Journal of Information Systems Engineering and Management
2025, 10(8s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Vulnerabilities and Encryption Applications of JWT-Based Authentication Methods

Seok-Woo Jang[1]Sang-Hong Lee[2]*
[1]*Department of Software, Anyang University, Republic of Korea*
[2]*Department. of Computer Science & Engineering, Anyang University, Republic of Korea*

ARTICLE INFO

Received: 16 Oct 2024

Revised: 15 Dec 2024

Accepted: 26 Dec 2024

ABSTRACT

Most modern web applications send and receive data using the HTTP (HyperText Transfer Protocol) transmission method. HTTP has the characteristic of being non-connected. Due to this characteristic, the client must inform the server every time that he or she is an authenticated user. There are currently several authentication methods for this, such as cookies, session cookies, and tokens. Currently, a JSON-formatted token called JWT is being widely used. JWT is used in domains such as QR check-ins and social logins due to its fast and simple characteristics. However, JWT has a critical security vulnerability because it contains the user's information as it is in the token. If an attacker steals a token on the network, the user's information can be easily obtained by simply base64 decoding. The purpose of this study is to protect the user's information from the attacker by encrypting the payload part containing the user's information and the header part containing the hash information.

Keywords: JWT, HTTP, JSON, encryption

INTRODUCTION

Recently, the web is being re-examined due to the expansion of the IoT industry and mobile application market[1]. The characteristic of programs in the web environment is that they can be accessed anytime and anywhere, and this characteristic is becoming more prominent as the mobile ecosystem continues to develop. As the number of users of web programs increases, it can be said that they are exposed to attacks by hackers as they can be accessed by an unspecified number of people. Many of these attacks target vulnerabilities in the authentication system to steal user information of the target web service.

The most commonly used method for sending and receiving data in the web environment is HTTP (HyperText Transfer Protocol)[2]. Since HTTP basically has the characteristics of being non-connected and stateless, the client needs a means to authenticate itself for each request. Concepts such as cookies, sessions, and tokens emerged for this characteristic. Among them, the 'JWT (Json Web Token)' authentication method, which is a fast and lightweight authentication method, is being used a lot recently[3].

Unlike the session-cookie method, JWT does not require a separate DB for authentication on the server side, and authentication/authorization can be performed with only the JWT sent by the client. However, since the token itself contains the client's information, if the token is stolen, the user's personal information can be easily revealed. In this study, we try to protect the information contained in the token by wrapping and encrypting the header and payload, which are components of the JWT token. In conclusion, the purpose is to design the token so that even if it is stolen by a hacker, the information contained in the token cannot be obtained.

RELATED RESEARCH

2.1 Session/Cookies

Currently, the most commonly used authentication methods are session/cookie and JWT. Among them, session/cookie stores the session ID on the server, so it can implement various functions other than authentication, such as checking the number of current users and restricting login accounts.

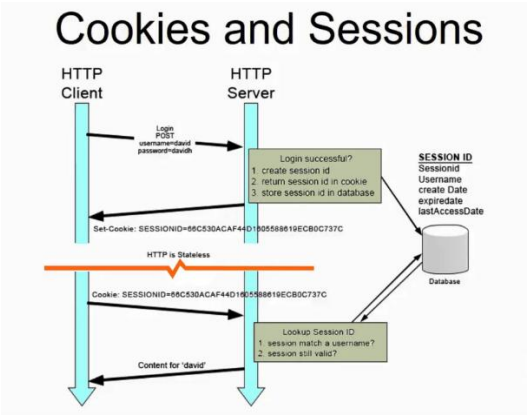

Figure. 1Session-cookie authentication process

The session/cookie method is a user authentication method that uses the concept of 'session', which is a storage on the server in Fig. 1. When a user initially attempts to log in, the server receives information such as the ID and PW and checks whether the user is valid. If the login request is from a valid user, the server generates a unique session key that does not overlap with other users and stores information that can identify the user as a value. Then, the unique key value generated for the user (browser) is stored in a cookie and transmitted.

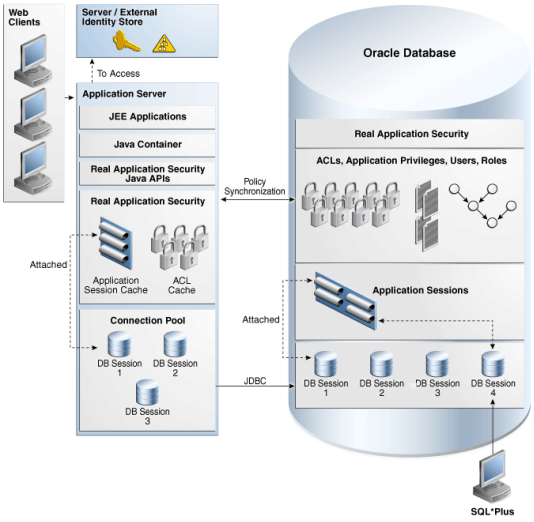

Figure. 2 Session Database Example

Afterwards, when the user who received the cookie sends a request to the server, the user sends a cookie with the session key value received from the server as the value to the server, and the server sends a request to the session DB to check whether the user's session key value received from the cookie is valid. Afterwards, if the session is valid, it is determined that the user is authenticated and the user's request is processed.

The session/cookie authentication method operates by storing the session key and user information [ex) user pk-id] in the form of Key-Value in Fig. 2. Due to this characteristic, when actually operating a service, session information is not stored in the memory of the WAS (Web Application Server), but is managed by a separate database[4].

The biggest disadvantage of the session/cookie method is that additional costs are required to build a separate DB to store session information. In particular, if the domain has a lot of user access, additional DBs must be built as the number of users increases. However, since the DB has a physical limit to storing data, if a lot of traffic occurs due to events, etc., the DB itself may go down and cause service disruption. In addition, since a connection to the session DB is also required to identify user information, additional overhead occurs, which affects the request-response performance.

Table 1. The advantages and disadvantages

	advantages	disadvantages
Session & Cookie	- Superior security compared to JWT method - Session control possible on the server side - Low network load	- Server load and cost due to session DB operation
JWT	- No need for separate storage for authentication - Fast authentication processing without separate I/O operations - Excellent scalability	- Network load increases as token length increases - Difficulty in forcibly expiring specific tokens

The session/cookie method also has advantages. Since the server stores information on logged-in users, it is easy to identify the number of currently logged-in users. In addition, if an unauthenticated user repeatedly sends malicious requests, the session can be expired, blocking additional access. The simultaneous connection limit currently used in many streaming services can also be implemented in this way. Additional functions such as these can be implemented by operating the DB separately. The advantages and disadvantages of each of the session/cookie and JWT authentication methods are shown in Table 1.

2.2 Json Web Token(JWT)

The JWT technique is a standard JSON object data format for more secure transmission of information exchanged between servers using a simpler and more independent method, and is more reliable because it is digitally signed using the HMAC algorithm or RSA[5][6]. In addition, JWT contains information in JSON format data units called claims, and has the advantage of using claims to confirm roles and permissions without accessing a database or external system each time a request is made[7].

According to the official OAuth homepage, OAuth 2.0 introduces and uses the JWT technique, but there is a clear difference. The OAuth framework stores the issued token in the service DB for comparison and cannot contain any content other than whether it is authenticated, but JWT can always be used once and can transmit a lot of information in JSON format data, so it has the advantage of being able to use information such as the token's expiration date in Fig. 3.

Figure. 3 The structure of JWT

JWT consists of three parts: header, payload, and signature, and each part is classified based on a period (.). The header is generally in the form of a JOSE header containing a `typ` value for the token type and an `alg` value for information about the hash algorithm, and information about the algorithm for encrypting the token from the authentication server is transmitted to the server in the header. The payload has a claim containing the data information to be used and additional metadata values. The actual authentication server generates information through credentials and uses this on the server. The header and payload parts are encoded in base64url format and each part is connected with a period (.). The signature part encodes the header and payload values in base64url format and contains the result value by applying the hash algorithm together with the secret KEY value.

Currently, open source libraries for development languages such as Python, PHP, Java, PHP, and Node.js are provided through online project hosting services such as Github. Tim McLean, an honorary security researcher at Auth0, has confirmed that these open source libraries for various development languages have serious vulnerabilities that bypass the verification step.

The biggest problem with the JWT technique is that it discloses the algorithm used by the server. The server determines which hash algorithm to use through the value of the ‘alg’ attribute in the header payload inside the JWT token. However, a header encoded simply in Base64 can be easily exposed to an attacker, providing the corresponding web service authentication system with which algorithm to use for authentication. In addition, if the value of the ‘alg’ attribute is modified due to a man-in-the-middle attack, the attacker will be helplessly using the algorithm intended by the attacker to verify access rights, and if the corresponding user is a super administrator or higher, the ripple effect will be very large.

The “none” algorithm is used when a user has already been granted access through another communication in a communication flow for user access rights. An attacker can easily manipulate the header part to change the existing `alg` attribute value. For example, if it is set to "alg": "HS256", an attacker can attack by changing it to “none” and sending the desired claim. After the authentication server approves the user credentials, a claim is created in the payload part, and there may be critical information that the attacker can refer to at this time. Since the header part and payload part are simply Base64 encoded, an attacker can easily check the contents. For example, if the address of the user's record is contained in the payload claim, an attacker is likely to infiltrate the inside through another attack and steal the user data of that address.

ENCRYPTED JWT DESIGN

The existing JWT structure has a header, payload, and signature structure in Fig. 4. In particular, the data included in the header and payload contains very security-sensitive data. The header contains information on the hash algorithm used in the signature in the attribute value called ‘alg’. Therefore, if this algorithm information is leaked, attackers can attempt various attacks. In the case of the payload, various information for verifying authorization in the application is included in the value as a data unit called claim. Even if the token is no longer valid, attackers can obtain various information necessary for the attack with just the values contained in the token[8][9].

The parts that were problematic in terms of security in the existing JWT were the header and payload. The new encrypted JWT has a structure of header+payload, signature, rather than the existing JWT structure of header, payload, and signature. Since the header and payload, which are sensitive information, are encrypted, even if an attacker obtains the token value, he or she will not be able to obtain meaningful information. The JWT authentication method must be able to know the user's information through the token value in Fig. 5. Therefore, a two-way algorithm that can be encrypted and decrypted was selected, and among them, the following three algorithms with excellent security were selected. 'AES256', "3DES", "RSA". First, AES256 (Advanced Encryption Standard-256) is an encryption method established by the National Institute of Standards and Technology. AES is a representative two-way symmetric key encryption algorithm that has been widely used worldwide since it was adopted by the US government. Also, it is included in the ISO/IEC 18033-3 standard and is the first algorithm to be publicly disclosed among the algorithms approved by the US National Security Agency for use in top secret.

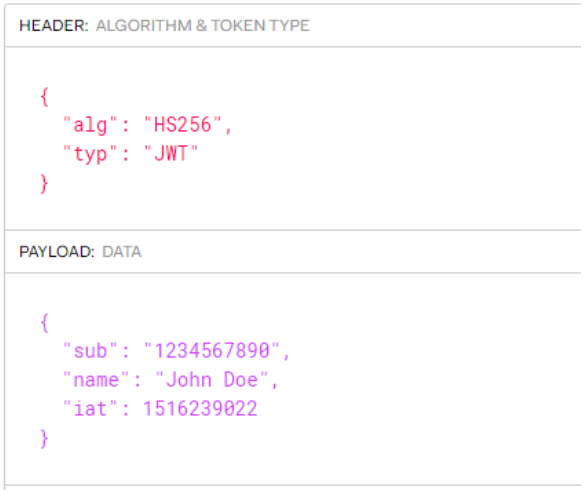

Figure. 4 The header, payload of JWT

3DES (Triple DES) is a common name for the Triple Data Encryption Algorithm block cipher that applies the Data Encryption Algorithm (DES) three times to each data block. It was developed to complement the security vulnerabilities of the Data Encryption Algorithm (DES), and it is an algorithm with some performance issues because it applies encryption three times.

Unlike the previous encryption algorithms, RSA is one of the representative public key algorithms and is the first algorithm that allows not only encryption but also electronic signatures. The stability of the RSA encryption system is based on the difficulty of factoring large numbers, and it is currently the most widely used encryption algorithm in SSL/TLS and Internet banking.

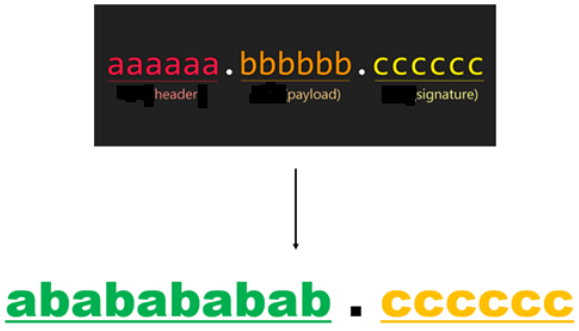

Figure. 5 The structure of encrypted JWT

The biggest feature of the JWT authentication method is that it allows fast authentication processing, so we measured the encryption/decryption performance of the same three algorithms examined above. The environments used for performance measurement were jdk11, JUnit5, and Intellj Ultimate. In order to reduce the deviation, the time was compared after performing encryption/decryption 100,000 times. The average time taken when encryption/decryption was repeated 100,000 times was AES256 – 1,055 / 3DES – 2,742 / RSA – 18,453. (Unit: ms) In conclusion, the AES algorithm took the shortest time for encryption/decryption, and showed about 2.6 times better performance than 3DES and about 17.5 times better performance than RSA. Based on these results, AES256 was used for JWT encryption.

ENCRYPTED JWT IMPLEMENTATION

To create an encrypted JWT, first, a secret key for AES256 encryption is created internally in the server in Fig. 6. If multiple servers are in operation, the secret key value is shared internally using techniques such as session clustering. Afterwards, when a login request comes in from a client, it is checked through a DB connection whether it is a valid user login request. If it is a valid login request, an existing JWT with a header, payload, and signature structure is created based on the user's information. The header and payload values are extracted from the created JWT, combined, and then AES256 encrypted[10][11]. The encrypted value and signature are combined and sent back to the user as a Secure Cookie. Afterwards, when the user sends a request to the server, the Encrypted JWT

value contained in the Secure Cookie is automatically transmitted together, and the server verifies this to confirm that the user is authenticated. In order to apply a JWT encrypted with AES256, the user authentication method of a web application called ModuForm was changed. Fig. 7 is one of the screens of a survey application called 'ModuForm'. All operations in this application are limited to logged-in users, and operations such as ‘edit, delete’ and ‘edit user information’ of survey and response posts are designed to be performed only by authorized users.

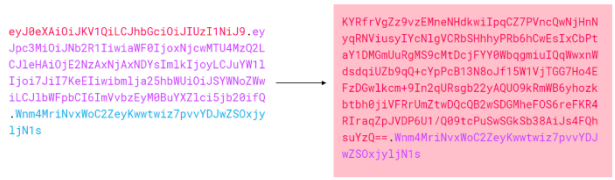

Figure. 6 Plain JWT & Encrypted JWT

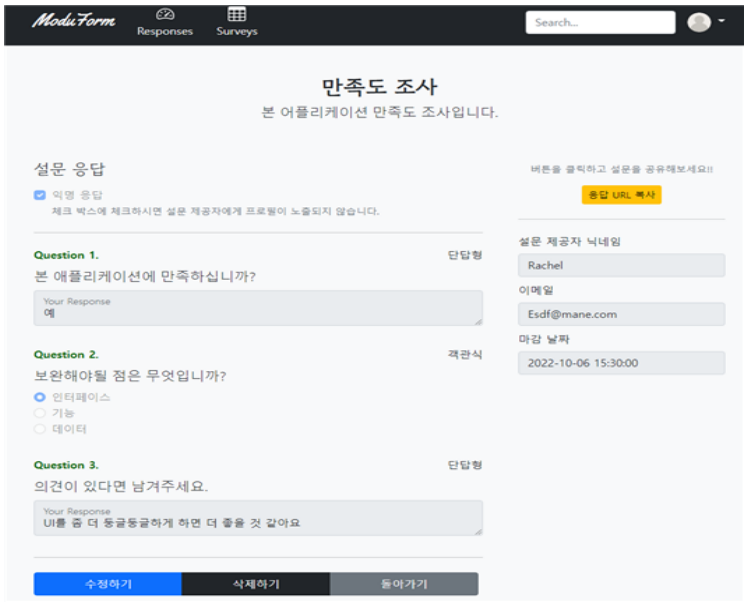

Figure. 7 Survey Response Screen

Fig. 8 shows the relationship of the classes responsible for issuing Encrypted JWT tokens. First, the user's login request is sent to the WAS, and the user Id and password values are sent to the User Controller through the Dispatcher Servlet. The database is accessed to check if there is a user with the corresponding values, and if it is a valid user, a cookie containing the token value is created in Jwt Handler. After that, the Encrypted JWT is created and encrypted in Custom Jwt Provider, and the token value is contained in the created cookie and sent to the client.

Fig. 9 shows the relationship of the classes responsible for Encrypted JWT token authentication. The user's request authorization was verified by using the Interceptor function provided by Spring without using the Spring Security framework. It is a form of existing JWT authentication method with decryption added.

Table 2 measures and compares the time required to generate and parse a general JWT and the time required to generate and parse an encrypted JWT. The test was conducted in jdk11, IntelliJ, and JUnit5 environments, and generation and parsing were performed repeatedly to reduce deviation.

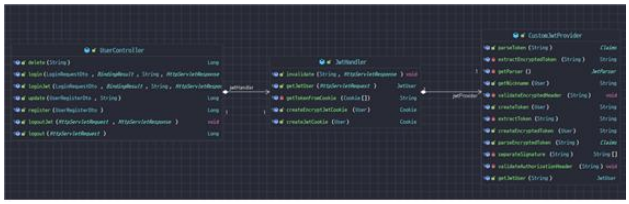

Figure.8 Encrypted JWT Issuance Class Diagram

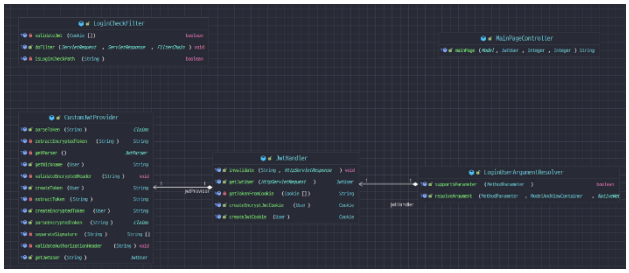

Figure. 9 Encrypted JWT Authentication Class Diagram

Table 2. The advantages and disadvantages

	Previous JWT	Encrypted JWT
1st trial	659 ms	1034 ms
2nd trial	680 ms	1115

CONCLUDING REMARKS

In this paper, we examined the authentication vulnerability of JWT and applied an actual encryption algorithm to encrypt and operate JWT. Through this, even if an attacker steals a token, it was possible to prevent the user's important information from being stolen through the token value. Through this, we can expect to build a web application authentication system with a higher level of reliability, safety, and security.

Furthermore, the encryption method can be considered in other token authentication methods, and this will allow the overall authentication method to be designed in a more security-friendly direction. Up until now, the AES256 technique has been used for JWT encryption. Since AES256 is a symmetric key encryption algorithm, all servers in web applications using the HTTP method must have the same Secret Key. Of course, the possibility of the internal network being exposed to the outside is low, but if all servers have the same Secret Key, there is a high possibility of security problems occurring in the process of sharing and applying the Key.

In addition, there are still many issues to consider in terms of actual operation, such as how to set the Secret Key replacement cycle and how to apply the Key when deploying the application. Also, if the token itself is stolen, the attacker can access the server's resources without knowing the user's ID and password. This problem is a vulnerability in the authentication methods of all applications that use HTTP, such as cookies, sessions/cookies, and tokens. Cookies and sessions are trying to improve this by introducing an expiration time, and JWT is trying to improve this by introducing an authentication procedure using the ACCESS and REFRESH Token methods, but the problem is that the user can make a request to the server with their authority even for a relatively short time. Overcoming these problems is a future task.

Conflicts of Interest

The Authors declare no conflict of interest.

Author Contributions

Conceptualization, methodology, and program development: Seok-Woo Jang,oversight, review, and editing: Sang-Hong Lee

REFERENCES

[1] Hong-Pil Kwon, Jae-Cheol Ha, "Power Analysis Attack of Block Cipher AES Based on Convolutional Neural Network", Journal of the Korea Academia-Industrial cooperation Society(JKAIS), Vol. 21, No. 5, pp. 14-21, 2020.

[2] K. S. Yoon & Y. H. Kim. (2019). A design and implementation of integrated content management system based on microservices architecture. Korea Information Processing Society, 8(3), 97-108.

[3] JWT. https://jwt.io/introduction.

[4] H. D. Yoo, Y. H. Kim, C. G. Song, H. E. Kim & B. J. Choi. (2018). A study on the comparative performance analysis of open source web server using JMeter. Korea Information Processing Society, 25(2), 2-4.
[5] M. Jones. RFC7519 – JSON Web Token (JWT). IETF. May. 2015.
[6] T. McLean. Critical vulnerabilities in JSON Web Token libraries, March 31, 2015
[7] Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519. Retrieved from IETF Tools.
[8] FIPS 197, “Advanced Encryption Standard (AES)”, November 26, 2001.
[9] Edwin NC Mui, "Practical Implementation of Rijndael S-Box Using Combinational Logic", Custom R&D Engineer Texco Enterprise Pvt.Ltd, 2007. 99.
[10] D. Hart, "The OAuth 2.0 Authorization Framework", IETF, RFC6749, Oct. 2012.
[11] Byung-kil Byun, "All-IP User Authentication and Authorization Mechanism by OTP in the SSL-VPN System", Journal of Korean Institute of Information Technology, Vol. 9, No. 9, pp. 139-146, Sep. 2011.