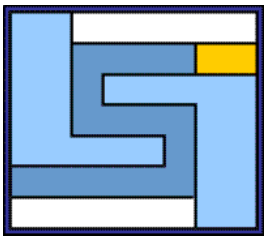


Item 7 (A++) - Report

Diseño y Pruebas

Grado de Ingeniería del Software

Curso 3

Armando Garrido Castro
Jorge Puente Zaro
Manuel Enrique Pérez Carmona
César García Pascual
Pablo Tabares García
Rafael Trujillo González

Fecha: 27 de febrero de 2018

Índice

1. Introducción 2
2. Componentes 2
3. Código Necesario 2
   3.1. layout.jsp 2
   3.2. security.xml 3
4. Test 3
5. Conclusión 4

1. Introducción

Dado que en este entregable hacemos énfasis en dar seguridad frente al “hacking”, hemos decidido ampliar el contenido incluyendo seguridad CSRF para evitar problemas con métodos CRUDS que supongan cambios en la capa de persistencia procedentes de otra dirección.

2. Componentes

La seguridad por tókens CSRF viene activada y configurada en versiones modernas de Spring, pero en la utilizada en el transcurso de esta asignatura usamos una versión más obsoleta, por lo que es necesario configurarlo manualmente.

Para ello simplemente necesitaremos edita el archivo “security.xml” de configuración e importar la funcionalidad en el código fuente de la “master page”, puesto que todas las vistas la extienden.

3. Código Necesario

3.1. layout.jsp

En este fichero añadimos al inicio las siguientes líneas:

```
20 <html>
21 <meta name="_csrf" content="${_csrf.token}"/>
22 <meta name="_csrf_header" content="${_csrf.headerName}"/>
23 <head>
24
25 <base
26     href="${pageContext.request.scheme}://${pageContext.request.serverName}:${pageConte
27
28 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
29 <meta name="viewport" content="width=device-width, initial-scale=1">
30
31 <link rel="shortcut icon" href="favicon.ico"/>
32
33
```

Y añadimos también el siguiente script para que se generen los tokens necesarios para que la página funcione:

```
109   </div>
110 </div>
111
112 </body>
113 <script>
114 var token = $("meta[name='_csrf']").attr("content");
115 var header = $("meta[name='_csrf_header']").attr("content");
116
117 $(document).ajaxSend(function(e, xhr, options) {
118     xhr.setRequestHeader(header, token);
119 });
120 </script>
121 </html>
```

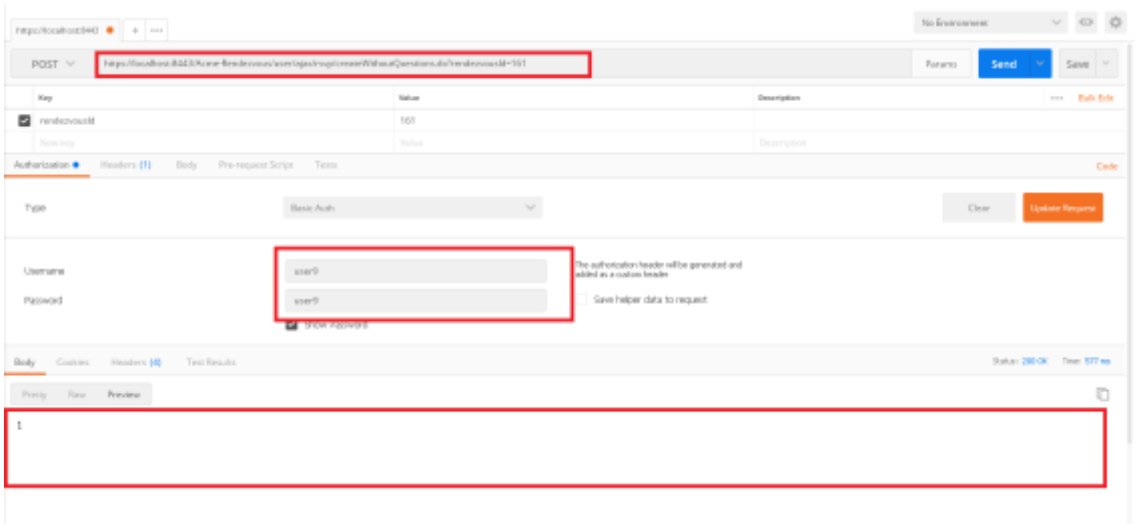
3.2. security.xml

Simplemente añadimos la siguiente línea dentro de la etiqueta <security:http>:

```
62              login-page="/security/login.do"
63              password-parameter="password"
64              username-parameter="username"
65              authentication-failure-url="/security/log
66
67          <security:logout
68              logout-success-url="/"
69              invalidate-session="true" />
70
71      <security:csrf/>
72      </security:http>
73
```

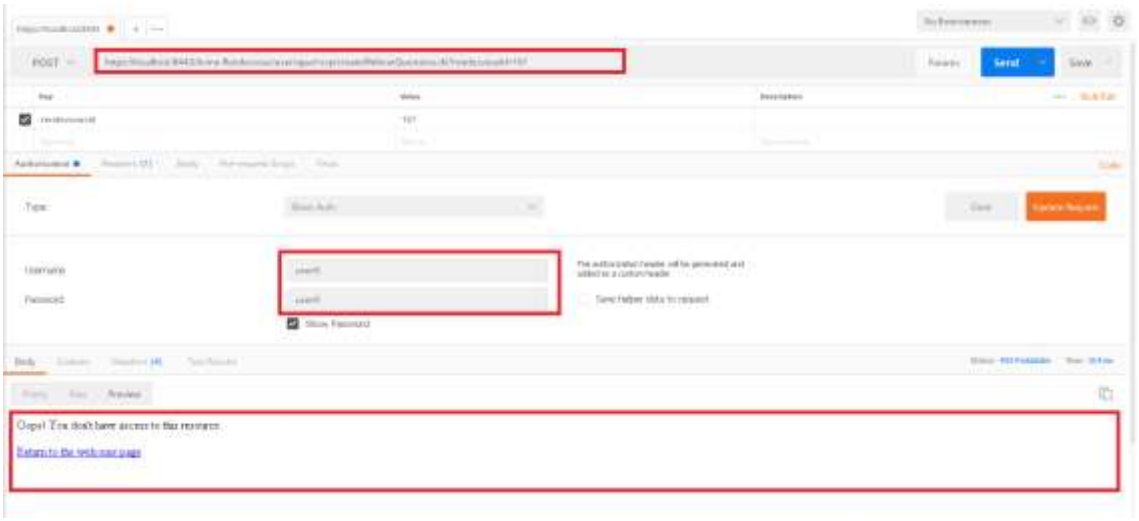
4. Test

Ahora con la ayuda de la herramienta POSTMAN comprobamos a realizar una sencilla operación POST, primero sin la seguridad CSRF activada:



Como podemos observar, nos permite realizar la operación puesto que la respuesta es un 1, señal de que el controlador de Ajax se ha ejecutado con éxito.

En cambio si activamos la seguridad y reintentamos la misma operación, nos devolverá un mensaje de que no tenemos acceso a dicha dirección:



5. Conclusión

Los ataques Cross-Site RequestForgery (CSRF), son muy comunes hoy en día aprovechando la situación en la que se encuentra internet (presencia masiva de publicidad intrusiva), esto incita a todo aquel con intenciones maliciosas a inyectar en algunos de los “banners” URLs que realicen operaciones que puedan perjudicar al usuario, por lo que es totalmente necesario implementar seguridad frente a estos ataques.