Week 4 Lesson 8 Readings

***Getting Started with JSON Web Tokens (JWT)***

* **Compact**: Because of its size, it can be sent through an URL, POST parameter, or inside an HTTP header. Additionally, due to its size its transmission is fast.
* **Self-contained**: The payload contains all the required information about the user, to avoid querying the database more than once.

These are some scenarios where JSON Web Tokens are useful:

* **Authentication**: This is the typical scenario for using JWT, once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used among systems of different domains.
* **Information Exchange**: JWTs are a good way of securely transmitting information between parties, because as they can be signed, for example using a public/private key pair, you can be sure that the sender is who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn’t changed.

JWT consist of three parts separated by dots = Header, Payload, and Signature

Header:

{

'alg': 'HS256',

'typ': 'JWT'

}

--------------------------------------------------------------------

Payload:

Reserved claims: These are a set of predefined claims, which are not mandatory but recommended, thought to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject), aud (audience), among others

* **Public claims**: These can be defined at will by those using JWTs. But to avoid collisions they should be defined in the IANA JSON Web Token Registry or be defined as a URI that contains a collision resistant namespace.
* **Private claims**: These are the custom claims created to share information between parties that agree on using them.

{

'sub': '1234567890',

'name': 'John Doe',

'admin': true

}

----------------------------------------------------------

Signature:

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

HMACSHA256(

base64UrlEncode(header) + '.' +

base64UrlEncode(payload),

secret)

***Develop, Debug, Learn?***

HTML was always known as a pass-through system since no matter what you had to do.

JavaScript was seen as a big gain since they were able to understand what was going on in the backend side of it.

We are here to help and support the end users.