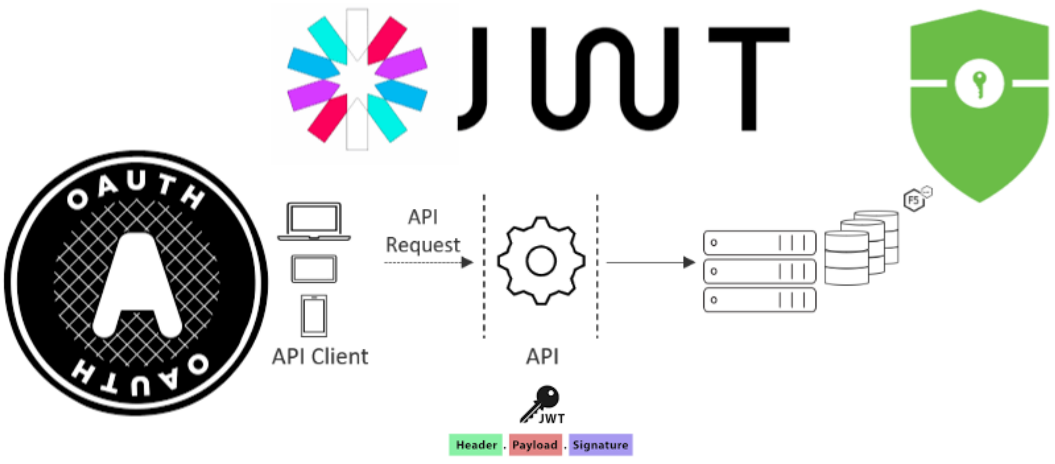
JWT Token Hacking

Karthik B

<u>What is JWT Token?</u>

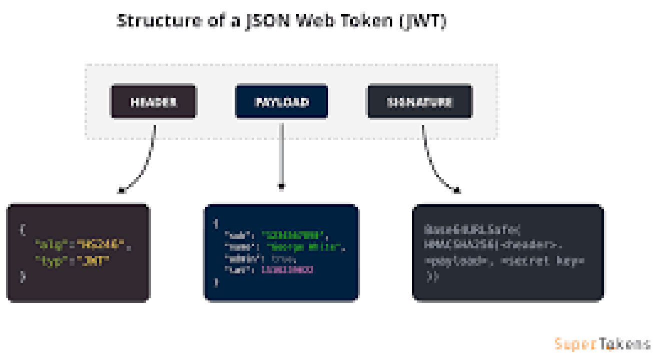
A JSON Web Token (JWT) is a widely accepted standard (RFC 7519) that outlines a concise and self-contained method for securely transmitting data between entities in the form of a JSON object. The integrity and trustworthiness of this information are ensured through digital signatures. JWTs can be signed using either a secret (utilizing the HMAC algorithm) or a public/private key pair through RSA or ECDSA.



Below are some scenarios demonstrating the application of JSON Web Tokens:

- **Authorization:** In the realm of user authentication, JWTs play a pivotal role. Once a user logs in, subsequent requests carry the JWT, granting access to specific routes, services, and resources authorized by the token. JWTs are particularly favored in Single Sign-On implementations due to their minimal overhead and seamless applicability across diverse domains.
- **Information Exchange:** JSON Web Tokens provide a secure means of exchanging information between different entities. By leveraging signatures, such as those generated by public/private key pairs, the authenticity of the senders can be verified. Moreover, since the signature is computed using both the header and payload, it becomes possible to ensure that the content remains untampered during transmission.

What is the structure of JWT Token?



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

```
{
  "sub": "1234567890",
  "name": "George [illegible]",
  "admin": true,
  "iat": 1516239022
}
```

```
Base64URLSafe(
HMACSHA256(<header>.
<payload>, <secret key>
))
```

The JSON Web Token (JWT) structure comprises three essential components:

1. **Header:** The header provides metadata about the token and specifies the algorithm used for its signature. Common attributes include:

i. alg: The signing algorithm (e.g., "HS256" for HMAC-SHA256).
ii. typ: The token type, typically set to "JWT."

Example: {"alg": "HS256",

"typ": "JWT"}

2. **Payload:** The payload carries the actual data or claims within the token, encompassing details like:

i. User information (username, email, user ID, etc.)
ii. Permissions or roles
iii. Token issuer
iv. Expiration time
v. Token audience

Example: {"userId": 123,

"name": "John Doe",

"admin": true}

3. **Signature:**

◆ A cryptographic signature is generated using the algorithm specified in the header and a secret or private key.
◆ It ensures the token's integrity and verifies that it hasn't been altered.

Example: dsfsdfsdfsdgs73543543543dfsdf (placeholder for an actual signature)

Concatenation:

These three parts are Base64URL-encoded and concatenated in structure of Header.Payload.Signature with dots (.) to create the complete JWT token.

Example of JWT Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4g
RG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

What is JWT Token Vulnerability?

JWT tokens, despite their widespread use due to simplicity and flexibility, are not impervious to malicious attacks. Hacking attempts targeting JWT tokens typically involve efforts to circumvent authentication and access controls, often by exploiting vulnerabilities in their structure or implementation.

Some of the common ways of JWT Token hacking are:

Signature Forgery:

- Falsify the token's signature to create the appearance of validity.
- If the signing key exhibits weakness or predictability, attackers may produce a valid signature for a counterfeited token.

- Through the identification of hash collisions (instances where two distinct messages yield the same hash), attackers could generate a new token featuring desired claims, even with a disparate payload.
- JSON Web Key Spoofing (JWKS Spoofing): In cases where the token utilizes a "jku" header specifying the public key's location, attackers might redirect it to a server under their control, enabling them to sign any token.

Header Manipulation: Manipulating the token header involves altering its properties.

- Modifying the "alg" parameter to a less secure algorithm could facilitate brute-forcing the signature.
- Changing the "typ" parameter to "JWT" might bypass security checks intended for different token types.

Payload Tampering: Objective: Modify the token payload to attain unauthorized access or privileges.

- Decoding and altering the Base64-encoded payload before re-encoding it.
- Injecting malicious content into the padding used in encoding/decoding.
- Acquiring a valid token and replaying it to gain access without undergoing proper authentication.

Other Attacks:

- Man-in-the-Middle (MITM) Attacks involve intercepting and altering tokens during transmission between the client and server.
- Social Engineering Techniques: Trick users into revealing their tokens or clicking on malicious links that facilitate token theft.

What are the impacts of JWT Token hacking?

The ramifications of a successful JWT token hack can be severe and widespread, contingent upon the attacker's objectives and the sensitivity of the information safeguarded by the tokens. The potential impacts include:

- **Account Takeover:** Unauthorized access to tokens enables hackers to assume the identities of legitimate users, posing risks to personal data, financial accounts, and internal systems. This could result in identity theft, data breaches, financial losses, and harm to reputation.
- **Data Breach:** Tokens housing sensitive details such as usernames, passwords, or Personally Identifiable Information (PII) are susceptible to extraction by hackers. This pilfered data may be traded on the dark web or utilized for subsequent cyber attacks.

- **System Manipulation:** Compromised tokens, in severe instances, grant attackers the ability to manipulate data, inject malicious code, or disrupt services within authorized systems.
- **Denial-of-Service Attacks:** Stolen tokens empower attackers to orchestrate botnets, overwhelming servers with authentication requests. This can lead to service outages, adversely affecting legitimate users.
- **Reputational Damage:** A successful token hack undermines trust in a company's security practices, resulting in financial losses from customer attrition and regulatory fines. The erosion of reputation can have lasting consequences for the organization.

How to mitigate JWT Token Vulnerability?

To mitigate the risks associated with JWT token hacking, consider implementing various strategies throughout the token lifecycle:

1. **Secure Signing and Verification:**

- ✧ Utilize strong and unpredictable signing keys, preferably stored in Hardware Security Modules (HSMs).
- ✧ Opt for robust cryptographic algorithms (e.g., HMAC-SHA256 or RS256) for both signing and verifying tokens.
- ✧ Validate all claims within the token, not limiting verification solely to the signature, ensuring comprehensive data integrity.

2. **Payload Content Control:**

- ✧ Minimize the inclusion of sensitive information in the token payload.
- ✧ Explore alternative approaches, such as sessionless authentication, in scenarios where extensive user data in tokens is unnecessary.
- ✧ Set short expiration times for tokens to mitigate potential damage in the event of a compromise.

3. **Secure Communication and Storage:**

- ✧ Enforce HTTPS for all JWT-related communications to thwart eavesdropping and man-in-the-middle attacks.
- ✧ Implement stringent access controls for the signing key and other sensitive components.
- ✧ Securely store token refresh tokens, avoiding storage within cookies or local storage.

4. **Implementation Best Practices:**

- ✧ Employ well-maintained and proven libraries for JWT processing to mitigate common implementation vulnerabilities.

- Conduct routine security audits and penetration testing to identify and rectify potential weaknesses in your system.
- Educate users about phishing attempts and provide guidance on best practices for safeguarding their credentials.

5. **Monitoring and Incident Response:**

- Implement robust logging and monitoring systems to track token activity and swiftly detect any suspicious behavior.
- Establish a clear incident response plan to promptly address and contain breaches in the event of compromised tokens.

References:

- https://jwt.io/introduction
- https://book.hacktricks.xyz/pentesting-web/hacking-jwt-json-web-tokens
- https://portswigger.net/web-security/jwt
- https://www.geeksforgeeks.org/json-web-token-jwt/