WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Providing Authentication using JSON Web Tokens for Enhancing User Security

Ashish Sharma, Dr. Vishal Shrivastava, Dr. Akhil Pandey, Er. Aarti Sharma

Department: Computer Science Engineering
Email: ashish2003152@gmail.com, vishalshrivastava.cs@aryacollege.in, akhil@aryacollege.in, aartisharma.ec@aryacollege.in

ABSTRACT:

In every organization dealing with sensitive user data, safeguarding security and privacy is paramount. Sensitive user data, notably including passwords essential for sessions, demands secure handling and storage. To alleviate the storage overhead within databases, tokens come into play. These tokens manage sessions and encapsulate user information efficiently. For all these operations one of the most famous used tokens is JSON Web Token also called as JWT token. This research paper offers an in-depth exploration of JWTs, covering their introduction, operational mechanisms, underlying algorithms, and more. It also delves into the advantages, drawbacks, potential vulnerabilities and security measures surrounding JWTs, contributing to a comprehensive understanding of their role in safeguarding sensitive data.

1. Introduction:

In the present time the continuous evolution of technology, the internet's impact on the economy, and the presence of various websites, from Wikipedia to e-commerce giants and educational platforms, underscore the need for robust online security. Web security centres on authentication, confirming a user's identity, and authorization, and determining access rights. Tokens, small code pieces containing vital user data for authentication and authorization, are employed upon login and remain active for a user's session, known as 'session tokens.'

However, session tokens often entail time-consuming database queries. In response, JSON Web Tokens (JWTs), lighter and stateless tokens, were introduced. These JWTs authenticate users upon login, authorize actions throughout their session, and are included in HTTP headers, streamlining authorization and data retrieval. With the use of a JWT token, the number of interactions between client and server can be reduced because it stores sensitive data in the token itself. JSON tokens are also very fast while exchanging the data between client and server. It supports various encryption algorithm that provide additional security to the sensitive data in the JSON web token.

2. JSON WEB TOKEN:

JSON web tokens are very lightweight in nature. These tokens are used to exchange data between client and server, authentication and authorization of user in a very simple and effective manner with very minimal interaction with the database. JWT also uses various cryptography encryption algorithms to provide internal security of sensitive data from attackers.

In the past, tokens were merely strings with no inherent value, requiring database access or cache checks for validation. However, JWTs have revolutionized this process by encoding and self-verifying their statements. As a result, stateless and short-lived JWTs have emerged, eliminating the need for database access and simplifying the design. This advancement has greatly improved security and efficiency.

A. JWT recommendation

1. While using JWT for sessions there can be chances of various security risks.
2. JWT should be always used with a secure signature scheme.
3. Encryption algorithms should be used to secure sensitive data in JWT.
4. JWT uses key management techniques.

B. Components of JWT

1. **Header:**

The header is the very first element of the JSON web token or JWT. This serves as the initial part of the JWT token. It contains some important information about tokens like the type of token, various algorithms used etc. JWT can use both symmetric and asymmetric algorithms as encryption algorithms.

2. **Payload:**

Payload is the second element of the JSON web token (JWT). This part of JWT includes information like user ID, the expiration date of the token, the creation date of the token, authorization of the token etc. All this information is coded in base64 format to create the second part of the JWT token.

3. **JSON Signature:**

The third component is the signature of the JSON web token (JWT). A secret key is used to combine the header and payload to create the signature of the JWT token. It is responsible for providing authenticity in the JSON web token. It is also coded in base64 format.

3. Cryptographic Algorithms for JWT:

It is important to note that JSON web tokens are not encrypted. This means that anyone with access to the token can easily decode its contents, which is a potential security risk. JWT have different components and these components can be decoded starting by header and payload, using a base64 decoder. That is why it is not recommended to store sensitive data in the second part of the JWT.

While trying to decode the signature of a JSON web token all contents and information can not be seen directly instead there will be a binary file that will contain all the cryptographic details about the signature. To generate a cryptographic signature any secret phrase is used that gets shared between only client and server. If there is a need for sharing then it is called a symmetric signature while if there is no need for sharing but public and private keys are used then it is called an asymmetric signature.

One of the key advantages of JWT is that it does not require a third party or a database to validate the identity of the user. JWT tokens are self-validating, making the process stateless.

A. Common JWT Signing Algorithms

1. **HS256 (Symmetric Signature):** This algorithm combines HMAC and SHA-256 with a shared secret key. HMAC (Hash-Based Message Authentication Codes) is the most common algorithm used in JWTs. It relies on a shared secret key for both the creation and verification of the token's integrity. SHA-256 is the underlying cryptographic hash function.
2. **RS256 (Asymmetric Signature):** RS256 combines RSA and SHA-256 with a private and unshared secret key. Both RSA and ECDSA (Elliptic Curve Digital Signature Algorithm) are digital signature and asymmetric encryption algorithms used in RS256. RSA is generally faster than ECDSA but requires larger keys for the same level of security. ECDSA is more efficient with smaller keys, making it suitable for small JWTs.
3. **None:** It's important to be aware that JWTs can also use the algorithm "None," which means that no signature validation takes place. This is a critical point to consider as it leaves the JWT vulnerable to tampering by malicious parties and should be used with caution.

4. Working of JWT:

A. If a user wants to use any website, they provide their personal details, including their username, contact number, and other necessary information. Later, they set their password or use alternative sign-up methods, such as Google or Facebook.

B. If a user wants to log in to a website, the website's backend database verifies and authenticates all the provided user details. Upon successful validation, a JSON Web Token (JWT) is generated and issued to the user.

C. When the user intends to make requests to the website's APIs, they must include the JWT in the HTTP header of their request, which is then sent to the database of the application.

D. For any website, the server side is responsible for the validation of the token. If the token is valid and has not been tampered with, the user should be able to access the requested data.

This process ensures secure user authentication, authorization, and data access within the website's ecosystem.

5. Pros of JWT:

1. **Stateless Validation:** JWT offers stateless validation, meaning it contains all the user details required for the validation of a user. This eliminates the need to store and maintain the data on the server side.
2. **Efficiency in Resource Usage:** Since user details are self-contained in the JWT, there's no need to fetch information from a database for every validation. This efficiency conserves memory and CPU time.

3. **Eliminates Database Dependency:** With user information stored directly in the token, there is no reliance on a database for every validation. This reduces the load on the database, enhancing system performance.
4. **Enhanced Security:** Placing the JWT in the HTTP header of every HTTP request enables continuous authorization checks at every step. This multi-level authorization enhances security on the server.
5. **Fast Authentication and Authorization:** JWTs are known for faster authentication and authorization due to their self-contained nature, eliminating the need for frequent database queries.
6. **Session Control:** JSON web tokens are used for creating and maintaining the sessions on various websites. These tokens are created by the website while signing up by the user but remain active for every login by the user.
7. **Less complex:** JWT is much less complex to create and implement, making it a user-friendly authentication method. However, it's essential to handle payload data with care, especially when dealing with sensitive user information.

6. Cons of JWT:

1. **Compromised Secret Key:** JWTs use a secret key for token generation, and if this key is compromised, it can lead to the unauthorized access and manipulation of tokens
2. **Limited Server-Side Control:** Unlike session tokens that can be managed and revoked server-side, JWTs cannot be easily invalidated or logged out. This lack of server control can be a drawback in some scenarios.
3. **Inability to Push Messages:** Since there is no centralized record of logged-in clients on the server-side, it's challenging to push messages or notifications to specific clients.
4. **Cryptography Algorithm Deprecation:** JWT relies heavily on the signature algorithm. While not a frequent occurrence, there have been instances in the past where encryption and signing algorithms have been deprecated or found to be vulnerable.
5. **Data Overhead:** JWT tokens are longer compared to traditional session tokens. Storing extensive information within JWTs can lead to increased token length, resulting in overhead for every validation. Each request must carry the token, impacting data transmission.
6. **Complexity:** JWT involves the use of cryptography and signature algorithms to verify data and extract user information from the token, which can make it more challenging to understand and implement correctly.

7. Security Measures for JWT::

A. **Algorithm Verification:** Always verify if the algorithm is specified in the JWT header to prevent algorithm manipulation.

B. **Minimal Payload Data:** Avoid loading unnecessary sensitive data in the payload of JWT. Include only the data required for authentication and authorization.

C. **Secure Secret Key:** Carefully select a secret key to ensure that it is robust against brute force attacks.

D. **Token Visibility:** Remember that JWTs are not encrypted but encoded, and the contents are visible to anyone with the token. Keep sensitive data secure.

E. **Storage Consideration:** Decide on the storage location for JWTs based on your implementation requirements, whether in local storage or cookies.

F. **Avoid "None" Algorithm:** Ensure that the JWTs you create do not accept the "none" algorithm in the header, as it can compromise security.

G. **Keep Tokens Short:** Shorter tokens reduce overhead in terms of both time and space, enhancing efficiency.

H. **Algorithm Selection:** Familiarize yourself with compatible cryptography algorithms for JWT and choose the one that best suits your application's security needs.

8. Methodology:

1. **Literature Review:** The research paper begins with an extensive literature review to understand the existing concepts and technologies related to web security, token-based authentication, and JWTs. This phase involves collecting relevant publications, articles, and related work.
2. **Data Collection:** The study collects information about JSON Web Tokens, their structure, and their usage. This involves analyzing JWT specifications and gathering examples to understand how JWTs work.

3. **Analysis of JWT Security Measures:** The research involves an in-depth analysis of the security measures, pros, and cons of JWTs as discussed in the paper
4. **Vulnerability Assessment:** The vulnerabilities of JWTs, including potential attacks and weaknesses, are assessed by examining and testing various scenarios mentioned in the paper.
5. **Security Best Practices Evaluation:** The study evaluates the security measures, guidelines, and best practices mentioned in the paper. This includes an assessment of recommendations for JWT usage.

9. Case Study: Improving Web Application Security with JSON Web Tokens (JWT)

A. **Background:** As the digital world continues to expand, numerous web applications handle sensitive data, including user passwords, financial transactions, and personal information. Ensuring the privacy and security of this data is of utmost importance. Historically, web applications have relied on session tokens, which involve storing user data in databases, causing overhead and potential security concerns.

B. **Challenge:** The challenge lies in providing a secure, efficient, and stateless solution for user authentication and authorization, minimizing database overhead while ensuring the privacy and integrity of sensitive information.

C. **Solution:** The implementation of JSON Web Tokens (JWTs) offers a viable solution to the challenge. JWTs are lightweight, self-contained tokens that contain user information required for authentication and authorization. They consist of three components: header, payload, and signature. The header specifies the token type and the algorithm used for the digital signature. The payload contains user-specific data, and the signature is used to verify the integrity of the token.

D. **Conclusion:** Implementing JSON Web Tokens (JWTs) in web applications provides a secure, efficient, and stateless solution for user authentication and authorization. However, developers must remain vigilant regarding the vulnerabilities and security measures associated with JWTs to ensure robust protection of sensitive user data.

10. Conclusion & Future Work:

Due to the vulnerabilities of JWT, In spite of using all the security measures to use JWT tokens, There are many circumstances where the security measures may fail. Thus there is a need of CSRF tokens which take care of all the security flaws of JWT. Future work involves research in storage methods of JSON web token. Also the difference between JWT and CSRF token and how CSRF token overcomes vulnerabilities of JWT.

11. References:

1. Ch. Jhansi Rani and SK. Shammi Munnisa, "A Survey on Web Authentication Methods for Web Applications," International Journal of Computer Science and Information Technologies, Vol. 7 (4), 2016.
2. Muhamad Haekal and Eliyane, "Token-based authentication using JSON web-token on SIKASIR RESTful web service," International Conference on Informatics and Computing (ICIC), IEEE (2016).
3. Hardt, D., "The OAuth 2.0 Authorization Framework," RFC 6749, RFC Editor, October 2012.
4. Yjvesa Balaj, "A Survey: Token-Based vs. Session-Based Authentication," Article, September 2017.
5. Yung Shulin, Wang Shaopeng, Hu Jeiping, and Cai Hungwai, "Implementation on Permission Management Framework based on token through Shiro," 2017 International Conference on Computer Technology, Electronics, and Communication (ICCTEC).