Works: If the signing algorithm is not properly validated (e.g., if the server accepts `none` as a signing algorithm), attackers can create a token without a valid signature.

2. Algorithm Confusion

- **Description:** Attackers exploit the server's acceptance of multiple signing algorithms.

- **How It Works:** If a server is configured to accept multiple algorithms, an attacker might modify the token’s header to change the algorithm to `none`, thus bypassing signature verification.

3. Replay Attacks

- **Description:** Attackers capture a valid JWT and reuse it to gain unauthorized access.

- **How It Works**: Since JWTs often do not expire immediately or are not validated against a session store; an attacker can use a captured token to impersonate a user.

4. Brute Force Attacks

- **Description:** Attackers attempt to guess the secret key used to sign the JWT.

- **How It Works:** If a weak or easily guessable key is used, attackers can generate valid tokens by brute forcing the secret.

5. Token Theft

- **Description:** Attackers steal JWTs from users or servers.

- **How It Works:** This can occur through various methods, such as cross-site scripting (XSS) attacks or unsecured transmission (not using HTTPS).

6. Session Fixation

- **Description:** Attackers set a user's session ID (or token) to a known value.

- **How It Works:** When the user logs in, the attacker can use the same session to access the user’s account.

7. Insufficient Expiration

- **Description:** Tokens that do not expire or have a long expiration time can be exploited.

- **How It Works:** An attacker can use a stolen token for an extended period if it is not set to expire soon.

8. Poorly Implemented Claims

- **Description:** Using claims incorrectly can lead to vulnerabilities.

- How It Works: For example, if sensitive information is stored in the payload without encryption, it can be accessed by unauthorized users.

By understanding these attack vectors and implementing security best practices, you can significantly reduce the risk of JWT-related vulnerabilities.

Impact and Mitigation of JWT token Hijacking

JWT token hijacking is a serious security threat where an attacker steals a valid JSON Web Token (JWT) and uses it to impersonate a legitimate user. The impact and mitigation strategies for this type of attack are as follows:

Impact of JWT Token Hijacking

1. Unauthorized Access:

- Attackers can gain access to sensitive resources and perform actions on behalf of the victim, potentially leading to data breaches or unauthorized modifications.

2. Data Theft:

- Sensitive user information, such as personal data, financial information, or confidential business data, can be exposed.

3. Reputation Damage:

- Organizations may suffer reputational harm if user data is compromised, leading to loss of trust.

4. Financial Loss:

- Financial repercussions may arise from data breaches, regulatory fines, or loss of customers.

5. Account Compromise:

- Attackers can take over user accounts, leading to further exploitation and potential malicious activity.

Mitigation Strategies

1. Use HTTPS:

- Always transmit JWTs over secure connections (HTTPS) to prevent interception during transmission.

2. Short-lived Tokens:

- Set a short expiration time (`exp` claim) for tokens to limit the window of opportunity for attackers.

3. Implement Refresh Tokens:

- Use refresh tokens to allow users to obtain new access tokens without requiring them to log in again, minimizing the lifespan of access tokens.

4. Token Revocation:

- Implement a mechanism to revoke tokens if suspicious activity is detected, or when users log out.

5. Strong Signing Algorithms:

- Use secure signing algorithms (e.g., RS256) and strong secret keys to prevent token forgery and brute force attacks.

6. Store Tokens Securely:

- Avoid storing JWTs in local storage or session storage in browsers. Instead, consider using Http Only cookies to prevent access via JavaScript.

7. Validate Claims:

- Validate all claims in the JWT, including issuer (`iss`), audience (`aud`), and expiration (`exp`) to ensure tokens are being used as intended.

8. Monitoring and Logging:

- Monitor for unusual login patterns or token usage. Implement logging to track token creation and usage, which can help identify suspicious activities.

9. User Education:

- Educate users about the importance of secure token storage and the risks of sharing tokens or credentials.

By implementing these mitigation strategies, organizations can significantly reduce the risk of JWT token hijacking and enhance their overall security posture.

REFERENCE:

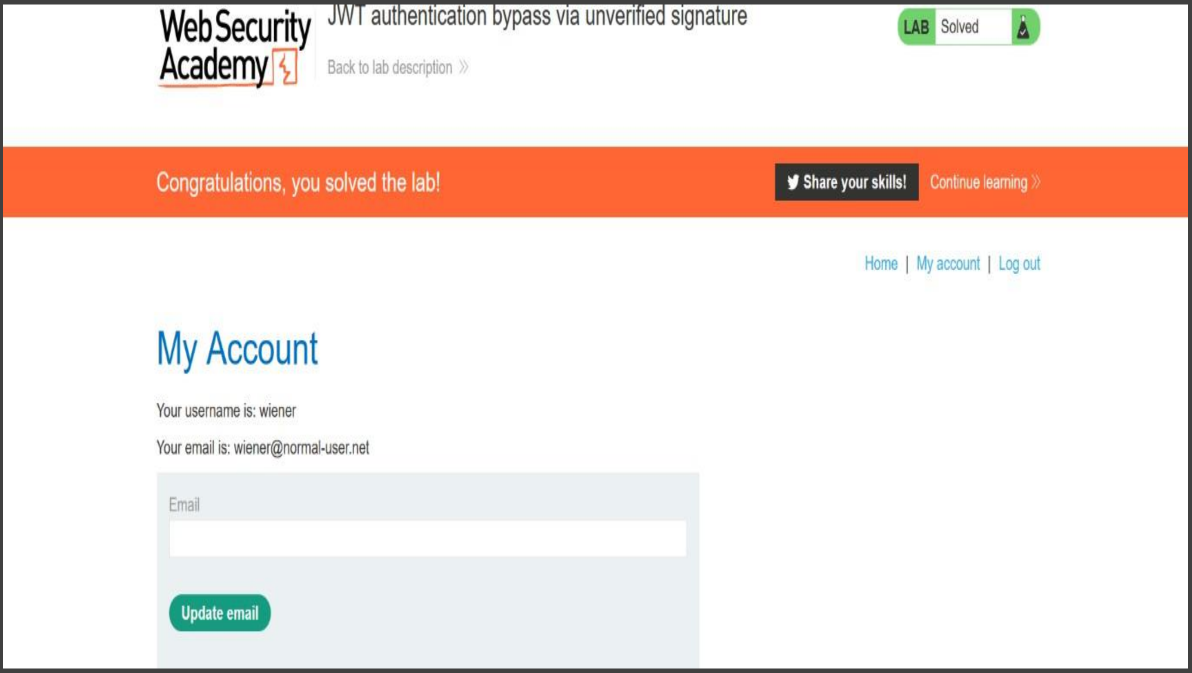
https://jwt.io/introduction

https://medium.com/@rajeevranjancom/jwt-json-web-token-attacks-6b82185ffed

https://www.invicti.com/blog/web-security/json-web-token-jwt-attacks-vulnerabilities/

https://portswigger.net/web-security/jwt/lab-jwt-authentication-bypass-via-unverified-signature

PRACTICAL


JWT authentication bypass via unverified signature

Congratulations, you solved the lab!

My Account

Your username is: wiener

Your email is: wiener@normal-user.net