JSON Web Tokens

Introduction

Lexicon

NAME	DEFINITION
JWT	JSON Web Token
JWS	JSON Web Signature
JWE	JSON Web Encryption
JWA	JSON Web Algorithm
JWK	JSON Web Key

The bank check analogy

A JWT can be seen as a check, containing a payload (which could be the amount of the check and the party to which you are transferring the money, or anything else) which is signed by yourself with your personal signature.

JWT

What is a JWT ?

A JWT gives you the standard by which you can structure your communication payload between 2 parties.

JWS or JWE

The actual implementation of a JWT is done using the JWS OR JWE.

JWS (JSON Web Signature) ensures payload integrity, meaning it verifies that the contents of the JWT have not been tampered with during transmission.

JWE (JSON Web Encryption) provides encryption, ensuring that the JWT payload remains confidential and cannot be read by unauthorized parties.

JWS

The JWS is composed of a Header, Payload and Signature.
The signature is used to provide integrity and authenticity.

HEADER	Contains metadata about the payload and signature. E.G. The signature algorithm
PAYLOAD	Contains the data to be transferred. The data is represented as attributes, called claims. E.G. iss (the issuer), exo (when the token expires), or any custom...
SIGNATURE	Calculated hash of the JWS header and payload. Providing integrity and authenticity.

JWE

The JWE is composed of a Header, Encrypted Key, an Initialization Vector, Cyphertext and Tag.

Header	Contains metadata about the encryption, such as the encryption algorithm and key management algorithm.
Encrypted KEY (CEK)	The content encryption key (CEK) used to encrypt the payload. This key is encrypted with the recipient's public key or a shared secret.
Initialization Vector (IV)	A random value used in encryption algorithms to ensure that the same plaintext does not encrypt to the same ciphertext.
Cyphertext	The encrypted payload, typically encrypted using the content encryption key and initialization vector.
Tag	A value used for integrity checking to ensure that the ciphertext has not been tampered with.

JWA

JWA is a specification for describing digital signature and encryption algorithms that can be used to create JSON Web Tokens (JWTs).

A JWA object is a JSON object that contains a set of objects, each of which defines a cryptographic algorithm. Each object has the following properties:

alg	The algorithm identifier.
enc	The encryption algorithm identifier (if applicable).
key_id	The identifier of the key to use for signing or encryption (if applicable).
Kty	The key type (if applicable).
use	The usage of the key (if applicable).

JWK

JWK is a JSON data structure that represents a cryptographic key.
It is used in JSON Web Algorithms (JWA) to specify the algorithm and parameters for cryptographic operations such as signing and encryption.

It contains a set of properties that describe the cryptographic key.
The following properties are mandatory:

kty	The key type, such as "RSA" or "EC".
kid	The key ID, which is a unique identifier for the key.

The following properties are optional:

n (RSA or EC)	The modulus of the key, in base64url encoding.
e (RSA)	The public exponent of the key, in base64url encoding.
d (RSA)	The private exponent of the key, in base64url encoding.
p (RSA or EC)	The first prime factor of the modulus, in base64url encoding.
q (RSA or EC)	The second prime factor of the modulus, in base64url encoding.
dp (RSA)	The first CRT coefficient, in base64url encoding.
dq (RSA)	The second CRT coefficient, in base64url encoding.
qi (RSA)	The third CRT coefficient, in base64url encoding.
alg	The algorithm identifier for the key, such as "RS256" or "ES256".
use	The intended use of the key, such as "sig" for signing or "enc" for encryption.

Token lifecycle

The token lifecycle refers to the various stages that a token goes through from its creation to its expiration or invalidation.

Token creation

1. A JWT is created by an authentication server or identity provider when a user successfully logs in or authenticates.

2. The server generates the JWT with relevant claims (such as user ID, roles, and expiration time) and signs it if necessary.

Token Issuance:

3. Once created, the JWT is issued to the client (e.g., a web browser or mobile app) as part of the authentication response.

4. The client receives the JWT and stores it securely, typically in local storage or a secure cookie.

Token Transmission:

5. The client includes the JWT in subsequent requests to the server, typically in the HTTP Authorization header or as a query parameter.

6. The server verifies the JWT's integrity and authenticity before processing the request.

Token Validation:

7. Upon receiving a request with a JWT, the server validates the token to ensure that it has not been tampered with and that it was issued by a trusted authority.

8. This involves verifying the token's signature (if applicable), checking its expiration time, and validating any additional claims.

Token Refresh (Optional):

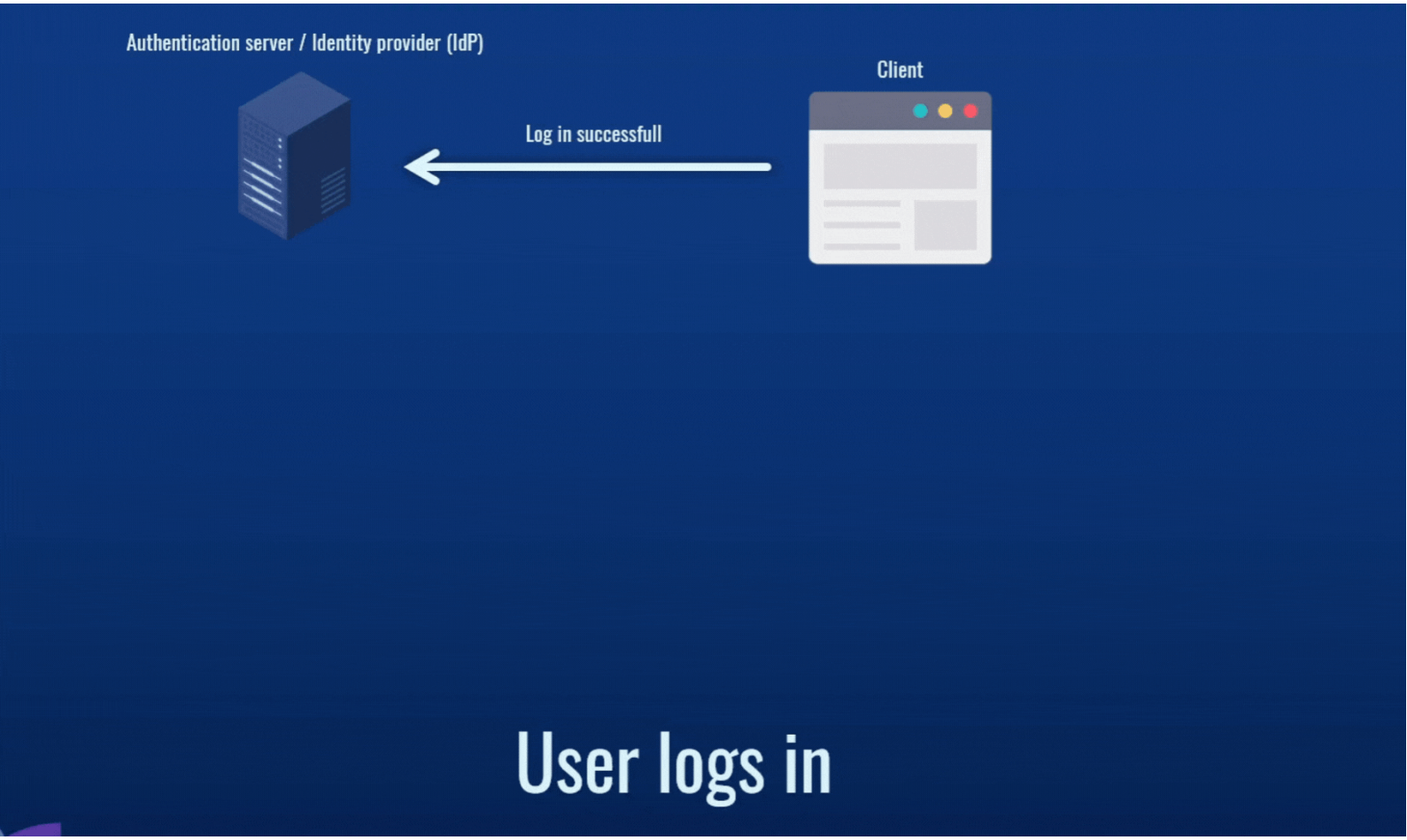
9. If the JWT has an expiration time (specified in the "exp" claim), the client may need to refresh it before it expires to maintain the user's session.

10. This typically involves requesting a new JWT from the authentication server using a refresh token or other means.

Token Expiration/Revocation:

- Eventually, the JWT expires either due to reaching its expiration time or being revoked by the authentication server.
- Once expired, the JWT is no longer considered valid, and the client may need to re-authenticate to obtain a new token.
- Token revocation mechanisms, such as token blacklists or logout endpoints, can be used to invalidate JWTs before they expire.

Demo



* For real example refer to the “IAM” documentation.
https://github.com/Matthiasbrat/learn-fast/tree/main/English/Authentication/IAM-Identity-and-access-management

Security tips

HTTPS	Always transmit JWTs over HTTPS to encrypt the data in transit and prevent interception or tampering by attackers.
Validate signature	Verify its signature to ensure that it was issued by a trusted authority and has not been tampered with. Use the appropriate cryptographic algorithm and the issuer's public key or shared secret to validate the signature.
Expiration time	Include an expiration time (exp claim) in JWTs to limit their validity period.
Short-lived tokens	Issue short-lived JWTs with relatively short expiration times to reduce the window of opportunity for attackers to misuse stolen tokens. Implement token refreshing mechanisms to obtain new tokens when needed.
Sensitive information	Avoid including sensitive information (such as passwords or personally identifiable information) in the JWT payload, especially if the token is transmitted in client-side requests.
Access controls	Verify that the JWT contains the necessary claims (e.g., user roles or permissions) before granting access to protected resources.
Private keys	Safeguard private keys used for signing JWTs to prevent unauthorized access or misuse.
Token revocation	Implement mechanisms for token revocation or invalidation, such as maintaining a token blacklist or using token revocation lists (TRLs). Invalidate JWTs when a user logs out, changes their password, or their session expires.
Audience	Verify that the audience (aud claim) specified in the JWT matches the intended recipient of the token. This helps prevent token replay attacks where a token intended for one service is used to access another service.
Secure token storage	Store JWTs securely on the client side, using mechanisms such as HTTP-only cookies, local storage, or session storage.

Libraries

Language / Framework	Library
JavaScript / Node.js	jsonwebtoken
Java	jjwt
Python	pyJWT
Ruby	jwt
PHP	lcobucci/jwt
Go	jwt-go
.NET/C#	System.IdentityModel.Tokens.Jwt
Ruby on rails	knock
Django	django-rest-framework-simplejwt
Spring Boot	spring-security-jwt
Quarkus	SmallRye JWT

Sources

*Source: JSON Web Tokens - jwt.io
*Source: RFC 7519: JSON Web Token (JWT) (rfc-editor.org)
*Source: RFC 7517: JSON Web Key (JWK) (rfc-editor.org)
*Source: RFC 7516: JSON Web Encryption (JWE) (rfc-editor.org)
*Source: RFC 7515: JSON Web Signature (JWS) (rfc-editor.org)
*Source: RFC 7518: JSON Web Algorithms (JWA) (rfc-editor.org)

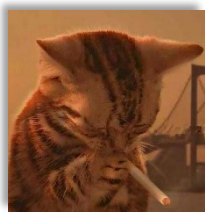
Made by :

Matthias Brat

Socials:

github.com/matthiasbrat

stackoverflow.com/users/17921879/matthias-b



matthiasbrat.dev