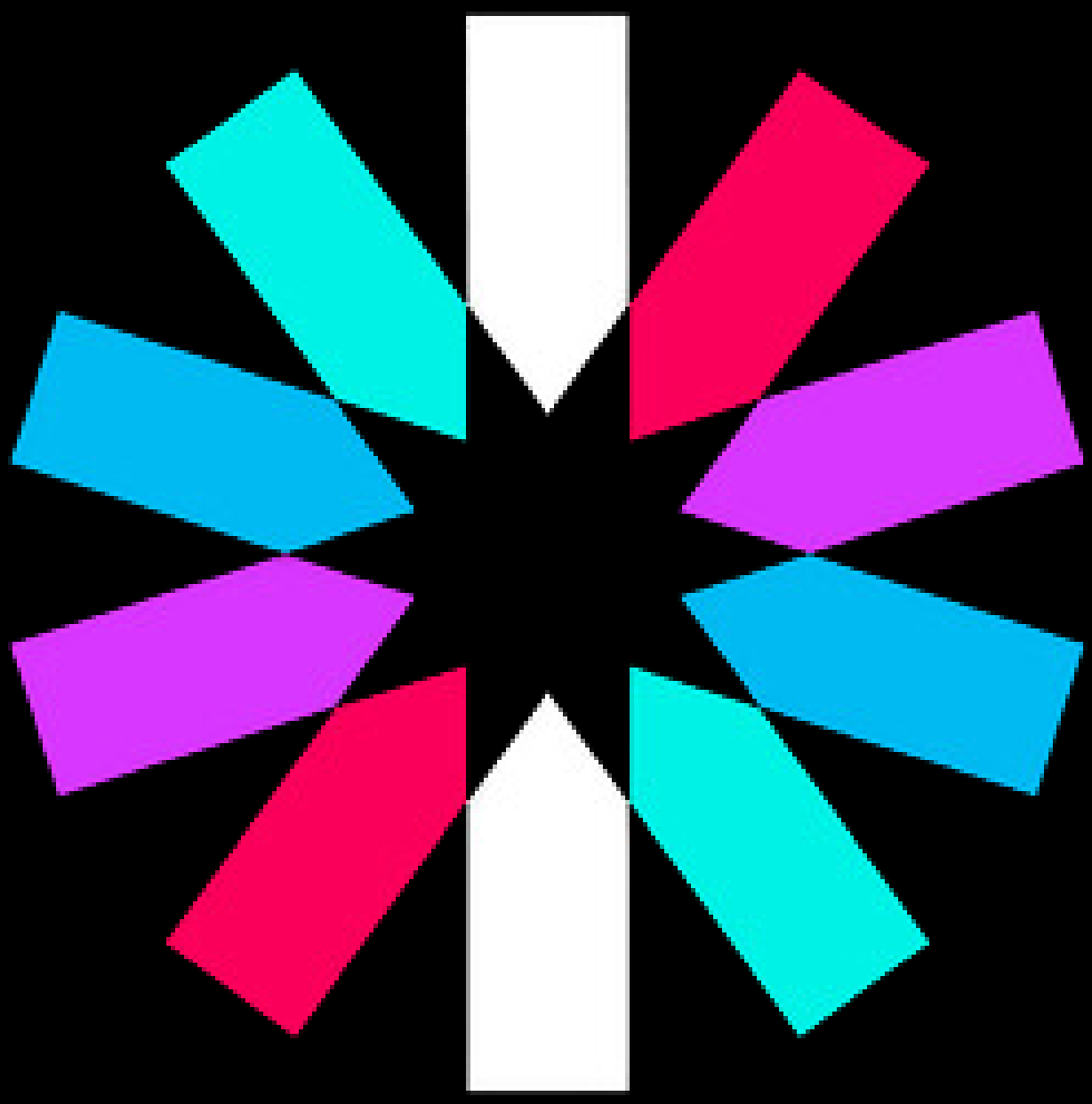
JSON WEB TOKEN

It’s easy



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.

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": "107691503500061507151130823 67",
    "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

HEADER

PAYLOAD

SIGNATURE

Header: ALGORITHM & TOKEN TYPE

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

Payload: DATA

```
{
  "sub":"user1001",
  "iat":"1569302116",
  "role":"admin",
  "user_id":"user1001"
}
```

Verify Signature :

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secretKey
)
```

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 Authetication

CLIENT

SERVER

Send token
in Request

Sign JWT with
secret token

Verity Token

Read
Information

UseCase

On the client-side, tokens can be stored in two different ways :

- stored in a cookie, or
- stored in the sessionStorage (or localStorage) of the browser.

sessionStorage or localStorage

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, il will be exposed in case of XSS vulnerabilities and might be stolen.

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

FOLLOW