Beaconfire Inc, Home Work, Week5 Day19.
By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank
Short Answer:

1. What is the difference between authentication vs authorization?

-> Authentication: Verify who a user is.

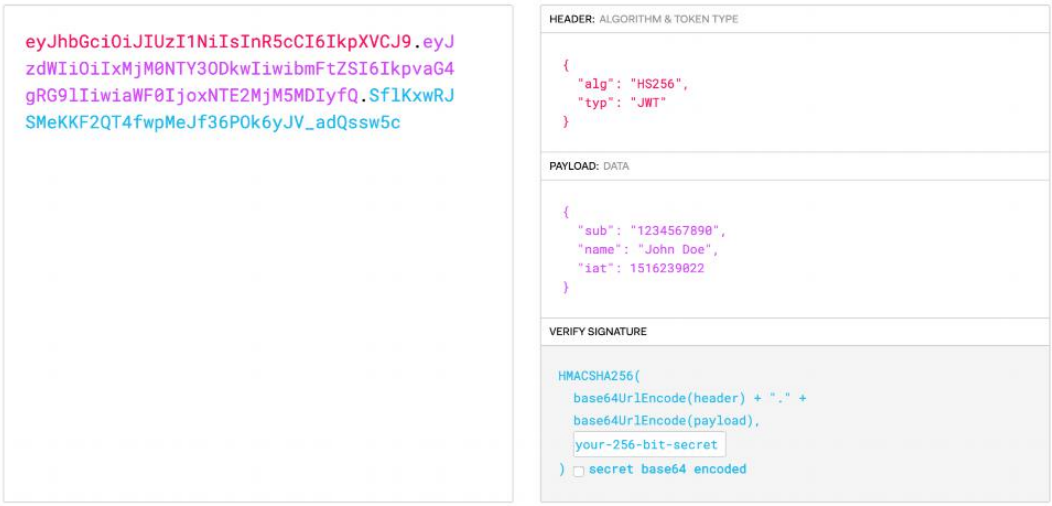
-> Authorization: Verify what access does a user have.

2. What is JWT?

-> • Json Web Token (JWT) — it is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

• In its compact form, JSON Web Tokens consist of three parts separated by dots (.)

- Header
- Payload
- Signature



3. What is Spring Security and what can it help us to achieve.

->Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications.

-> • Spring Security is another module provided by Spring team

• Spring Security integrates with the Servlet Container by using a standard Servlet Filter.

• This means it works with any application that runs in a Servlet Container.

• More concretely, you do not need to use Spring in your Servlet-based application to take advantage of Spring Security.

-> Spring Security Core Components for Authorization:

• SecurityContextHolder — to provide access to the SecurityContext

• SecurityContext — to hold the Authentication (and possibly request-specific security information).

• Authentication — to represent an authenticated user

• UserDetails — to provide the necessary information to build an Authentication object from your application's DAOs or other source of security data.

• GrantedAuthority — to reflect the application-wide permissions granted to a principal.

-> Spring Security Core Components for Authentication:

• AuthenticationManager — to perform authentication process base on the supplied AuthenticationProvider
• AuthenticationProvider — to tell AuthenticationManager how to authenticate a user
• UserDetailsService — to create a UserDetails when passed in a String-based username (or certificate ID or the like).

4. How do we secure a RESTful API?

By using Spring Security + JWT work together:
• Before users can use a web service, they need to be authenticated.
• Users need to provide their username and password to the Authentication Server.
• Authentication Server then authenticates these information (can be done using Spring Security as well)
• If provided username and password are correct, then an JWT will be generated using certain key and algorithm, containing user identification information and associated authorities.
• After being authenticated, users can then use our web service.
• When users send requests to our web service, the JWT need be included in every request (normally in the request header).
• When web service receive the requests, it needs to decrypt the JWT using the same key and algorithm to authenticate the JWT.
• If the JWT is valid, then it means that the current user is already logged in.
• Then we need to use Spring Security to store the current user's authorities, so that Spring Security can perform Authorization for us.
• And then, we need to configure Spring Security in our web service.

https://medium.com/javarevisited/how-to-secure-the-rest-apis-b682e21821a1

• API key :

When to use: It fits designing APIs for 3rd parties services integration and limited access, not public access purpose. For instance, your company provides SMS gateway as web services and other companies would like to use your services.

How to use: validate the api-key either request param or Request Header.

How it works: Create Servlet Filter Security and validation either looking at the request param api_key and X-API-Key as HEADER and whitelist IPs address (optional). So every user makes a request, the Filter Security will check and validate first before it continues into the API controller part.

• Basic authentication

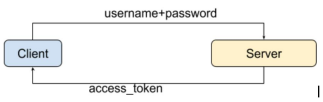
When to use: It fits designing APIs for 3rd parties services integration and limited access, not public access purpose, it is very similar to the API-KEY. How to use: Request Header Authorization BASIC mode (username + password + Base64 algorithm encoding How it works: One simple way is to create Servlet Filter Security and validation Authorization BASIC mode. So every user makes a request, the Filter Security will check and validate first before it continues into the API controller part

• Oauth2

When to use: When the users own their password with the context of the websites, e-commerce, mobile backend, and others.

How to use: username + password + access_token + expiration token.

How it works: The main idea of the Oauth 2.0 standard is that after a user log-in into the system with their username and password successfully, the server will return back with the tokens (access token and a refresh token). Within the access token, The server can exchange with the current user login such as user-profile, roles, permission, and more.



• Oauth2 + JWT

When to use: It seems similar to the Oauth2 context above.

How to use: username + password + JSON map+ Base64 + private key + expiration date

How it works: When user login with username + password for the first time, the system will exchange back the access token, which this token represents a JSON map containing all user information, such as user profiles, roles, and privileges, encoded with Base64 and signed with a private key.

The main difference with Oauth2 is that we store user information state in the token, while services are stateless. This means that the server can decrypt the token into the user information state and there is no need for additional looking up from the database with this token. This is a huge benefit in reducing the load on the server.