Web Application Security using spring with multiple approaches and their merits and demerits

Let’s discuss something Web application security which is hugely using in every small scale and large scale industry

i will try to cover about main difference between Spring Rest security , JWT (Json Web Token) and Oauth2 and why one after another security approaches coming to market

1. **Spring Security:**

75% people are using this approach to apply security using spring rest Role base security Authentication and Authorization, and this way is more secure

what internal process going on by server and client let's go bit depth

-> when i first time try to login with valid credential then in server side there will be one CSRF(Cross-site request forgery) token will be created by server itself and based on implementation it will be stored for further used

->so next in every request server will compare with generated token with existing one if it matched then autorized success else some message?

-> so in middle some enhancement came, why my server will generate token in every request why not we will store somewhere that CSRF token until the user accessing my application that's why they came up with approach let's bind that CSRF token either in session or in Cookies and when user will do log out then session will be expiry automatically so based on requirement they specify session time out in web.xml

**Problem:**

Assume i logged in from desktop and didn't logout and opened same application from different device then in both of the case my session is active that's why its not expiary and Storage requirements grow as more users have active sessions and it's not possible to handel in single server so it will kill the scalability of our application

So to overcome this Storage issue JWT introduced.

**2.JWT (Json Web Token):**

JWT Token. This token is self-descriptive, it contains all necessary information about the token itself , like this token is segmented by 3 catagory

*1.Header(ALGORITHM & TOKEN TYPE):*

->Where Header contains your hashing algorithim information like

{

"alg": "HS256",

"typ": "JWT"

}

here HS256 and RS256 is alogorithim like jwt will takes care to convert our raw data to encode format using above menton hashing technique

*2.Payload Data:*

->where Payload contains your username , password and user roles like

{

"sub": "8867992920",

"name": "Basanta Hota",

"admin": true

}

*3.VERIFY SIGNATURE:*

->where it contains your encrypt and decrypt format based on your hashing algorithim which you provide in first segement

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

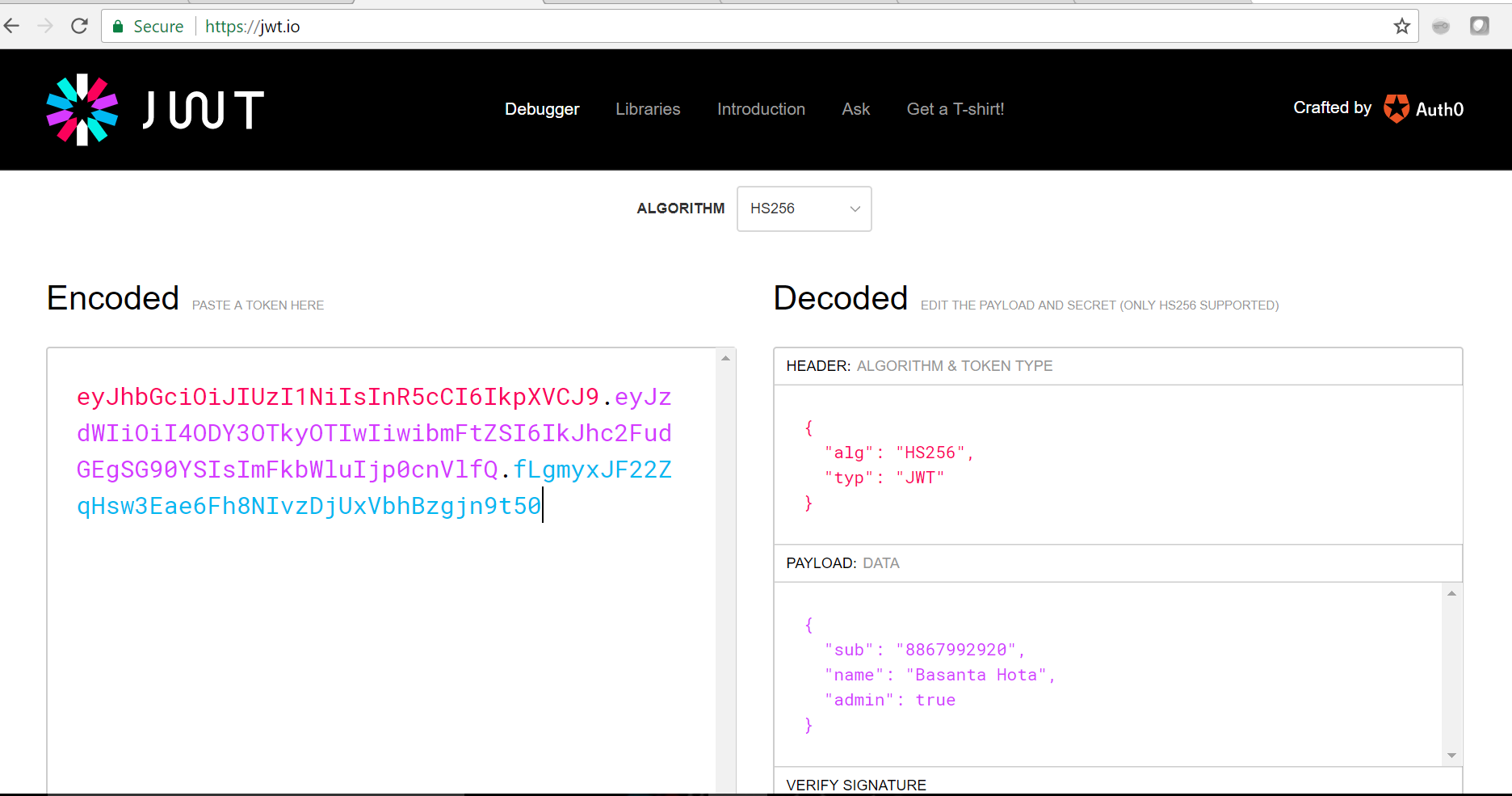
secret

)

So user cannot change for example expiration date or any other claim, because this token is generated (signed) by the server with secret keyword like below as per i mentioned above in 3 segements

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI4ODY3OTkyOTIwIiwibmFtZSI6IkJhc2FudGEgSG90YSIsImFkbWluIjp0cnVlfQ.fLgmyxJF22ZqHsw3Eae6Fh8NIvzDjUxVbhBzgjn9t50

You can check in https://jwt.io/ portal how to encode and decode see the below screen shot



In this case the data is stored in a JWT that is passed around instead of on the server.so no server side storage issue

**Problems:**

• The JWT size could be larger than a session ID. It could affect network performance since it is included with each HTTP request.

• The data stored in the JWT is readable by the client. This may be an issue.cause as it is standard encryption format so hackers can break our security bridge

3. **Oauth2:**

Already we have 2 other approaches to implement Security features so we can't say Oauth2 is resolving above mention issue like storage or hacking issue and Oauth2 is not any mechanism to implement security in new way .we can apply security by Oauth2 also.

Oauth2 is just a framework who plays the main roles between your client and Server so we can say Oauth2 is one of the middle component to talk client and server

The main purpose to introduced Oauth2 is that it is a framework who support us delegate accessing

Example: Suppose assume any third party want to access your application so He must have to authorized before access and as a owner you don't want to give full authority to him for your application you want to delegate authority for partial features

***Let’s discuss one real world example***

Assume i developed one application where i added a features to sign up my application using any social media like Facebook, twitter and Google account so to develop such kind of features i should go for Oauth2 implementation

suppose i clicked sign up through my Facebook account then request will go to Facebook server with client request id then Facebook gives you popup hey Basant this application want to communicate with you do you want to give authorized him to access for these these things ?

Once i clicked OK. Then as Facebook is delegating authority access to your application , Facebook will send one access token to your application for specific time interval

So like this client server and third party (Facebook) will communicate with each other for delegate authority access

**Problems:**

• You depend on the third party in order for your users to use your service. If their service goes down or they discontinue it then you need to figure something else out. Eg: how do you migrate the user's account data if their identity changes from "foo@a.com" to "bar@b.com"?

• Usually you have to write code for each provider. eg Google, Facebook, Twitter.