



# JSON Web Token (JWT)

## What is a JWT?

- They contain JSON objects which have the information that needs to be shared between client and server.
- These tokens are then sent on every HTTP request, which allows the server to authenticate the user.
- Each JWT is also signed using cryptography (hashing) to ensure that the JSON contents (also known as JWT claims) cannot be altered by the client or a malicious party.

## Example

For example, when you sign in with Google, Google issues a JWT which contains the following claims / JSON payload to know exactly who the end-user is.

```
{  
  "iss": "https://accounts.google.com",  
  "azp": "1234987819200.apps.googleusercontent.com",  
  "aud": "1234987819200.apps.googleusercontent.com",  
  "sub": "10769150350006150715113082367",  
  "at_hash": "HK6E_P6Dh8Y93mRNtsDB1Q",  
  "email": "jsmith@example.com",  
  "email_verified": "true",  
  "iat": 1353601026,  
  "exp": 1353604926,  
  "nonce": "0394852-3190485-2490358",  
  "hd": "example.com"  
}
```

# Structure of a JWT



A "Signature" section, that is the result of Header and Payload, concatenated and then encrypted with the private key.

JWT in the serialized form represents a string of the following format:

```
[header].[payload].[signature]
```

# API Authentication

CLIENT

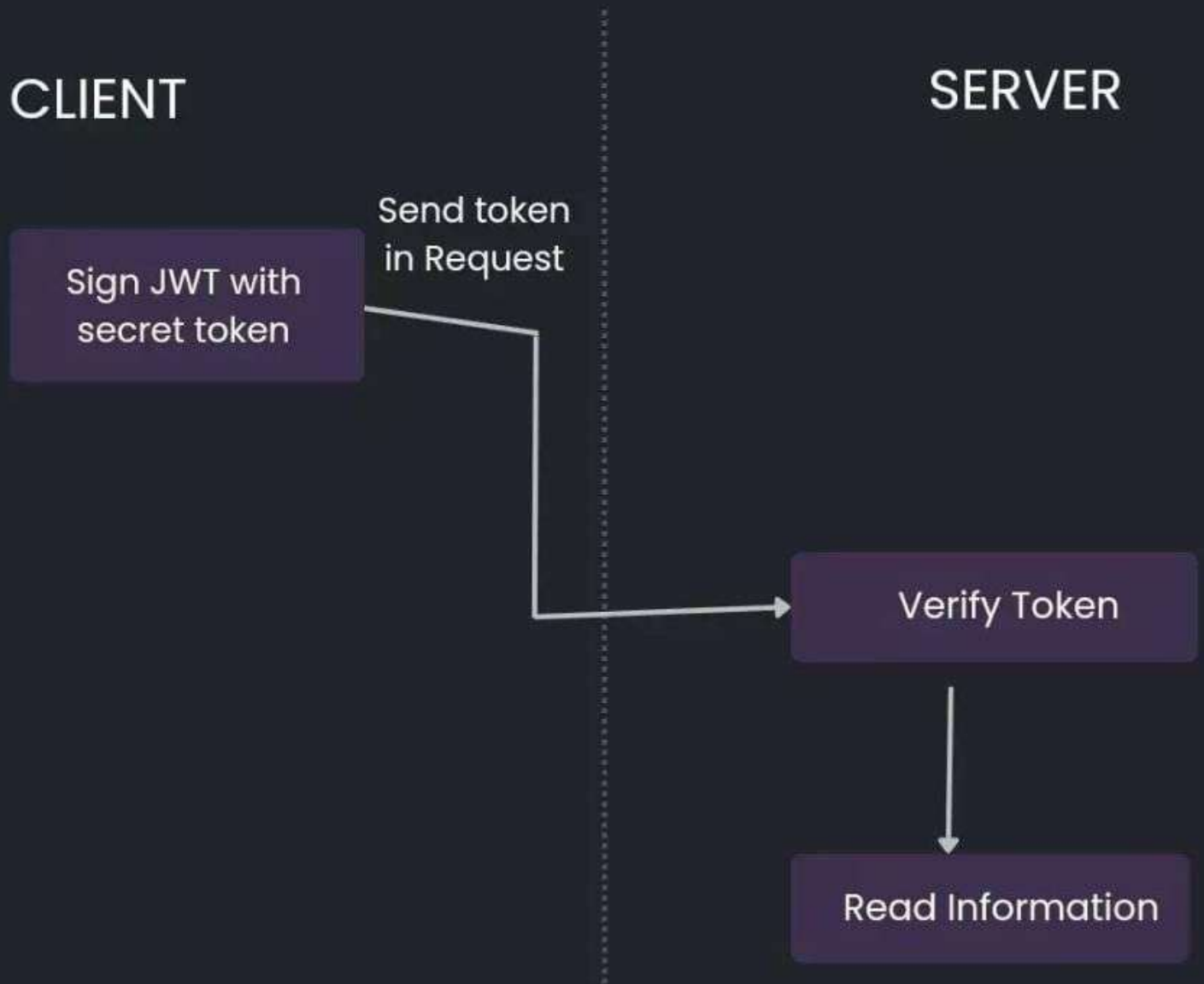
SERVER

Sign JWT with  
secret token

Send token  
in Request

Verify Token

Read Information





## **Token stored in a cookie**

When stored in the browser's cookies, it is possible to set the "HttpOnly" flag (and "Secure"), to get protected against token theft in case of XSS attacks.

## **Drawbacks**

No CSRF protection can be expected from the token. Indeed, the token is automatically sent with the cookies (and therefore with any CSRF attack request).

## **Possible solution**

Store your access token in memory and store your refresh token in the cookie

## **Token stored in the sessionStorage or localStorage of the browser**

Token to be included in every request sent to the server, for instance with a header "Authorization: Bearer <token>".

### **Drawbacks**

since the token must be made available to the JavaScript application, it will be exposed in case of XSS vulnerabilities and might be stolen.