**CSRF in Spring Security**

* CSRF stand for Cross-Site Request Forgery.
* When client send data to server through post method, CSRF indicate that client is valid, not an attacker.
* By default CSRF is enabled in Spring Security.

**Pom.xml**

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.thymeleaf.extras</groupId>

<artifactId>thymeleaf-extras-springsecurity5</artifactId>

</dependency>

**HomeController.java**

**package** com.deepsingh44.blogspot.controller;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

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

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

@Controller

**public** **class** HomeController {

@GetMapping("/")

**public** String welcome() {

**return** "index";

}

@PostMapping("/add")

**public** String add(String name, Model model) {

model.addAttribute("name", name);

**return** "home";

}

}

**resources/templates/index.html**

<form action=*"add"* method=*"post"*>

<p>

Name:<input type=*"text"* name=*"name"*>

</p>

<p>

<input type=*"submit"*>

</p>

</form>

**resources/templates/home.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.w3.org/1999/xhtml"*>

<p th:text=*"${name}"*></p>

Now if you want to send data from client to server then you need to do following things:

* Disable CSRF Token
* Pass CSRF Token

Let’s start with first approach Disable CSRF Token:

@PostMapping ("/add")

**Public** String add (String name, Model model) {

model.addAttribute ("name", name);

**Return** "home";

}

Add this class in and override configure (HttpSecurity http) that disable CSRF.

**SecurityConfig.java**

**Package** com.deepsingh44.blogspot.config;

**Import** org.springframework.context.annotation.Configuration;

**Import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**Import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**Import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

**Public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {

@Override

**Protected** **void** configure (HttpSecurity http) **throws** Exception {

http.csrf ().disable ();

}

}

Now you can send data but suppose that if someone creates the same form in desktop and file name anything with html extension like below:

**Hacker.html**

<form action="**http://localhost:9999/add**" method="post">

<p>

Name:<input type="text" name="name">

</p>

<p>

<input type="submit">

</p>

</form>

If this hacker send data so we can also send data so may it can save malicious data store in your database so how can you send Authorized data not any one can send data so in that case we have to use second way:

So firstly remove or comment below code from **SecurityConfig .java**.

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable();

}

Now after this both client and hacker also can send data so for protecting your request you need to create a CSRF token.

<!DOCTYPE html>

<html lang=*"en"* xmlns:th=*"http://www.w3.org/1999/xhtml"*>

<form action=*"add"* method=*"post"*>

<p>

Name:<input type=*"text"* name=*"name"*>

</p>

<p><input type=*"hidden"* th:name=*"${\_csrf.parameterName}"* th:value=*"${\_csrf.token}"*></p>

<p>

<input type=*"submit"*>

</p>

</form>

Now the code is Done.

We can do more like if you want to disable in a particular URL so you can disable CSRF.

Like below:

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

//http.csrf().disable();

http.csrf().ignoringAntMatchers("/users/\*\*");

}

If you want to create your own CSRF Token so you need to create a class that implements **CsrfTokenRepository** interface :

**package** com.deepsingh44.blogspot.token;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.security.web.csrf.CsrfToken;

**import** org.springframework.security.web.csrf.CsrfTokenRepository;

**import** org.springframework.security.web.csrf.DefaultCsrfToken;

**public** **class** CustomToken **implements** CsrfTokenRepository {

@Override

**public** CsrfToken generateToken(HttpServletRequest request) {

**return** **new** DefaultCsrfToken("X-CSRF-TOKEN","\_csrf", "deepsingh44");

}

@Override

**public** **void** saveToken(CsrfToken token, HttpServletRequest request, HttpServletResponse response) {

// **TODO** Auto-generated method stub

}

@Override

**public** CsrfToken loadToken(HttpServletRequest request) {

// **TODO** Auto-generated method stub

**return** **null**;

}

}

And you have to call from SecurityConfig.java class like below:

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().csrfTokenRepository(**new** CustomToken());

}

You can inspect html and get your custom CSRF Token Thanks.