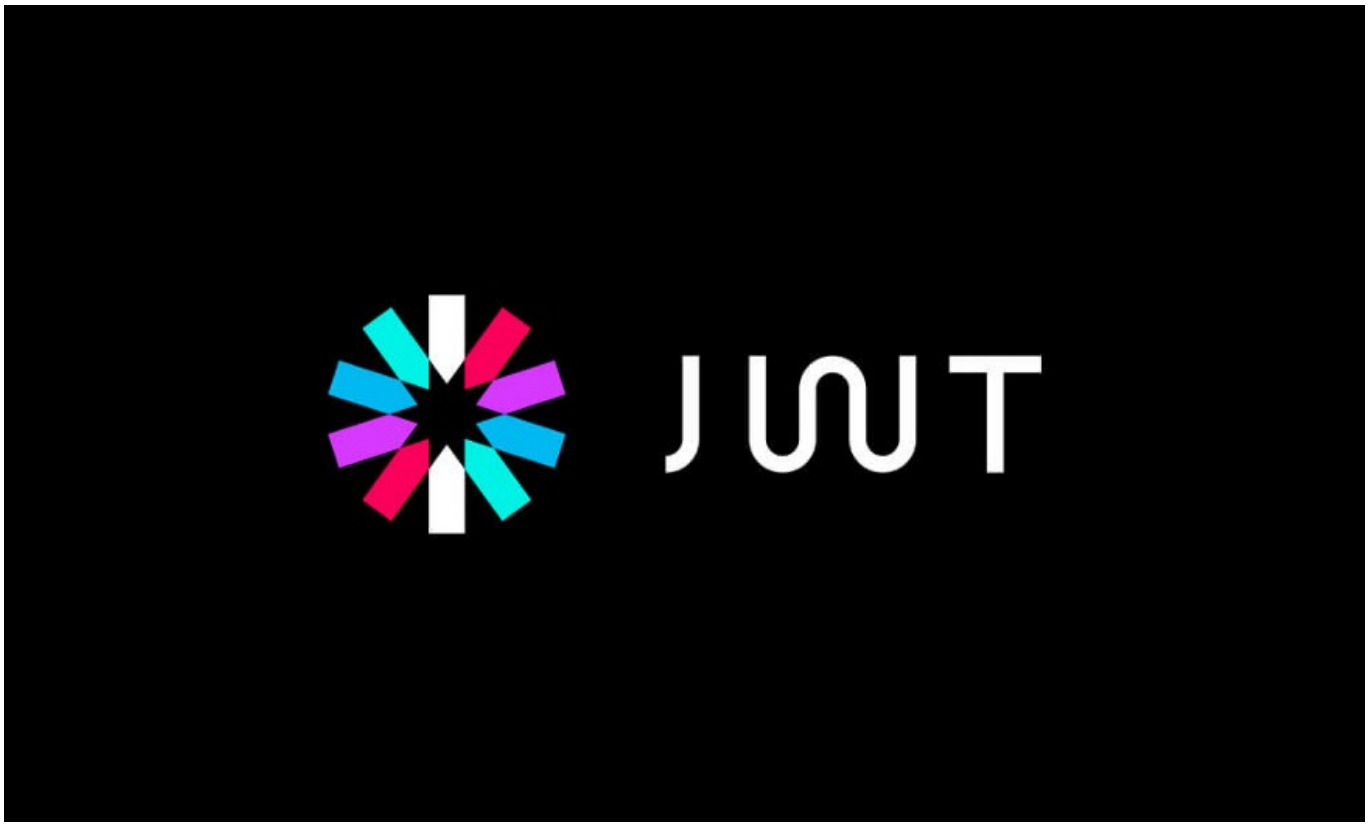
Five easy steps to understand JSON Web Tokens (JWT)

Shaurya Sharma Follow
Dec 21, 2019 · 2 min read



JSON Web Token (JWT)

> *JSON Web Token (JWT) Is a JSON object and it is considered one of the safest ways to transfer information between two participants. To create it, you need to define a header with general information on the token, payload data, such as the user id, his role, etc. and signatures.*

The application uses *JWT* to verify user authentication as follows:-

1. First, the user logs on to the authentication server using an authentication key (it can be a *username / password* pair , or a *Facebook* key, or a *Google* key, or a key from another account).
2. The authentication server then creates the *JWT* and sends it to the user.
3. When the user makes a request to the application API, he adds the previously received *JWT* to it .
4. When a user makes an API request, the application can check whether the user is what he claims to be, using the *JWT* request . In this scheme, the application server is configured to be able to check whether the incoming *JWT is* exactly what was created by the authentication server (the verification process will be explained later in more detail).

JWT structure

JWT consists of three parts: header, Payload and signature. Let's go through each of them.

Step 1. Create a HEADER

The *JWT header* contains information on how the *JWT* signature should be calculated . A header is also a *JSON* object that looks like this:

> ***header = { “alg”: “HS256”, “typ”: “JWT”}***

It will be used when creating the signature. **HS256** Nothing but, **HMAC-SHA256** for its calculation you need only one secret key.

Step 2. Create PAYLOAD

Payload is the **payload** that is stored inside the *JWT* . In the example the authentication server creates a *JWT* with information about the **userId**

> ***payload = { “userId”: “b08f86af-35da-48f2–8fab-cef3904660bd” }***

There is a list of standard *applications* for *JWT* payload — here are some of them:

- *iss* (issuer) — defines the application from which the token is sent.

- *sub* (subject) — defines the topic of the token.
- *exp* (expiration time) — token lifetime.

Step 3. Create SIGNATURE

The **base64url** algorithm encodes the header and payload created in **step *1* and *2*** . The algorithm concatenates encoded strings through a dot.

> ***// header eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9***
>
> ***// payload***
> ***eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ***
>
> ***// signature -xN_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM***

Step 4. Now combine all three JWT components together

Now that we have all three components, we can create our *JWT*

const token = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' + encodeBase64Url(signature)

OR

// JWT Token
//eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ.-xN_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM

You can try to create your own *JWT* Online *jwt.io* .

JavaScript Jwt Token Hacking Shaurya Sharma Bug Bounty


Download on the
App Store


GET IT ON
Google Play