Insights of JSON Web Token

Pooja Mahindrakar, Uma Pujeri

Abstract—In almost every organization where user sensitive data is available, security and privacy of the data plays a vital role. As far as computer science is concerned, it is just a game of saving data in unrecognizable format and accessible to authorized person. User sensitive data mainly includes passwords which are required for the sessions but need to be handled and stored safely.As storage of these information is overhead in database, Tokens are generated which handles sessions and also self contains user details. One of such widely used stateless token is Json Web Token. This paper deals with the introduction, working and algorithms of Json web token. Also pros, cons, hacking possibilities, Proper usage and security measures of JWT are discussed.

Keywords— token, authentication, JWT , security, privacy, sessions, encryption.

I. INTRODUCTION

Over the years, technology has evolved drastically. Internet has main impact on country's economy. We find many websites(one with static data) and web applications(one with dynamic data and user interactions) on internet.\cite{b3}These spread over a wide range right from Wikipedia where every single information is available to the huge shopping web applications like amazon,flipkart etc where huge financial transactions are involved. Also not to forget are various educational sites where paid and unpaid courses are available.In all these kinds, huge capital is involved digitally thus hacking by intruders is an obvious task. Hence security for these sites plays a crucial role in this era.

Security over internet involves privacy,storage and validation of sensitive client information.Two terms are widely used when we talk about web security, one is authentication and the other is authorization. Authentication means verifying if the person is the one who he claims to be\cite{b1}.Authorization means verifying if the person is the one who has rights to access the requested information . Thus to achieve the two terms authentication and authorization, a term called 'token'\cite{b6} was introduced. Token is merely a piece of code containing client's sensitive information required for authentication and authorization before fetching any sensitive data from the web servers. This is required to avoid illegal access to sensitive data.

'Token' usually is used as soon as a person logins into the website and ends as the client aborts the session.Such token are called session tokens.These tokens store client data into database and fetch details each time to validate the client.The major drawback of such tokens is the time .To secure each action of the client,database hits are involved which is a major overhead. Hence new token type is introduced called Json Web Tokens. These tokens self contain client sensitive data and database overhead are completely abolished.

Revised Manuscript Received on February 15, 2020.

Pooja Mahindrakar, School of Computer Engineering and Technology, MIT World Peace University, Pune, Maharashtra, India. E-mail: mahindrakarpooja@gmail.com

Dr. Uma Pujeri, School of Computer Engineering and Technology, MIT World Peace University, Pune, Maharashtra, India. E-mail: uma.pujeri@mitcoe.edu.in

Thus these tokens are called light weight and stateless tokens. These tokens are stored either in local storage or in cookies. Whenever a client logins to the web application, token authenticates the client and each time during the session, if any actions are performed, tokens authorize the client without database overhead thus making more efficient security to web applications. These token are included in http header. Whenever any client requests sensitive data, http request is generated with JWT token in the header so as to ease authorization and timely fetch of requested data.\cite{b6}

II. JSON WEB TOKEN

Json Web Token(JWT) is a lightweight means of exchanging data between two parties so as to ease authentication,authorization and security. Each JWT statement is stored as a json entity and each Json entity i used as plaintext of Json Web Encryption or as a payload of JSON Web Signature (JWS) which allows claims to be digitally secured and authenticated with the Message Authentication Code (MAC).

A few years ago, a ‘token’ was only a string with no inherent value before the revolution of JWT, e.g. 2pWS6RQmdZpE0TQ93X. This token was later checked in a server where token statements were stored. The drawback of this approach is that each time the token is used, database access (or a cache) is required.

Now a days, JWTs encode the statements and check their own statements (by signing). Thus stateless(read: self-contained, don't rely on anyone else) short-lived JWTs are evolved. There is no need of database. Thus Database burden is eliminated and design is simplified because only the server that issues the JWTs has to think about entering the database / persistence layer (the refresh token).

A. JWT guidelines

- The JWT must always use correct signature scheme.
- If data is sensitive, encryption should be done.
- Jwt requires proper key management.
- If jwt used for sessions, risk can be introduced.

B. Components of JWT

Json Web Tokens are never encrypted but are encoded with base 64(UTF-8). JWT consists of three parts, header,payload and signature. Each part is separated by full stop(.) and encoded independently. that means header alone if tried to decode base 64, valid output is obtained. not necessary to consider entire token for decode. Part by part is valid.Let us understand each component of json web token in details.\cite{b1}

1) Header: First component of json web token is header. It consists of information viz algorithm, token type etc. Algorithm type varies from symmetric HS256 to asymmetric RS256 depending upon the usage. this bundle is encoded in base 64 format and placed as a first component of jwt token.

JSON Header Format:

```
{
"alg" : "RS256",
"type":"JWT"
}
```

2) Second component of json web token is payload. It consists of information viz client id, company id,access right of the token and the expiry date of the token.This bundle also is encoded in base 64 format and placed as second component of jwt token.Json Payload format:

```
{
        "sub": "0987654321",
        "name": "Joe",
        "iat": 9875456678
}
```

3) JSON Signature: Third component of Json web token is signature. Both header and payload are bounded together with a secret key to a signature.This signature maintains the integrity of the token.JSON Signature Format:

```
        HMACSHA256(
         base64UrlEncode(header) + "." +
         base64UrlEncode(payload),
) secret base64 encoded
```

Authorization Bearer: A Bearer token is only an arbitrary string,used for permission. Bearer token can be jwt when jwt is used for authorization.

eyJhbGciOiJbcEcbyjn6kbRghuR843Rfh6t89Rt709u9.ey3ORT4tfvbR843Rfh6t8Tgbj578EnjyTnhyufQ.SftykmR843Rfh6t8bhj3DY4ugbmDgjhvhy46t87FD

Types of Signature:

1) Symmetric signature :This signature is produced using HMAC function and verified using only one secret key.Such signatures work well within same application.
Asymmetric signature :This signature uses dual keys one is secret key used for signing and other is public key used for verification.

III. RELATED WORK

In the publication "Token based authentication using Json web-token on SIKASIR RESTful web service"\cite{b1}, author explains the basic of Json Web Token and its structure also describes the usage of JWT in Sikasir

(SME) model."A survey:Token-Based vs Session-Based Authentication"\cite{b2}, author explains the evolution of stateless token over session based token with some details of Json Web Token.

"OAuth 2.0 Authorization Framework"\cite{b3} the author explains the OAuth framework in brief and describes how JSON web token is used for token-based authorization and in the publication "OAuth 2.0 Authorization Framework"\cite{b5} Bearer Token, authorization using JWT token as bearer token is described ..The paper "Simplified Authentication and Authorization for RESTful Services in Trusted Environments"\cite{b7} describes authentication and authorization in restful applications. "The OAuth 2.0 Authorization"\cite{b9} the author describes the value of the Bearer variable that has a token on it, attached to the application header that is used by the user.

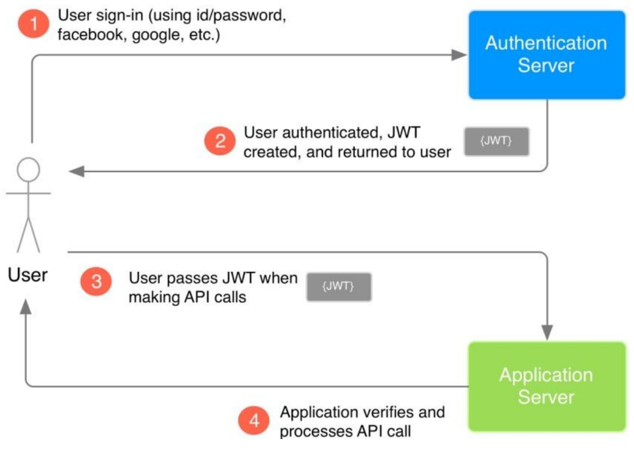
IV. CRYPTOGRAPHIC ALGORITHMS FOR JWT

Jwt contents are only encoded in base64 format. Those are not encrypted! This reason any base64(UTF8) decoder can easily display the contents of the Jwt. Jwt can be decoded in parts that is header, payload and signature can be decoded separately using a base64 decoder.As decoding task can be easily performed, including sensitive data into token is at high risk.Hackers can easily obtain personal sensitive data and misuse it in illegal ways. Thus no sensitive data must be included in the payload part of Jwt. When we try to decode the signature part of Jwt, we cannot see the contents of those instead we can see a binary file downloading which contains cryptographic details. On the other hand one can directly decode and see contains of header and payload. The cryptographic signature is generated using a secret passphrase which is shared only between sending and receiving parties. If sharing is involved it is a type of symmetric signature if secret is not shared it is asymmetric signature where public and private keys are used to play the scenario. JWT does not require third party or database to validate the person who he claims to be, instead it self validates making itself stateless.

A. The common algorithms used by JWT

- HS256 (symmetric signature): This is a combination of two hashing algirithms namely, HMAC and SHA256 along with a shared secret key.Most common algorithm used in JWt is HMAC. Hash-Based Message Authentication Codes (HMACs) are a collection of shared key based signing algorithms . These include hash function like SHA256.
- RS256(Asymmetric signature): This is a combination of RSA and SHA-256 along with a secret key which is private and unshared. Both RSA and ECDSA are digital signature and asymmetric encryption algorithms. With these asymmetric algorithms the possibility of creating a new one for decrypting or verifying a message is least. This is key for certain use cases. The main difference between RSA and ECDSA is speed and key size. ECDSA requires small sized keys to achieve the same level of security as RSA. This makes it a great choice for small JWTs. RSA, however, is usually faster than ECDSA.
- Algorithm can also be"None" which means that no signature validation takes place. This is a major key for any hacker to bypass validation process.

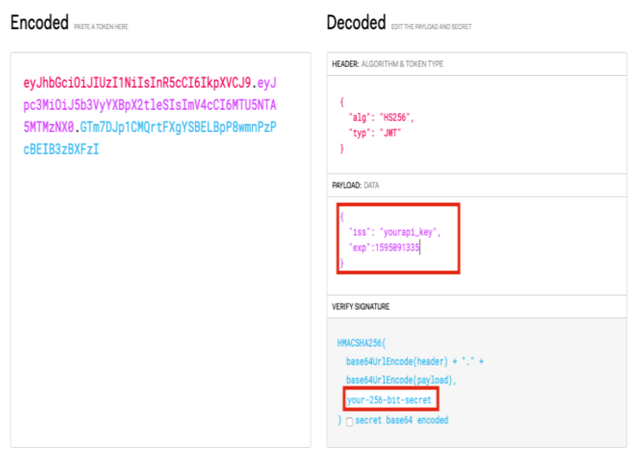
B. Working of JWT


1 User sign-in (using id/password, facebook, google, etc.)
Authentication Server
2 User authenticated, JWT created, and returned to user
{JWT}
User
3 User passes JWT when making API calls
{JWT}
Application Server
4 Application verifies and processes API call

- When user signs up initially into a website, all his details containing his username, contact number etc are entered. Later password is set.

Many a times, sign up is done using google, facebook or other account.

- One the user sign in the website at the database end, all the user details are authenticated and verified. On validation, JWT token is generated and token is sent to the user.
- When user want to make call to any API, He has to pass JWT in the http header and send to application server.
- Application server validates the token. If validated, then user is authorized to access the requested data thus the data is returned to the user.


Encoded PASTE A TOKEN HERE
Decoded EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
PAYLOAD: DATA
VERIFY SIGNATURE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pc3MiOiJ5b3VyYXBpX2tleSIsImV4cCI6MTU5NTA
5MTMzNX0.GTm7DJp1CMQrtFXgYSBELBpP8wmnPzP
cBEIB3zBXFzI
```

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

```
{
  "iss": "yourapi_key",
  "exp":1595091335
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

JWT.io is a website which decodes any jwt token from base 64 to readable format. It displays the decoding result into three parts header,payload and signature. If any algorithm is involved in formation of the token, the algorithm details is displayed in the header part.The secret key used is kept secret and not included in jwt. Only those parties possessing secret key can verify the user.The integrity of jwt is maintained solely by the signature. Thus secret key of signature plays a vital role.If secret key is compromised, any intruder can modify details and reform a new jwt.

V. PROS

- JWT provides stateless validation: JWT self contains user details required for validation.It is termed stateless because jwt does not maintain any state as it provides self-validation.
- No need of memory or cpu time: For each validation, user details are not required to be fetched from database, thus memory and cpu time is saved.
- No need of database: As Token itself store necessary user details, for every validation, database is needed.
- Validation of sender for every http request when jwt is placed in http header: Whenever any http request is placed by the user, jwt from local storage or cookie is placed into http header and thus every step authorization is checked thus enhancing security to server.
- Faster authentication and authorization: Termed faster because of self -validation only.
- It can be used for session control: jwt are created during sign up but for each login it is active acting as a stateless session token.
- Simpler to use if careful: JWT is easy to code but while placing payload data sensitivity of user information must be taken care.

VI. CONS

- Compromised Secret Key: Usually key rotation system is used for selecting secret key for jwt generation. Once secret key chain is compromised, tokens are easily hacked.
- Cannot manage client from the server: If a user in any case wants to log out, we cannot simply delete token as in case of session token which is store in database. Also, we cannot delete user id from the table as it any create multiple dangling pointer in database.
- Cannot push Messages to clients (Identifying clients from server): As we have no record about the logged-in clients on the DB end, we cannot push messages to all the clients.
- Cryptography algorithm can be deprecated: JWT relies completely on the Signing algorithm. Now, though it is not frequent, but in the past many Encryption/Signing algorithms have been deprecated.
- Data Overhead: JWT tokens are longer than that of a normal Session tokens. We have to very carefully store information into jwt as it linearly increases in length. And, if length increases, overhead is created for every validation. Remember, each request needs the token in it for request verification.
- Complicated to understand: JWT uses cryptography Signature algorithms to verify the data and get the user-id from the token.

VII. ATTACKING JWT

- Exposure of sensitive data: JWT.io helps to decode the json web token. If user sensitive information is stored in the token,attacker can decode the token and misuse the information obtained.
- Change in the signing algorithm: JWT header if contains algorithm especially asymmetric algorithm like RS256(which contains private key as secret key to create signature and public key to verify the signature),then it can be changed to HS256(symmetric algorithm) where publicly available public key can be used as secret key to generate and verify the signature.
- Changing algorithm to none: Most of the JWT's accept algorithm as none by default where no signature is present. Hackers try to eliminate the algorithm by introducing alg: none in the header. This may sometimes work to skip the jwt verification.
- If we specify algorithm in the header and leave out signature part in the token, sometimes applications may consider the token as correctly signed.
- compromising secret key: There are many tools that can brute force the HS256 signature on these tokens:
  - .NET implementation.
  - Python script (PyJWT) to do the decoding.
  - John the Ripper (a tool)

VIII.SECURITY MEASURES

- Always check if the algorithm is specified in JWT header.
- Never load any extra user sensitive data in payload part of JWT other than the data required for authentication and authorization.

- Always choose secret key very carefully so that brute force on the key should be difficult.
- Always remember jwt is encrypted, it is only encoded and clearly visible to the ones who obtain the token.
- Storage of JWT must be decided carefully based on the implementation requirements either in local storage or in cookies.
- Try to make sure the the jwt you create should avoid accepting alg = "none" in the header.
- Try to always keep token short as that overhead in time and space is reduced.
- Try to know all the cryptography algorithms which are compatible with JWT and use the one that best suits your application.

IX. Result

Many JWT attacks were performed and vulnerabilities in JWT are noted. Firstly the none algorithm in JWT will disable signature verification thus leading to tampering of sensitively data. Then the jwt.io website clearly displays the user credentials which is hashed but yet visible and can be stolen. Later the secret key used for digital signature of JWT can be hacked using many tools. Thus secret key generation and storage should be given major importance.

X. 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.

ACKNOWLEDGMENT

Sincere thanks to my guide Dr. Uma Pujeri for providing immense support in this research. Also, I thank Dr. Balaji Patil sir for helping me the best way possible.

REFERENCES

1. Muhamad Haekal,Eliyane ,"Token based authentication using Json web-token on SIKASIR RESTful web service". International Conference on Informatics and Computing (ICIC) IEEE(2016)
2. Yjvesa Balaj,"A Survey: Token-Based vs Session-Based Authentication " Article September 2017
3. Hardt, D.: “The OAuth 2.0 Authorization Framework.” RFC 6749, RFC Editor, October 2012.
4. Yung Shulin,Wang Shaopeng,Hu Jeiping,Cai Hungwai, "Implementation on Permission Management Framework based on token through Shiro" 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)
5. Ch.Jhansi Rani ,SK.Shammi Munnisa "A Survey on Web Authentication Methods for Web Applications"(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (4) , 2016
6. Xiang-Wen Huang, Chin-Yun Hsieh, Cheng Hao Wu and Yu Chin Cheng, "A Token-Based User Authentication Mechanism for Data Exchange in RESTful API" , vol. 00, no. , pp. 601-606, 2015, doi:10.1109/NBiS.2015.89
7. Brachmann E., DittmannG., Schubert KD. (2012) “Simplified Authentication and Authorization for RESTful Services" in G. (eds) Service-Oriented and Cloud Computing. ESOCC 2012. Lecture Notes in Computer Science, vol 7592. Springer, Berlin, Heidelberg.
8. Obinna Ethelbert,Faraz Fatemi Moghaddam, Philipp Wieder, Ramin Yahyapour,"A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Application" 2017 IEEE 5th International Conference on Future Internet of Things and Cloud
9. Jones, M.B., Hardt, D.:”The OAuth 2.0 Authorization " October 2012
10. Jit dhulam,"Json Web Token In Django REST API" (article) 2018

AUTHORS PROFILE

Pooja Mahindrakar, was born in Vijayapura, India, She is currently pursuing her **M.Tech** from MIT World Peace University, Pune. She has worked as Google facilitator for 2 years. Published a research paper in deep learning. She is Gate qualified with Interests in: Artificial Intelligence, Machine Learning, Deep Learning, Communication Networks, CCNA, Cyber Security. Network Management.

Dr. Uma R. Pujeri, was born in Sangli, India, in 1981. She has received **M.Tech** degree from PSG Tech college of Engineering Coimbatore in 2008. She has received doctorate degree from Anna University Chennai in May 2017. Her research area is computer network congestion control algorithm. She has worked as a Assistant Professor in Adithya College of Engineering Coimbatore for six years. Currently she is working as a Associate Professor in MIT College Of Engineering Pune Maharashtra. She has total 12 years of teaching experience. She is a Life Member of the Indian Society for Technical Education (ISTE). She has total 20 publications in International journal.