Security Implications for Json web Token Used in MERN Stack for Developing E-Commerce Web Application

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 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 research that follows implementation of authentication and authorization technique using JSON web token which will make web service a role based one .In the project under taken, Json web token is generated in a more secured way by choosing the secret key for web token wisely. Usually key for the token was a mere string or the set of keys stored in a key ring in the database and used alternately for the users to create the token. Or one more trial model is created where captcha was used in short a random number was generated and used as secret key for token generation but the main issue was increased storage. Thus storage is tried to reduce also less predictive secret key is generated in this project.*

***Keywords**: Token, Authentication, JWT, Security, Privacy, Sessions, Encryption.*

I. INTRODUCTION

In the coming years, technology has evolved immensely. As the world is becoming more and more digitally active, there is great need of Internet. And this digital world is evolving drastically, internet is playing a vital role. As a result online websites are increasing more and more ranging from shopping cart, e-commerce to many educational sites. Whatever the website is, security plays a main role in maintaining any website. Security of a website has many features either it may include security against stealing the customers information, hijacking the session, protecting the database, secure login, secure sign up, token generation , token expiry, token key storage etc. In old days when internet was just introduced, many web-sites were launched which had static data and not dynamic changing data. Thus storage and retrieval was bit easier compared to the ones with changing data. AS the dynamic data started increasing, there was a need for more developed website like application where data which is rapidly changing could be handled efficiently. Thus there was an emergence of Web applications. These web applications usually do not allow client to directly connect to the database as there may be many security concerns. Thus there usually is a middle-ware hearing from the client and getting information from the database. Consider huge data websites like amazon,flipkart, Alibaba etc where

huge customer interactions are involved per second, in such websites securing each transactions is again a major concern in web security. As huge financial transactions are involved, attackers attraction over these websites is an obvious thing. Thus securing the play a crucial role in web security. Now let us look into deeper what kind of security involved in such scenarios. Security includes authentication, authorization, data privacy, data integrity, data confidentiality. All these things involve privacy, validation and storage of client's sensitive data. Two major concerns of security are authentication and authorization. What is authentication and authorization and difference between them is a main task to be done before diving details into the web security and data privacy. When a person initially sign ups and logins he enters all his essential information like email, password etc in order to login. At the back-end these information are gathered and validated as to this is the same person who has signed up. Once this validation is done, we say authentication is complete and now the person is eligible to login to the website in short now hw has right to enter into the website and get web site information from the database. Now comes the main task as to what part of database or how much of the data access should that person be allowed to access. Deciding how much, what data and which data part of the database can a person access is called authorization. Main part to note is that authorization is done only after authentication. No authorization without authentication because only if the person is validated, we can think of assigning right to restricted part of database. In order to liable the authorization to any person visiting the website, we have to first assign a ticket to a person after authentication denoting that the person is already authentication now he can get authorization. These tickets are issued by the server at the back-end. these tickets should be carried by every user every time they want to access the database. This ticket was once called "token". At the very beginning, token very popular were the session token. So what is meant by session is the thought. Session is that duration of time the user stays after logging into the website until he leaves off the website. Based on this name session token were created.

Revised Manuscript Received on September 15, 2020.
* Correspondence Author

Pooja Mahindrakar*, M.Tech, MIT World Peace University, Pune (Maharashtra), India.
Dr. Uma R Pujeri, Associate Professor, MIT College of Engineering, Pune (Maharashtra), India.

These tokens are valid only during the session is active. Once the session is over the session token expires. Every time the session is started, the session tokens are created. But a major drawback is that if a user who already visited our website, his details are not stored in the token each time authentication must be done and to do so, database should be accessed each time so the database hit must occur which is a major overhead.In order to avoid this overhead a different set of tokens are developed which itself stores all the user information required for authentication and authorization. Which means they are self contained token and no need to go to database and fetch customer data just to authenticate and authorize. Instead the self contained token having information such as username password expiry date or time of the token etc should be attached with the each database request raised by the user, Thus authorization which is done during every click of the user, can be made easy and the database access overhead can be reduced. One of such self contained tokens are the JSON web token commonly called JWT.Json web tokens are called lightweight tokens as the time taken to parse these is very less as they are self contained. Json web tokens ease the task of authentica-tion, authorization and security of the website. The statements of JWTs are stored in Json format as Json entities and each of these Json entity is used as the payload or plain-text for Josn web encryption or payload for Json web signatures which helps us make the digital signature to claim that the person is the same one as he calims to be. Json web token is never encrypted it is only encoded. As discussed above, JWT are stateless (Self-contained),short-lived Tokens . Initially, token was only a string, E.g 2pWS6RQZpE0T4I0pOX. Now a days it is a big one with encoded in Base64 (UTF-8) format. The burden of the database are reduced by using these light weight tokens.

II. JWT STRUCTURE[1]

Json Web Token is divided into three parts, each one separated by full stop. Following discussion states the same.

A. Header

Header: This contains information about various algorithms used to create signature. Here we consider an example where JWT uses HMAC256 algorithm. The JSON Token is always encoded with base64 algorithm

The header format of JSON is as follows:

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

B. Payload

Payload: This comprises of access right to the token, client ID, who gives the token, the expiry date of the token.

The JSON format for payload is as follows:

```
{
"sub":"0987654321",
"name": "sarang",
"iat": 8550022722
}
```

1) Signature: Signature: This is created by joining header and en-coding them using base64 algorithm, later it uses HMAC-256 algorithm to encrypt the data.

The pseudo-code of the token creating the signature is as follow:

```
HMACSHA256(base64Url, Encode(header)
+ "." +base64Url, Encode(payload),
) secret base64 encoded
```

C. Authorization Bearer

An arbitrary string is called bearer token which can be used for getting permissions . A bearer token can be JWT token which can be used for authorization. A Bearer token is only an arbitrary string, used for permission. Bearer token can be jwt when jwt is used for authorization.

Eg: eyJhbGvbdi9fcbjderu7crbeufy9Rt709u9. ey3ORT4tfvbR843Rfh6t8Tgbj578EnjyTnhyufu. SftykmR843Rfh6t8bhj3DY4ugbmgjhvhy46t87FD

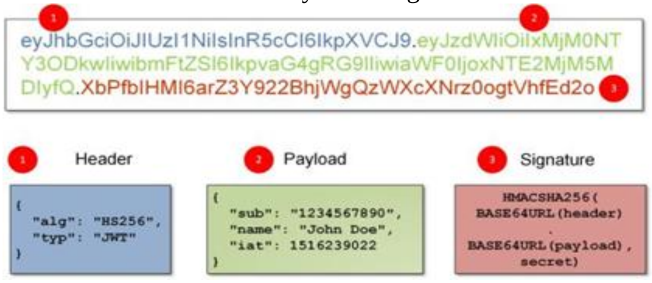
III. TYPES OF SIGNATURES:

Symmetric Signature : Symmetric signatures are the ones which rely on same secret key for verifying and gen-erating signatures using an HMAC function . Symmetric signatures are setup friendly and mostly used within a single application

Asymmetric Signature Asymmetric signatures depend on a key pair for signing and verification. The public key is publicly available and is used for verification and private key is kept secret and used for signing. Asymmetric signatures are useful in distributed scenarios.

Header + Payload

Header + Payload + signature



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o
```

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

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

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

JWT PSEUDOCODE

$$\textbf{Token} = f(Base64Encode)\sum_{n=\alpha,\beta}^{\infty}(header.payload.signature)$$

Step 1: Encoding the token type(JWT) and algorithm used for JWT generation (here HMAC SHA 256) with base64encode. Forms first part of the token.

Step 2: Encoding the payload details with base64encode. Payload contains user details .Forms second part of the token.

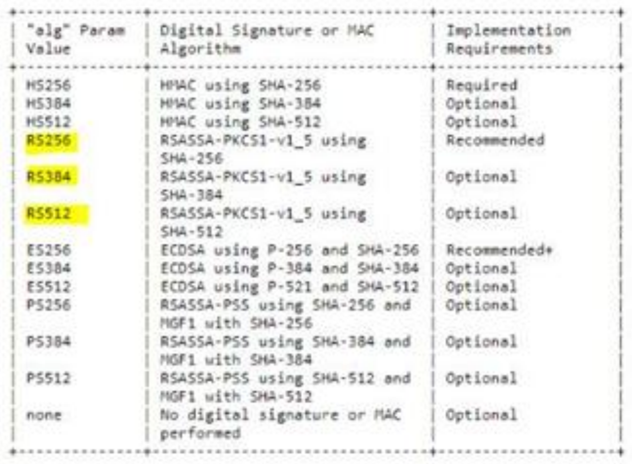
Step 3: Creating the hash of the header and the payload. This uses HMAC algorithm.

Step 4: Now the hash created is again hashed with SHA 256 and the secret key to form the final signature part which is encoded with base64.Forms third part of the token.

Step 5: All the three encoded parts are concatenated and separated with dots. A minimum od 304 byte long JWT is generated.

IV. CRYPTOGRAPHY ALGORITHMS USED BY JSON WEB TOKENS :

JWT token are never encrypted they are only encoded in base 64 format. The reason is JWT contains all the details of the user used for authentication and authorization also we do not want to expose the details in the token as easily to every user thus they have added encode part so that the decoding of the user contents in the token can be made easy. When we decode the tokens, we decode it in three parts header, payload and signature separately but we cannot display the details of the signature part as the secret key is missing. The one who knows the secret key can only manipulate the token details but interested ones can look into details of the customer. JWT tokens as mentioned above have 3 parts, header, payload and signature also the formats are mentioned above. Now we need to know more about how the algorithms used. Look at the figure to know the types of algorithms used along with the details.

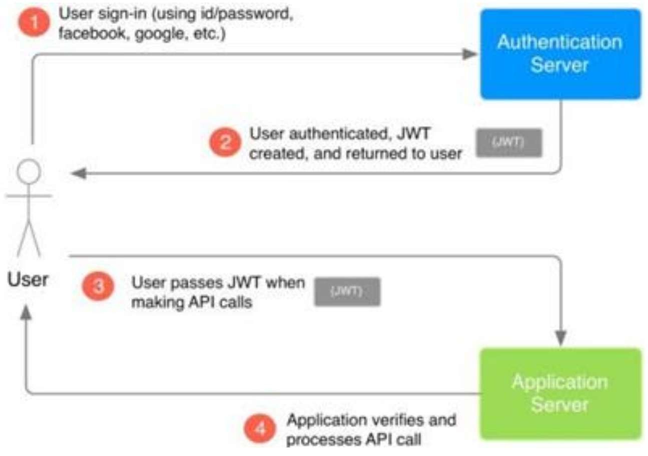


"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

JWT token contains various encryption algorithms used in combinations .They use hash functions or hash algorithms , symmetric algorithm and asymmetric algorithms all in com-binations with each other. JWT token use two algorithm in a combination because they apply one algorithm usually hash algorithm to keep header and payload data persistent away from data tampering later along with the encryption key which is the secret key, these hashed parts are encrypted and final signature is generated. Thus JWT uses two algorithms in combination it might be hash+symmetric algorithm eg HS256 which uses HMAC hash algorithm to hash the header and payload part later it uses SHA-256 to hash the final one bundle with the secret key used for digital signature. This way key are used for encrypting and creating the digital signatures.

V. JWT WORK-FLOW

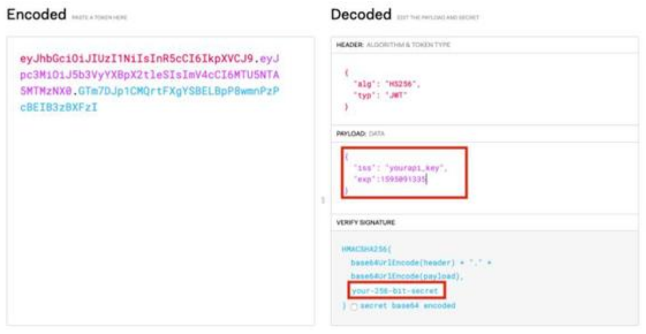
- User initially sign in or raises a login request to the server or middle-ware.
- A secret key in the form of private key is used to sign a token and JWT is generated and sent to the user.
- Jwt created should also be stored somewhere so that for each login, the user can append the request with the token .So storage varies from cookie storage to the local storage.
- suppose that the JWT json web token is stored in the local storage.
- when a client want to raise a request to access some data from the database using any HTTP request or basically an API, it has to append the token as bearer token and send to the server.
- The server identifies the bearer token and validates the user, if found correct, it checks the permissions and if permissions are found correct, it replies back the data to the user.


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

VI. WORKING OF JWT

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,signup 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 retuned to the user.

VII. JWT.IO



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pc3MiOiJ5b3VyYXBpX2tleSIsImV4cCI6MTU5NTA
5MTMzNX0.GTm7DJp1CMQrtFXgYSBELBpPBwmnPzP
cBEIB3zBXFzI
```

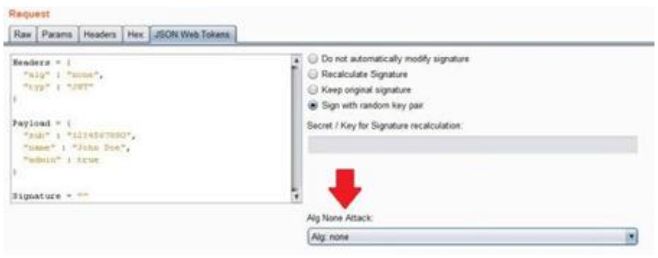
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

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

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

JWT.io is a website which helps us decode the JWT token. It helps us look at the header and payload part of a jwt token .Where as the signature part of JWT token is not disclosed. A secret key is needed to know the signature and alter the contents of JWT token. Signature means digital signature here.JWT.io website helps us check whether the token taken is JWT or no also it checks if the token is expired or alive. JWT.io merely takes tye token, divides it into header, payload and signtature by considering the full stops as the split part. It then takes each one of split part and decodes using Base64 decoder and displays the result in the right side of the website each in respective boxes. The signature part is left unrecognized and demands a secret key which is the main security feature of JWT token.

VIII. ATTACKING JWT



```
Headers = {
  "alg" : "none",
  "typ" : "JWT"
}

Payload = {
  "sub" : "1234567890",
  "name" : "John Doe",
  "admin" : true
}

Signature = ""
```

- ( ) Do not automatically modify signature
- ( ) Recalculate Signature
- ( ) Keep original signature
- (●) Sign with random key pair

Secret / Key for Signature recalculation:

Alg None Attack: Alg: none

As known , Json web tokens have a field called algorithm where the algorithm used by the token is specified usually this is present in the header part of the token

• Now we have a great vulnerability that by mentioning the algorithm that way helps the hackers to identify which algorithm combinations are used to generate this token.

• As a note there is an algorithm parameter which is allowed and that is using ”none”.

• In such cases, Json web tokens does not use any algorithms and mentioning it publicly helps the hackers to easily hack the token.

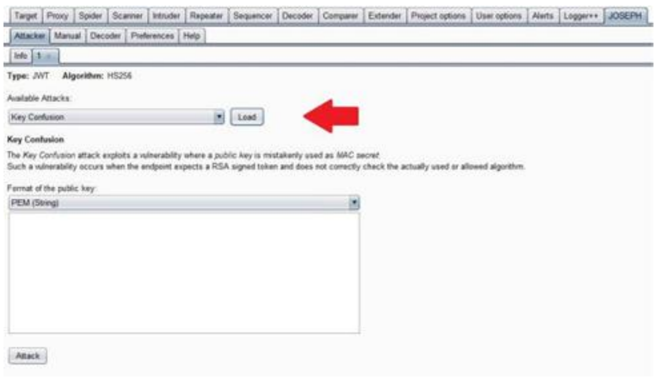
• Because if the token uses no algorithms that means digital signatures are not created thus the signature part does not preserve the data and data tampering can be done.

• Using burp suite we can do this hack easily.More precisely, we have ”json-web-token-attacker” burp suite extension using which we can make this attack on the fly .We do have another hack method which occurs if the JWT token use RSA algorithm.

• If the JWT token is using RSA it means it is using asymmetric keys which are public key and private key. • In such RSA usages in JWT, the private key is used to sign the token and public key is used for validating the token

• Again using the burp suite, we can change the alg part of the header from RS256 to HS256 which means we are degrading the algorithm from asymmetric key to symmetric key and we use the public key as symmetric key.

• Thus the attack on the RSA possessed JWT can be made successfully.



Type: JWT Algorithm: HS256

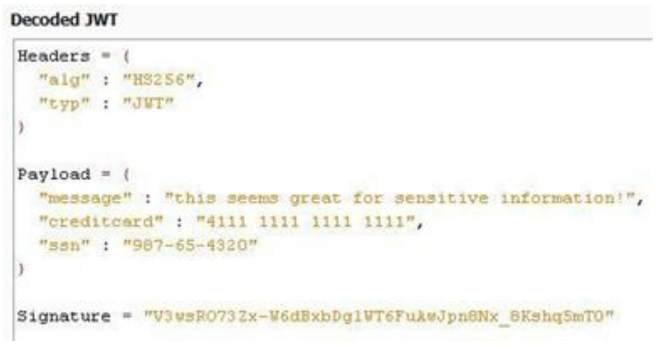
Available Attacks: Key Confusion

Key Confusion

The *Key Confusion* attack exploits a vulnerability where a *public key* is mistakenly used as *MAC secret*.
Such a vulnerability occurs when the endpoint expects a RSA signed token and does not correctly check the actually used or allowed algorithm.

Format of the public key: PEM (String)

Again using the burp suite, we can change the alg part of the header from RS256 to HS256 which means we are degrading the algorithm from asymmetric key to symmetric key and we use the public key as symmetric key. Thus the attack on the RSA possessed JWT can be made successfully.



Decoded JWT

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

Payload = {
  "message" : "this seems great for sensitive information!",
  "creditcard" : "4111 1111 1111 1111",
  "ssn" : "987-65-4320"
}

Signature = "V3wsRO73Zx-W6dBxbDg1WT6FukwJpn8Nx_8KshqSmTO"
```

IX. MERN STACK a) Introduction to MERN Stack: Now a days developers are putting efforts to create a user friendly application and enhance their experience by providing the products under strict timeline. To achieve this , stacks can be used to build application in restricted time period . One among such stacks is a JavaScript stack (which in blooming now-a-days) called MERN stack. MERN helps developers to build efficient web applications in short duration by just mastering java script. A great advantage of JavaScript stack is easy integration and efficient testing. With the growing online demands, demand for web applications has increased tremendously. Websites which earlier would be the combination of HTML, CSS, php or complex JavaScript , now no longer suffice for the current demands. Websites are moved to web applications where highly dynamic data is involved. With increase in demand for dynamic data, demand for great user experience has also increased. Thus there is pressure among developers for delivering new web applications smarter, faster and efficient. Even after development enhancing the experience has become tedious task for developers. Thus to ensure the web applications are highly efficient and scaled appropriately, developers now a days are adopting a set of technologies to make things possible. These set of technologies is collectively called stack.On the urge of the current user demand, software engineers are using stack based web development in which they develop web applications based on pre exixsting frameworks(like javascript framework.

Two popular frameworks are evolved from javascript and mostly in demand viz MEAN and MERN. These both stacks are made from open source components and provide end to end framework for building dynamic comprehensive web application that allow browser to attach with database. Two vital advantages of using stack are : • Confusion in coding can be avoided by just coding in one language. • Flexibility can be evolved in all the web applications developed using stack. • Mongodb:

Mongodb referred as NoSQL, is an open sourced databased source. Unlike SQL, Mongodb is a document collection oriented databse where rows column and table kind of notations are eradicated. It uses JSON as documents in combination with schemas. Enables the use of JSON as documents in conjunction with schemas. Over the past few years, there has been a need for unrelated data due to the huge increase in volume and variety of data.

• Express

Express.js is a web application framework that is a small and flexible . Express helps and eases the task of creating APIs due to middleware access. Middleware Functions are ones that have application access and responsiveness also the following function. As spring, Express also provides an easy to use web application.

• React

React was emerged out of Facebook system , in the Facebook ad agency. Initially, developers in facebook used the standard MVC client model to start with but it had all the data for both templates and bindings.

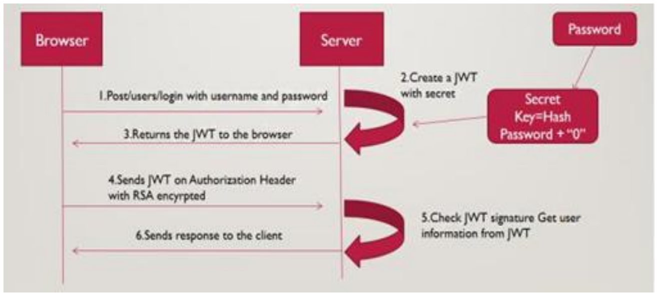
Views are the ones which respond to the changes in the models just by resizing them. AS the complexity increased, the app came with the new change. As there will be subtle differences in the update refresh code,

depending on the cause of the update, Cascading updates are difficult to maintain,.

This is where they began to think of building something descriptive instead of important [5].

• Node.js

Node.js is an event-driven JavaScript implementation which is designed to implement scalable network applications. Node.js uses an event-driven asynchronous model [2], which does not preclude an I / O model that makes it efficient, simple, and highly efficient [3]. X. PROPOSED MODEL The below figure describes the steps mentioned above so that a secured authentication based access control is provided for an e-Commerce website In the proposed model, consider the browser and server of Web Application. • Browser Displays Login credentials during sign-up through an HTTP post request to the Web Server.


Browser
Server
Password
1.Post/users/login with username and password
2.Create a JWT with secret
Secret Key=Hash Password + "0"
3.Returns the JWT to the browser
4.Sends JWT on Authorization Header with RSA encyrpted
5.Check JWT signature Get user information from JWT
6.Sends response to the client

• At the server side, Secret key is generated using hash password and a string zero or ASCII value "48".

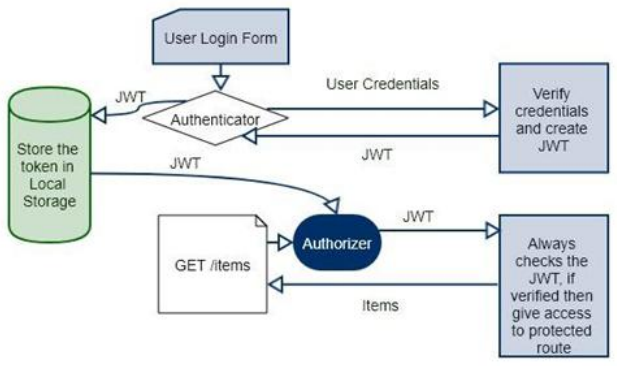
• At the Server JSON web token is generated considering the secret key.

• The generated token is sent to the browser for future requests.

• Now on, Along each HTTP request, in HTTP header, JSON web token is sent to authorize and authenticate the user by the browser

• Server considers the token and validates if the user is authorised to access the data requested

. • If authentication and access is permitted, Browser gets the intended data as response through HTTP.


User Login Form
User Credentials
JWT
Authenticator
Verify credentials and create JWT
JWT
Store the token in Local Storage
JWT
Authorizer
JWT
GET /items
Always checks the JWT, if verified then give access to protected route
Items

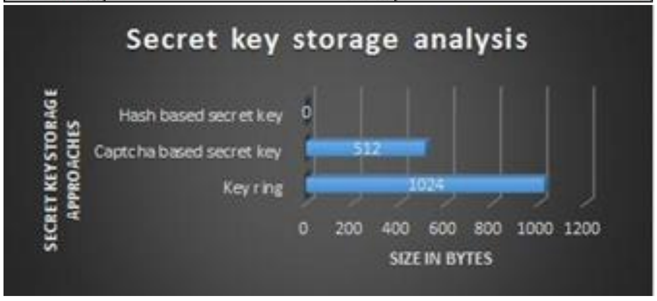
IX. COMPARISON AND ANALYSIS

The trial model was generating the secret key used for Json web token in a different way. The trial model had a login with captcha. The captcha used was v3 of google captcha which has a captcha random key generated. This key generated was used as the secret key to generate the token. The overhead here was that the token used the secret which was the captcha key but this captcha key should also be remembered . To do so the captcha need to be stored in the database. But plain captcha key storage would invite the database injection attack and the stored captcha at some time can be stolen. To avoid this we must store the captcha in hash format. This hash format storage is again a overhead. As we cannot decode the hashed value we must used the hashed captcha value as the secret key which again storage is an issue. Thus a storage overhead of over 256 bytes is created in the database.

We also consider a scenario where a key ring is used con-sisting of a set of secret keys stored in the database. The main disadvantage is that if the entire key ring is compromised, All the tokens can be attacked and tampered in simple. Also approx minimum 1024 bytes data should be stored in the database which is again a overhead of storage. Here mere storage of plain keys is again a vulnerability introduced. In the proposed model where we are using the secret key as the hashed password + string "0". However the password must be stored in the database in hashed format.

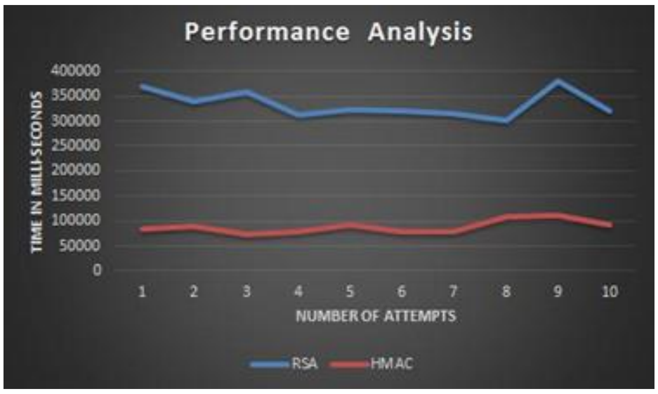
Utilizing its storage and security ,we can create a secured secret key by just concatenating and string of your choice thus the storage is also reduced and no key storage is required. The analysis and comparisons are given in the following graphs. The following graphs are the analysis from the test cases:

Key ring	Captcha based secret key	Hash based secret key
1024	512	0



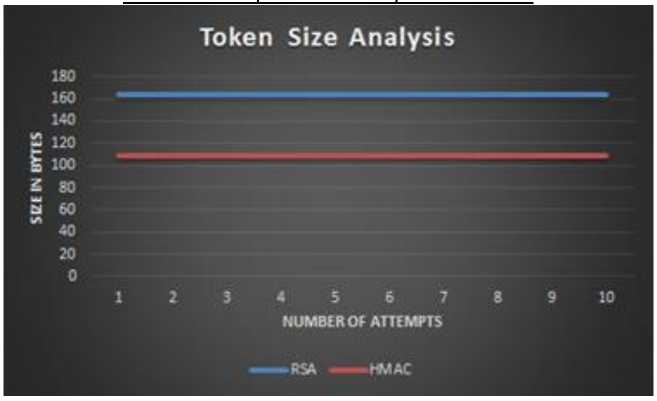
In the graph 11 and 12 ,two tokens are considered one is Json web token using RS256 algorithm which uses asymmetric keys and also used encryption standard RSA other one is Json web token using HS256 which uses no encryption but mere hashing algorithms. These two types of tokens are analysed based on the storage taken to store the secret keys of the tokens so that each time tokens are analyzed, hit to databases increases.

	RSA	HMAC
1	368412	83660
2	339241	91155
3	359306	74678
4	311298	77904
5	322892	100363
6	319571	78579
7	315724	79581
8	301652	138004
9	379141	110295
10	320035	93199



In the graph 13 and 14 two tokens are considered one is Json web token using RS256 algorithm which uses asymmetric keys and also used encryption standard RSA other one is Json web token using HS256 which uses no encryption but mere hashing algorithms. These two types of tokens are analysed on time bases as to how much response time of token are taken and are noted in the table and graph is drawn accordingly.

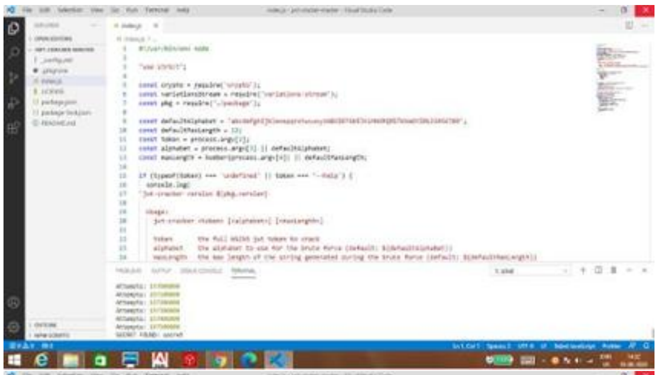
	RSA	HMAC
1	164	109
2	164	109
3	164	109
4	164	109
5	164	109
6	164	109
7	164	109
8	164	109
9	164	109
10	164	109



In fig 15 and 16Here Two type of tokens again one is Json web token with RS256 and other is Json web token with HS256 is considered. For reference a secret used here is ”secret” and token sizes are noted in the tabular format. Graph is drawn accordingly

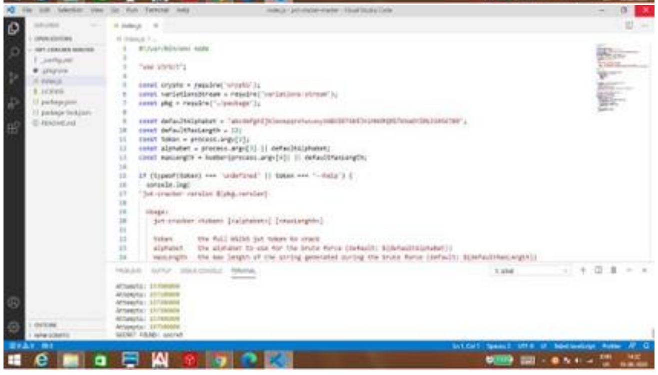
X. JWT PENETRATION TESTING TEST 1:

JWT secret cracker: As we have gone through the project we seem to secure the secret key used for JWT token generation. The same is validated using a tool called JWT cracker. Jwt cracker is a windows based tool used to crack secret key of the jet tokens. For a plain text secret used it takes nearly 10 minuted to 30 minutes to decode the secret used in json web token generation. Thus taking into consideration the token secret security, A sample token was generated using plain text secret called ”secret” and this generated token was taken for testing in jwt cracker . The secret was appropriately found by the tool please check the figure 7 for this demo screenshot. Now the proposed model genrated a secured secret key for token generation. This was also given as input to JWT cracker with which after waiting for four and half hours the result. obtained was no solutions found. Check the figures 8 and 9 for the result Thus after this penetration testing we can say that the JWT generated is having a less predictive secret key used for token generation. The hack was conducted on windows 8.1 with intel processor core i5 with 4gb ram specifications.



```
#!/usr/bin/env node

'use strict';

const crypto = require('crypto');
const variationsStream = require('variations-stream');
const pkg = require('./package');

const defaultAlphabet = '[illegible]';
const defaultMaxLength = 12;
const token = process.argv[2];
const alphabet = process.argv[3] || defaultAlphabet;
const maxLength = Number(process.argv[4]) || defaultMaxLength;

if (typeof token === 'undefined' || token === '--help') {
  console.log(
`jwt-cracker version ${pkg.version}

  Usage:
    jwt-cracker <token> [<alphabet>] [<maxLength>]

    token       the full HS256 jwt token to crack
    alphabet    the alphabet to use for the brute force (default: ${defaultAlphabet})
    maxLength   the max length of the string generated during the brute force (default: ${defaultMaxLength})
```

```
Attempts: [illegible]
Attempts: [illegible]
Attempts: [illegible]
Attempts: [illegible]
Attempts: [illegible]
Attempts: [illegible]
SECRET FOUND: secret
```



XI. 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

XII. CONCLUSION AND FUTURE WORK

Json web token used in this project considers all the security issues and tries to reduce the attacks caused on json web Fig. 20. JWT hack on windows on a token bearing secret as hashed password tokens. The vital part is the secret key of the token. Generation of secret key and usage of the key appropriately in rh token generation is done successfully by concatenating a string to the hashed user password. It is proved that the encryption involved are zero and the storage of the keys is completely reduced by smartly using security feature.

Thus we can conclude that this way of using secret key is easy and appropriate for any web applications involving json web tokens. In the future work involves rigorous pen test on the token generated so as to find out loop hole in the proposed system. The implementation can also be tried using many other

REFERENCES

1. Muhamad Haekal, Eliyane ,"Token based authentication using Json webtoken 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) ServiceOriented 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.