**Objectives**

* Demonstrate implementation of JWT Authentication for RESTful Web Service using Spring Security
  + Securing web application with Spring Security, HTTP Basic Authentication, @Configuration, @EnableWebSecurity, AuthenticationManagerBuilder, in memory authentication, URL authorization configuration with antMatchers, JWT stands for JSON Web Token, JWT Process Flow, JWT structure, Base64 encoding and decoding, Authorization header, create JWT, set token expiry, authorize requests using Spring Filters
    - Spring Security Example - https://www.mkyong.com/spring-boot/spring-rest-spring-security-example/
    - JWT User Manual - https://github.com/jwtk/jjwt#install-jdk-maven
    - Authorizing JWT - https://auth0.com/blog/implementing-jwt-authentication-on-spring-boot/

**Create authentication service that returns JWT**   
  
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
**Request**

curl -s -u user:pwd http://localhost:8090/authenticate

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

**Program:**

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>jwt-auth-springboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>JWT Auth Spring Boot</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/>

</parent>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>2.7.18</version>

</plugin>

</plugins>

</build>

</project>

**SecurityConfig.java:**

package com.example.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

@Configuration

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User

.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate", "/").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic();

return http.build();

}

@Bean

public AuthenticationManager authManager(HttpSecurity http) throws Exception {

AuthenticationManagerBuilder authBuilder = http.getSharedObject(AuthenticationManagerBuilder.class);

authBuilder.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());

return authBuilder.build();

}

}

**AuthController.java:**

package com.example.jwt.controller;

import com.example.jwt.model.AuthResponse;

import com.example.jwt.util.JwtUtil;

import javax.servlet.http.HttpServletRequest;

import org.springframework.web.bind.annotation.\*;

import java.util.Base64;

@RestController

public class AuthController {

@GetMapping("/")

public String home() {

return "✅ JWT Authentication Server is running!";

}

@GetMapping("/authenticate")

public AuthResponse authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Basic ")) {

throw new RuntimeException("Missing or invalid Authorization header");

}

String base64Credentials = authHeader.substring("Basic ".length()).trim();

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

String[] values = credentials.split(":", 2);

if (values.length != 2 || !values[0].equals("user") || !values[1].equals("pwd")) {

throw new RuntimeException("Invalid credentials");

}

String token = JwtUtil.generateToken(values[0]);

return new AuthResponse(token);

}

}

**AuthResponse.java:**

package com.example.jwt.model;

public class AuthResponse {

private String token;

public AuthResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

**JwtUtil.java:**

package com.example.jwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "mysecretkey";

private static final long EXPIRATION\_TIME = 1000 \* 60 \* 10;

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**JwtApplication.java:**

package com.example.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtApplication {

public static void main(String[] args) {

SpringApplication.run(JwtApplication.class, args);

}

}

**Application.properties:**

server.port=8080

**.replit:**

language = "java"

run = "mvn spring-boot:run"

[[ports]]

localPort = 8080

externalPort = 80

**replit.nix:**

{ pkgs }: {

deps = [

pkgs.maven

pkgs.openjdk11

];

}

**Output:**



