**JWT(Java Web Token)**

1. JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.
2. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.
3. Although JWTs can be encrypted to also provide secrecy between parties, we will focus on signed tokens.
4. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties.
5. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.
6. JSON web tokens can be used for authorization, information exchange

🡪Authorization : This is the most common 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 across different domains.

🡪Information Exchange : JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are 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 been tampered with.

1. In its compact form, JSON web tokens consists of three parts seprated by dots(.), which are :
   1. Header
   2. Payload
   3. Signature

🡪Therefore the JWT typically looks like the following :

xxxxx.yyyyy.zzzzz

1. The output is three Base64-URL strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact when compared to XML-based standards such as SAML.

🡺JSON Web Tokens

1. In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required.
2. You also [should not store sensitive session data in browser storage due to lack of security](https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage).
3. Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the **Authorization** header using the **Bearer** schema. The content of the header should look like the following:
4. This can be, in certain cases, a stateless authorization mechanism. The server's protected routes will check for a valid JWT in the Authorization header, and if it's present, the user will be allowed to access protected resources. If the JWT contains the necessary data, the need to query the database for certain operations may be reduced, though this may not always be the case.
5. Note that if you send JWT tokens through HTTP headers, you should try to prevent them from getting too big. Some servers don't accept more than 8 KB in headers. If you are trying to embed too much information in a JWT token, like by including all the user's permissions, you may need an alternative solution, like [Auth0 Fine-Grained Authorization](https://fga.dev/).
6. By the help of the jwt.verify() method we can verify the token and the user