https://blog.angular-university.io/angular-jwt/

JWT - JSON Web Tokens

Pre-requisite

———————

1. Use HTTP’s for all the requests

2. SERVER ———Every request should have signed JWT , before reaching to server

[JWT is small + signed serialised JSON ]

3. CLIENT ——— Client have to obtain signed JWT before making requests , as every request must include signed JWT’s

Client Obtains JWT ———————— Server Recives

Client Needs Internet —————————— Server does not , as it stores JWT

Important

————————

————————

1. Deploy JWT secret at servers - In order to verify JWT Signature , the secrete must be deployed at server

2. Server decides and tell client ———— If client is authorised to access a resource or execute a following operation

[ HEADER BODY-PAYLOAD SIGNATURE ]

HEADER - Information how the JWT is encoded

Information means what type of Hashing Mechanism is used to generate JWT token

<https://scotch.io/tutorials/the-anatomy-of-a-json-web-token#:~:text=The%20payload%20will%20carry%20the,names%2C%20and%20private%20claim%20names>.



Header has **2** parts



{

type - JWT : this is just an information

Which algorithm is used to generate token - e.g HMAC SHA256

}

Payload - Ye sab kuch hai , also called JWT claims

Payload == Claims

Payload



{

User information – what to be transferred

Miscellaneous information about JWT

{

expity, intended recipient , subject and issuance, iss ,sub , exp etc.

}

}

Signature - Hash of **3** components



{

1 . Header

2. Payload

3. Secrete – This part is held by authentication server called the private key . Secrete is a signature held by server. Basically this is a server’s signature . Old tokens and new tocken is signed by secrete held by server . Ye server ke sign hai . Authentication server ke sign

}

————

Question – Where can you use the token ?

Answer – In url , header and in the POST parameter.

Very well explained : <https://stackoverflow.com/questions/27301557/if-you-can-decode-jwt-how-are-they-secure>

[

Header – Meadata about token

Paylod – information , not encrypted and will be encoded in the signature

Signature – Validation {Hashing (Header+Payload+Signature on Server)} – if server validate and then only wil allow an access to its services

]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Important Understanding of JWT

1. JWT is not about encryption its about the validation , your secrete key in signature is validated against the secrecte key stored at server
2. JWT token must travel over secured network with https and not http , because its not encrypted
3. Authentication / Validation:
   1. Application Server – Validates JWT
   2. Authentication Server – Generates JWT

First :::::Client to -----🡪 Authentication Server to --🡪 server with signature (store and send server to client)---🡪 client store in cookie or local storage and sends with every request --🡪

Second :::::Client sends encoded request with signature post applying Hashing algorithm 🡪 server decode using hashing and match the key or generate the signature using the key [public + private ]

Third ::: Now validated , so can access services and are ok to proceed

MAC – Message Authentication Code [header**.**payload.signature] All are seperted by dots

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/------------------------------------------------------------------------------------------

Basic Understanding in Depth

1. Base64 – encoding format, JWT is encoded using Base64 encoding format

JWT Token = [Base64(Header) + Base64(Payload) + Base64(SecreteKey)]

1. Two Types of signature
   1. HS-256 Digital Signature (function name is SHA-256 )
   2. RS-256 Digital Signature

Difference between Encryption and Hashing :

1. Encryption is a reversible action called **decryption**

Hashing no reversible is possible , we have to apply the hashing algorithm again to get the original value i.e **Reproducible**

**Important Points**

1. HS-256 is Hashing Alorithm and function is SHA
2. RS-256 is a encryption Algorithm
3. Encryption is slower than Hashing therefore the entire payload is not encrypted
4. RS-256 is safer than HS-256 Why?

(i). HAS-256

* 1. HS-256 : Anyone can be Creator and Validator i.e both Application server and Authentication server , hence easy to decipher
  2. Main Reason: Previous secrete key must be known in HS-256 , hence rotation can take time
  3. Hashing mechanism

(ii). RS-256

1. Authentication server – can only create key called private key
2. App Server – can only Validate key called public key
3. Therefore Cannot be deciphered , as private key cannot be known at any cost.
4. Process:
   * 1. HS-256
        + 1. HS-256 – Hashing Alorithm
          2. SHA-256 – function to be applied
          3. Client 🡪Authentication server Generate JWT Token Hashing function to generate hash(Base64(Header+Payload+signature(secrete-key))-🡪 Send to server saves 🡪 Server sends to client and then (Request + JWT taken is send with every request) --- > Server now apply hashing on JWT and if the key matches after Hashing i.e Apply hashing on hash(Base64(Header+Payload+signature(secrete-key)) is signature matches then all ok. And Validated to proceed.
          4. **Note : Request + JWT – Request is separate and JWT is separate : Payload does not means request ,don’t be confused , payload is some set of information to generate token , and request is an actual JSON data to be processed**
     2. RS-256
        + 1. Encryption/Decryption Algorithm and not hashing Algorithm
          2. Public Private key concept : RSA key is Private key at Authentication server side only, it can only hold
          3. Public key – all app servers holds
          4. Encryption At Authentcation Server using Private key – Encrypting using hashing on header and payload

Encryption {(Hashing-

{SHA256(Header+Payload-not request its separate)) }+ Private Key}.

* + - * 1. Decryption – Using Public key
        2. Encrypted Signature Hash is compared with Decrypted Signature Hash – If matches awesome
        3. Note remember Both the Hashes are compared , one signature hash is send along with. (request +JWT token ) and other is generated with same mechanishm using the public key (Hash Encrypt (SHA(Header+Payload)+Public Key)) : If both Hash matches then only its ok.

***HS-256 :Encoded Hash should be == Decoded Hash***

***RS-256 : Encrypted Hash should be == Decrypted hash***

**Signature is produced at both end and are matched**

------------------------------------------------------------------------------------------/

Receiver - Validates the content of payload , by validating the signature.

Header - Contains the information of JWT

{

“alg”:”RS256”, This is a signature type used for token

“typ”:”JWT”

}

PAYLOAD / BODY - “MEAT —— > { CLAIMS LIVES HERE } ” - No need to put important information , as JWT is not encrypted

SIGNATURE — Provides the security. Its the last part and is also called **“MAC - Message Authentication Code “**

**So Header - has info about JwT, which algorithm we will use for signature , But actual signature is the last part , and payload is a data**

====================================================

Client —————————— Application Server Authentication External Server { LDAP or AUTH0 or anything}

App Server and Auth Server are completely independent

App server is stateless , and does not need to store secrete key or tokens

Understanding - JWT its a JSON payload containing a claim

Key property is , to authenticate we just have to look - no need to store anywhere , or contact third party server.

**Because** it carries **MAC (** Message Authentication Code **)**