**Securing RESTful Web Services with Spring Security**

**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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<properties>

<java.version>17</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>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.anyRequest().authenticated()

.and()

.httpBasic();

}

}

**CountryController.java**

package com.cognizant.springlearn.controller;

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

import java.util.\*;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<Map<String, String>> getCountries() {

List<Map<String, String>> countries = new ArrayList<>();

countries.add(Map.of("code", "US", "name", "United States"));

countries.add(Map.of("code", "DE", "name", "Germany"));

countries.add(Map.of("code", "IN", "name", "India"));

countries.add(Map.of("code", "JP", "name", "Japan"));

return countries;

}

}

**SpringLearningApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**application.properties**

server.port=8090

**Output:**

****

**Creating users and roles in Spring Security**

**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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

</parent>

<properties>

<java.version>17</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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**CountryController.java**

package com.cognizant.springlearn.controller;

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

import java.util.\*;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<Map<String, String>> getCountries() {

List<Map<String, String>> countries = new ArrayList<>();

countries.add(Map.of("code", "IN", "name", "India"));

countries.add(Map.of("code", "US", "name", "United States"));

countries.add(Map.of("code", "JP", "name", "Japan"));

return countries;

}

}

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.\*;

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

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

import org.springframework.security.config.annotation.web.configuration.\*;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private static final Logger LOGGER = LoggerFactory.getLogger(SecurityConfig.class);

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

@Bean

public PasswordEncoder passwordEncoder() {

LOGGER.info("Start passwordEncoder()");

return new BCryptPasswordEncoder();

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable().httpBasic()

.and()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER")

.anyRequest().authenticated();

}

**}**

**application.properties**

server.port=8090

**Create authentication service that returns JWT**

**pom.xml**

<dependencies>

<!-- Existing dependencies -->

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<!-- JWT Library -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

**JwtUtil.java**

package com.cognizant.springlearn.security;

import io.jsonwebtoken.\*;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

private final long EXPIRATION\_TIME = 1000 \* 60 \* 60; // 1 hour

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

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

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

.signWith(key)

.compact();

}

}

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.security.JwtUtil;

import jakarta.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.\*;

import org.springframework.security.authentication.\*;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

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

import java.util.\*;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<Map<String, String>> authenticate(HttpServletRequest request) {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

String username = auth.getName();

String token = jwtUtil.generateToken(username);

Map<String, String> response = new HashMap<>();

response.put("token", token);

return ResponseEntity.ok(response);

}

}

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.\*;

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

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

import org.springframework.security.config.annotation.web.configuration.\*;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private static final Logger LOGGER = LoggerFactory.getLogger(SecurityConfig.class);

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

@Bean

public PasswordEncoder passwordEncoder() {

LOGGER.info("Start passwordEncoder()");

return new BCryptPasswordEncoder();

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable().httpBasic()

.and()

.authorizeRequests()

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

.anyRequest().authenticated();

}

}

**application.properties**

server.port=8090

**Generting Token:**

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

**Output:**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNjg5ODIxNTkzLCJleHAiOjE2ODk4MjUxOTN9.6eYQ0Fl5JWBvVZZyoqydgNVA5j6L3DO9JIzTtMQ8\_gY"}

**Create authentication controller and configure it in SecurityConfig**

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START - /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END - /authenticate");

return map;

}

}

**SecurityConfig.java**

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable()

.httpBasic().and()

.authorizeRequests()

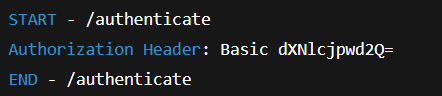
.antMatchers("/countries").hasRole("USER")

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated();

}

**Output:**



**Read Authorization header and decode the username and password**

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START - /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

LOGGER.debug("Decoded user: {}", user);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END - /authenticate");

return map;

}

private String getUser(String authHeader) {

LOGGER.debug("START - getUser()");

// Extract base64 encoded credentials by removing "Basic " prefix

String encodedCredentials = authHeader.substring("Basic ".length());

LOGGER.debug("Encoded credentials: {}", encodedCredentials);

// Decode Base64 to byte array

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

// Convert byte array to string ("user:pwd")

String decodedCredentials = new String(decodedBytes);

LOGGER.debug("Decoded credentials: {}", decodedCredentials);

// Extract username (before the colon)

String username = decodedCredentials.split(":")[0];

LOGGER.debug("Extracted username: {}", username);

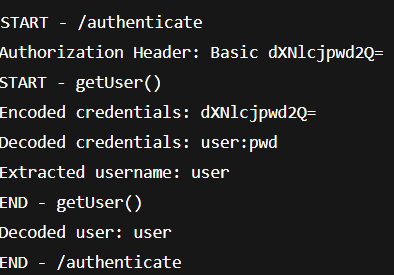
LOGGER.debug("END - getUser()");

return username;

}

}

**Ouput:**

****

**Generate token based on the user**

adding jjwt dependency in **pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START - /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

LOGGER.debug("Decoded user: {}", user);

String token = generateJwt(user);

LOGGER.debug("Generated Token: {}", token);

Map<String, String> map = new HashMap<>();

map.put("token", token);

LOGGER.info("END - /authenticate");

return map;

}

private String getUser(String authHeader) {

LOGGER.debug("START - getUser()");

String encodedCredentials = authHeader.substring("Basic ".length());

LOGGER.debug("Encoded credentials: {}", encodedCredentials);

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes);

LOGGER.debug("Decoded credentials: {}", decodedCredentials);

String username = decodedCredentials.split(":")[0];

LOGGER.debug("Extracted username: {}", username);

LOGGER.debug("END - getUser()");

return username;

}

private String generateJwt(String user) {

LOGGER.debug("START - generateJwt() for user: {}", user);

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 minutes

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

String token = builder.compact();

LOGGER.debug("Generated JWT token: {}", token);

LOGGER.debug("END - generateJwt()");

return token;

}

}

**Curl:**

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

**Output:**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzI..."}

**Authorize based on JWT**

**JwtAuthorizationFilter.java**

package com.cognizant.springlearn.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

private static final Logger LOGGER = LoggerFactory.getLogger(JwtAuthorizationFilter.class);

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

LOGGER.info("Start - JwtAuthorizationFilter Constructor");

LOGGER.debug("AuthenticationManager: {}", authenticationManager);

}

@Override

protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)

throws IOException, ServletException {

LOGGER.info("Start - doFilterInternal()");

String header = req.getHeader("Authorization");

LOGGER.debug("Authorization Header: {}", header);

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(req, res);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

SecurityContextHolder.getContext().setAuthentication(authentication);

chain.doFilter(req, res);

LOGGER.info("End - doFilterInternal()");

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

Jws<Claims> jws = Jwts.parser()

.setSigningKey("secretkey")

.parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

LOGGER.debug("User from JWT: {}", user);

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException ex) {

LOGGER.error("JWT Parsing failed: {}", ex.getMessage());

return null;

}

}

return null;

}

}

**SecurityConfig.java**

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable()

.httpBasic().and()

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}

**Curl:**

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

**Output:**

**curl -s -H "Authorization: Bearer YOUR\_TOKEN\_HERE"** [**http://localhost:8090/countries**](http://localhost:8090/countries)

**Call secured API with JWT token**

curl -s -H "Authorization: Bearer YOUR\_TOKEN\_HERE" http://localhost:8090/countries

**Try tampered or invalid token**

curl -s -H "Authorization: Bearer invalidtoken123" <http://localhost:8090/countries>

**Output:**

