**Name**: Amogh Girish Nagarkar  
**Superset ID**: 6403503

**Spring Security Hands-on Report**

**Objective:**  
To secure a Spring Boot RESTful service using in-memory users with roles and restrict access to a specific endpoint based on user roles.

**1. Users and Roles Configuration**

Two in-memory users were created:

* **Username:** admin, **Password:** pwd, **Role:** ADMIN
* **Username:** user, **Password:** pwd, **Role:** USER

This was done using the InMemoryUserDetailsManager bean in the SecurityConfig class.

**2. Password Encoding**

Passwords were encoded using BCryptPasswordEncoder, as required by Spring Security.

**3. Access Restriction**

The /countries endpoint was configured to allow access **only to users with role USER**.

**4. Testing with curl**

**Success with correct user credentials:**

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

**Unauthorized with incorrect password:**

curl -u user:wrong http://localhost:8090/countries

**Forbidden with correct admin credentials but wrong role:**

curl -u admin:pwd http://localhost:8090/countries

**Base64 Authorization header demo:**

curl -v -u admin:pwd http://localhost:8090/countries

This prints the encoded string:

Authorization: Basic YWRtaW46cHdk

Decoded using [base64decode.net](https://www.base64decode.net/) yields:

admin:pwd

**5. Final Code**

**SecurityConfig.java**

package com.example.countries\_api;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

UserDetails user = User.withUsername("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.withUsername("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

**CountryController.java**

package com.example.countries\_api;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<String> getCountries() {

return List.of("India", "USA", "France", "Germany");

}

}

**6. Error Faced and Resolved**

**Problem:** Application failed to start with:

Web server failed to start. Port 8090 was already in use.

**Solution:**

1. Identify process using the port:
2. netstat -ano | findstr :8090
3. Kill the process:
4. taskkill /PID <pid> /F

**7. Limitations of Basic Authentication**

* Requires credentials in every request (stateless).
* Credentials are encoded in Base64 and can be easily decoded.
* Not secure unless combined with HTTPS.

**8. Future Enhancements**

* Add a new /admin endpoint secured with hasRole("ADMIN").
* Move to database-based authentication using Spring Data JPA.
* Implement JWT-based authentication for more secure and stateless communication.