Spring Security

By

Anand Kulkarni

anand.kulkarni1@zensar.com

Table of Content

Module	Topic
Module 1:	Spring Security Basics
Module 2:	Adding spring security in Spring Boot app
Module 3:	Authentication
Module 4:	Authorization
Module 5:	JWT
Module 6:	OAuth2

What is Spring Security?

Spring security is an application framework that provides application level security including:

1) Login & Logout functionality
2) Allow/block access to URLs to logged-in users
3) Allow/block access to URLs to logged-in users and with certain roles

Adding spring security will handle most of common vulnerabilities like Session Fixation, Clickjacking etc.

What we do with Spring Security?

1) Username/password authentication
2) SSO/Okta/LDAP
3) Application level Authorization
4) Intra App Authorization like OAuth
5) Microservice level security (using tokens, JWT)
6) Method level security

Spring Security core concepts

1) Authentication –

   Its similar to the question "Who are you?“. Authentication can have 2 types: ‘Knowledge based authentication’ i.e. asking for username/password or pin etc. & Possession based authentication i.e. asking to enter OTP sent to phone etc.

2) Authorization –

   Its similar to the question "Can this user do this“?

Spring Security core concepts

3) Principal –

Principal is the person you have been identified through the process of authentication. It means Principal is a "Currently logged in user".

4) Granted Authority –

User is trying to do something in the application & application has authorized him to do so, its call 'Granted Authority'.

5) Roles –

Role is similar to the group of authorities. For example manager, admin, clerk etc.

Adding security to Spring Boot App

Add dependency into pom.xml i.e. spring-boot-starter-security.

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
```

Just by adding this dependency, you will notice that it will automatically show you login page if you try to access any REST service. This happens because Spring Security introduces Filters that intercept your all requests.

Behavior added by Spring Security

A) It adds mandatory authentication for all URLs except few like error page.

B) Adds login form (user is 'user' & its password finds on console)

If you wish to override this default username then simply add below properties inside application.properties file:

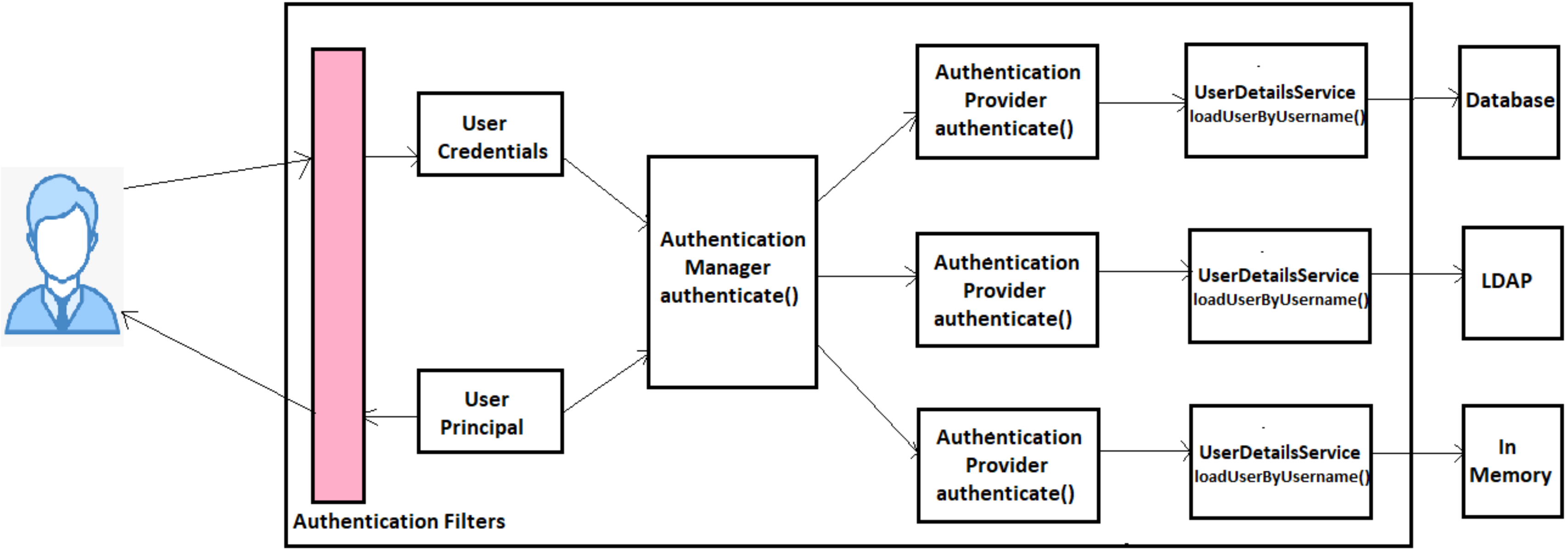
spring.security.user.name=anand

spring.security.user.password=anand123

C) It handles login error i.e. check whether entered credentials are valid or not.

D) Creates a user and sets default password.

Authentication flow in Spring Security


User Credentials
User Principal
Authentication Filters
Authentication Manager authenticate()
Authentication Provider authenticate()
Authentication Provider authenticate()
Authentication Provider authenticate()
UserDetailsService loadUserByUsername()
UserDetailsService loadUserByUsername()
UserDetailsService loadUserByUsername()
Database
LDAP
In Memory

Spring Security Authentication Flow

Authentication Flow Explained…

1) User passes his credentials to Authentication Filter.
2) Authentication Filter is a chain of internal filters those intercept every user request.
3) Authentication Filter passes user credentials to AuthenticationManager by calling authenticate() method.
4) AuthenticationManager finds out suitable AuthenticationProvider & calls authenticate() method on it.
5) In order to retrieve the user details, AuthenticationProvider calls loadByUsername() method on UserDetailsService.

Authentication Flow Explained…

6) Once user details are received, AuthenticationProvider writes logic to confirm whether user credentials are valid or not.

7) If credentials are invalid then AuthenticationProvider throws exception to the Authentication Filter, so that Authentication Filter forwards error page to the client.

8) If credentials are valid then AuthenticationProvider sends User Principal object to the Authentication Filter. This Principal object is stored inside session so that subsequent user requests won't require to pass credentials again.

Authentication Manager

- Authentication Manager manages authentication in spring security.
- It has method calls authenticate() that returns success or exception.
- If you wish to override default behavior of building AuthenticationManager then write a class that defines UserDetailsService bean.

Overriding Authentication Manager

```
@Configuration
public class SecurityConfiguration {
        @Bean
        public UserDetailsService getUserDetails() {
                UserBuilder userBuilder = User.builder();
                UserDetails tomUser =
userBuilder.username("tom").password(passwordEncoder.encode("tom123")).roles("USER").build();
                UserDetails jerryUser =
userBuilder.username("jerry").password(passwordEncoder.encode("jerry123")).roles("ADMIN").build();
                return new InMemoryUserDetailsManager(tomUser, jerryUser);
        }
```

Configure authorization in Spring Security

1) Using HttpSecurity, you can configure the endpoint paths & its security restrictions.
2) In order to implement Authorization, you need to return bean of the type SecurityFilterChain.

Configure authorization in Spring Security

```
@Configuration

public class SecurityConfiguration {

        @Bean

        public SecurityFilterChain getSecurityFilterChain(HttpSecurity httpSecurity) throws Exception {

                httpSecurity.csrf(csrf->csrf.disable()).authorizeHttpRequests(auth->

                auth.requestMatchers("/user").hasAnyRole("USER", "ADMIN")

                .requestMatchers("/admin").hasRole("ADMIN")

                .requestMatchers("/all").permitAll()

                .anyRequest().authenticated()).formLogin(Customizer.withDefaults());

                return     httpSecurity.build();

        }}
```

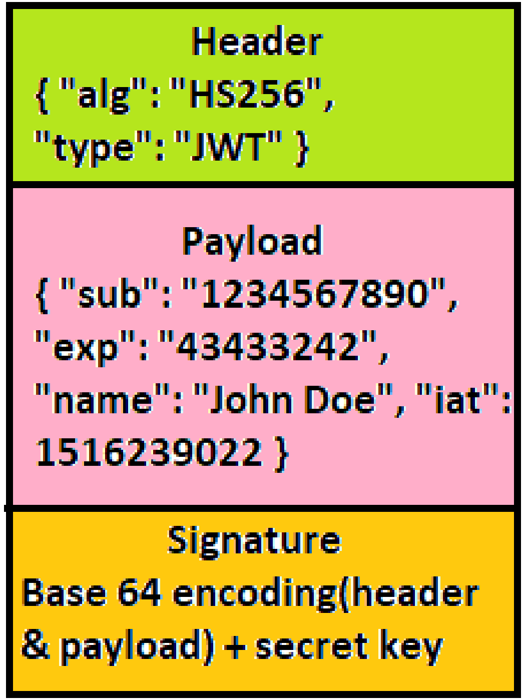
JWT

What is JWT?

1) JWT stands for JSON Web Token. Refer https://jwt.io/ for more information.
2) JWT is an open standard to transmit the information securely between parties as a JSON object.
3) JWT is used for Authorization where client passes a header having key as “Authorization” & value as “Bearer <token>”
4) JWT is most suitable in SSO (single sign on) applications due to easy usage across different domains. JWT is also used for securely transmitting data across parties.

JWT Structure

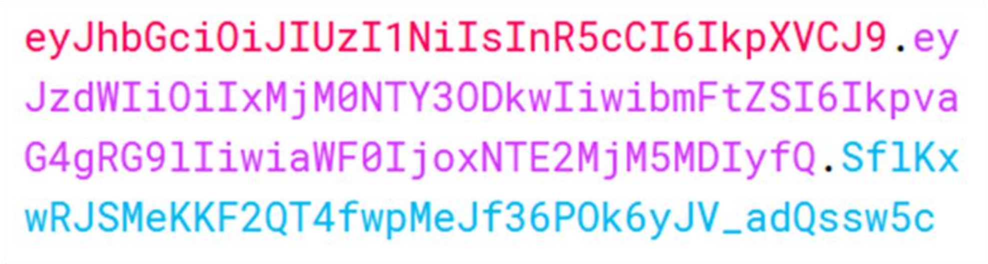
JWT token is divided into 3 parts: Header, Payload & Signature.



Header
{ "alg": "HS256",
"type": "JWT" }

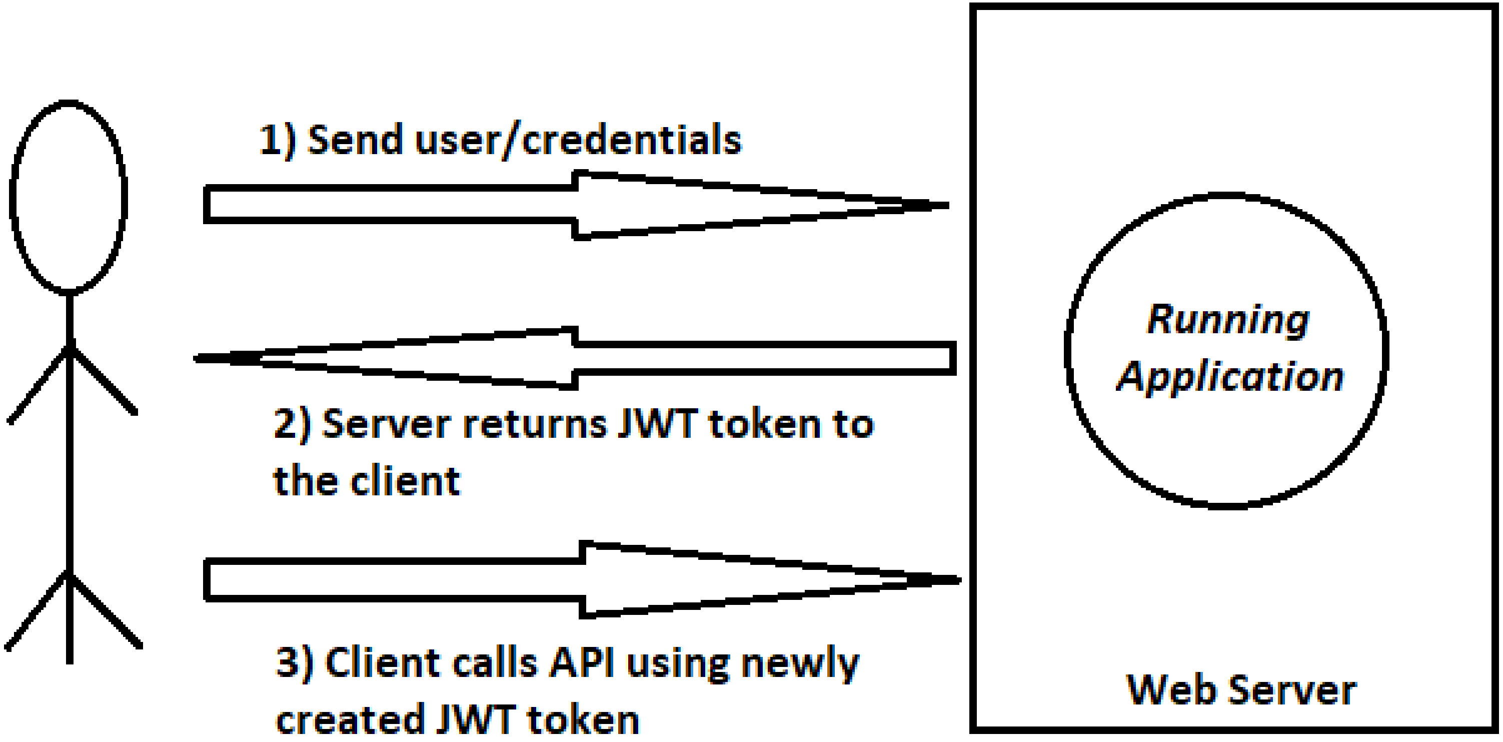
Payload
{ "sub": "1234567890",
"exp": "43433242",
"name": "John Doe", "iat":
1516239022 }

Signature
Base 64 encoding(header
& payload) + secret key



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

JWT Flow



1) Send user/credentials
Running Application
2) Server returns JWT token to the client
3) Client calls API using newly created JWT token
Web Server

OAuth2

What is OAuth2?

1) In OAuth word, 'Auth' is meant for 'Authorization'. Thus OAuth stands for ‘Open Authorization’.
2) OAuth is meant for a service to authorize another service.
3) In OAuth, services try to access each other on behalf of the user.
4) Earlier we had OAuth 1.0 but now its OAuth 2.0.

OAuth2 Terminologies

1) **Resource:** Resource is something that you wish to access. for example a photo, pdf file etc.

2) **Resource owner:** Owner of the resource who can grant permission to the resource.. For example google user.

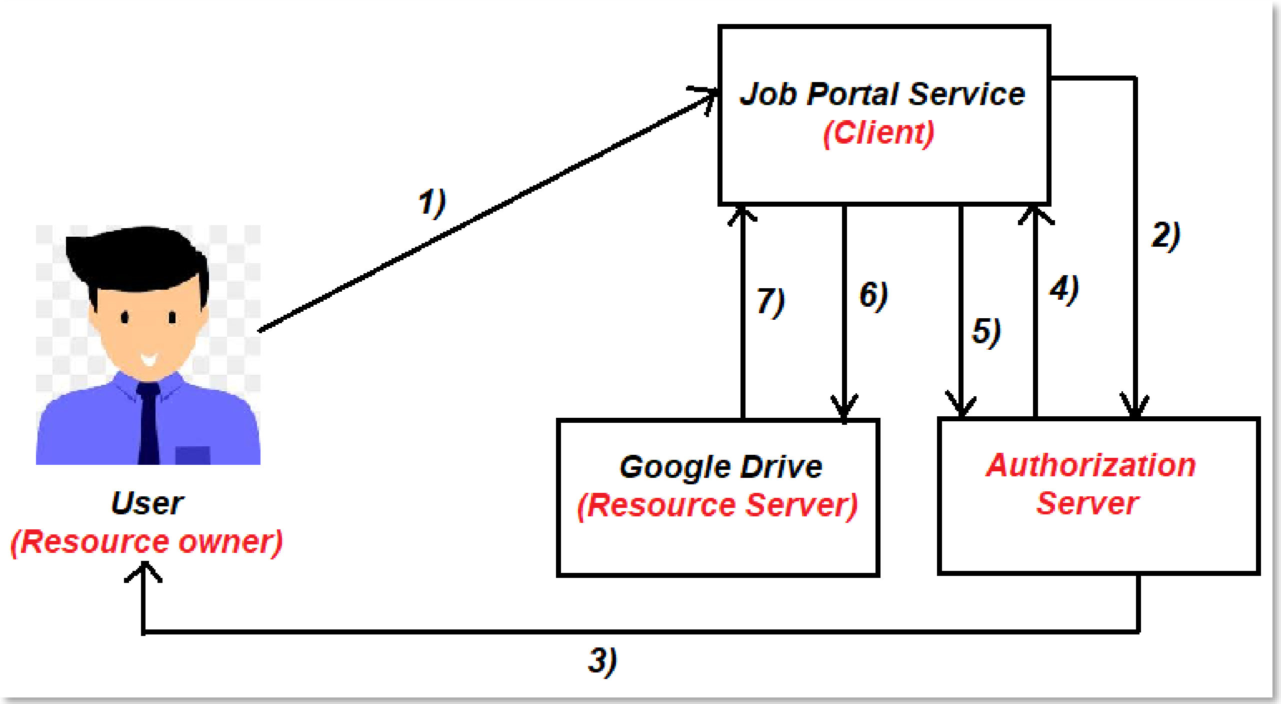
3) **Resource server:** It is the server that is protecting the resource. for example google drive.

4) **Client:** Client is an entity who is requesting for getting access of the resource on the behalf of the resource owner. for example ‘Job portal service’.

OAuth2 Terminologies

5) **Authorization server:** Authorization server issues access tokens to the client. When user validates the Client, Authorization server first sends Authorization token. Using this token, client will access Authorization server to get Access token. Once access token is received, the client to contact Resource Server to get acceeess of the resource.

OAuth2 flow


Job Portal Service
(Client)
1)
2)
7)
6)
5)
4)
User
(Resource owner)
Google Drive
(Resource Server)
Authorization
Server
3)

OAuth2 flow

1) Resource owner(User) logins into Job portal service. Resource owner wants to upload his resume to job portal service(client) but resume is placed on google drive(Resource server). Hence, Resource owner requests client to get resume from Resource server.
2) Client contacts 'Authorization Server' to get access to Resource owner's resume.
3) Authorization Server contacts resource owner and asks whether it should share the resume document with the client?

OAuth2 flow

4) If resource owner grants permission then Authorization server sends Authorization token to the client.

5) Client contacts Authorization server with recently received Authorization token to get second token i.e. Access Token. If Authorization token is found valid then Authorization server sends Access Token to the client.

6) Client contacts Resource Server with Access Token & request for resume document.

7) Resource server validates the Access Token with Authorization Server & if found valid then it provides the resume document access to the client.

Thank you!!