**1. Securing RESTful Web Services with Spring Security**

* **Add Spring Security dependency** in pom.xml:

<dependency>

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

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

</dependency>

* **Create SecurityConfig** class with:
  + @Configuration
  + @EnableWebSecurity
  + Extend WebSecurityConfigurerAdapter
* **Test endpoint:**

curl -s http://localhost:8090/countries

Output will be:

{"status":401,"error":"Unauthorized"}

**2. Creating Users and Roles**

**In SecurityConfig.java:**

@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() {

return new BCryptPasswordEncoder();

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().httpBasic().and()

.authorizeRequests()

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

}

**Test Commands:**

curl -u user:pwd http://localhost:8090/countries # Allowed

curl -u admin:pwd http://localhost:8090/countries # Forbidden (403)

curl -u user:wrongpwd http://localhost:8090/countries # Unauthorized (401)

**3. Limitations of Basic Auth**

* Base64 encoding is **not secure**
* REST is **stateless**, so credentials are sent **on every request**
* Easily decoded online (e.g., YWRtaW46cHdk = admin:pwd)

**4. Understanding JWT**

* JWT = **Header + Payload + Signature**
* Used to securely pass **user identity** between client and server
* Example payload:

{

"sub": "user",

"iat": 1710469185,

"exp": 1710470385

}

**5. Create Authentication Controller**

**Controller:**

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

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

LOGGER.info("START");

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

LOGGER.info("END");

return map;

}

}

**SecurityConfig Update:**

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

**6. Decode Authorization Header**

**In AuthenticationController.java:**

private String getUser(String authHeader) {

String encoded = authHeader.substring(6); // After "Basic "

byte[] decoded = Base64.getDecoder().decode(encoded);

String decodedStr = new String(decoded);

return decodedStr.split(":")[0]; // Returns "user"

}

**7. Generate JWT Token**

**Add dependency in pom.xml:**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

**Generate JWT:**

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1200000)) // 20 minutes

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

return builder.compact();

}

**Sample Response:**

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

**8. Authorize Requests Using JWT**

**Create JwtAuthorizationFilter.java:**

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

public JwtAuthorizationFilter(AuthenticationManager authManager) {

super(authManager);

}

@Override

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

throws IOException, ServletException {

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

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

chain.doFilter(req, res);

return;

}

UsernamePasswordAuthenticationToken auth = getAuthentication(req);

SecurityContextHolder.getContext().setAuthentication(auth);

chain.doFilter(req, res);

}

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();

if (user != null) {

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

}

} catch (JwtException e) {

return null;

}

}

return null;

}

}

**Add Filter in SecurityConfig:**

.addFilter(new JwtAuthorizationFilter(authenticationManager()))

**9. Testing the JWT Flow**

1. **Get token:**

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

1. **Use token:**

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

1. **Use invalid token:**  
   Returns:

{"status":403,"error":"Forbidden"}