Example with File based authorization protection

1. Presentation

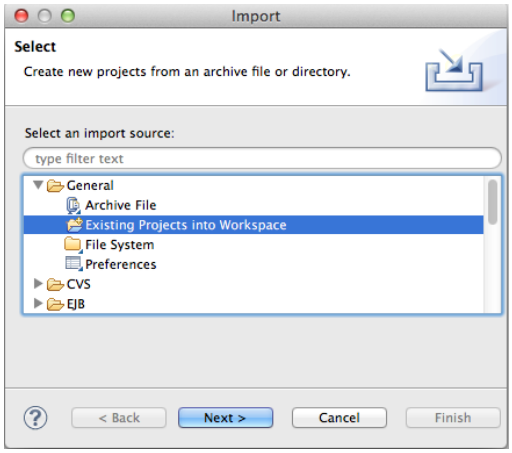
This first example is a game that consists to guess a number between 1 and 100. This application indicates only if the number proposed by the user is smaller or larger than the solution. There are two types of players, admins and ordinary users.

For admins accounts, the solution is indicated on the page. For the ordinary users, this information is not visible, they can not cheat.

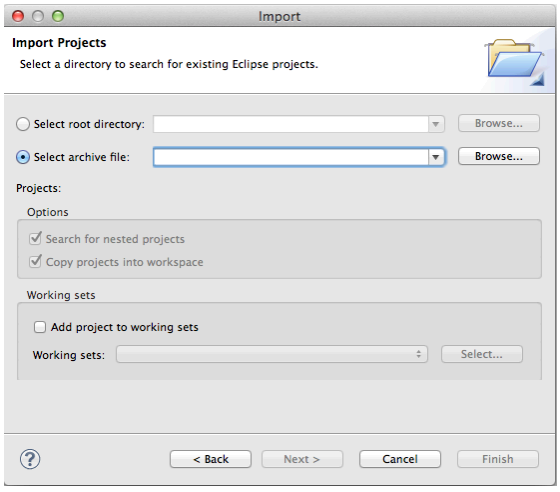
2. 7 steps installation

1. Download the guess_example eclipse project in the ‘example’ section.

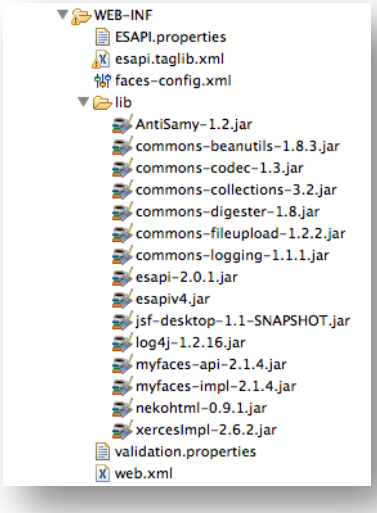
2. Import the project in your Eclipse environment by selecting File -> import.

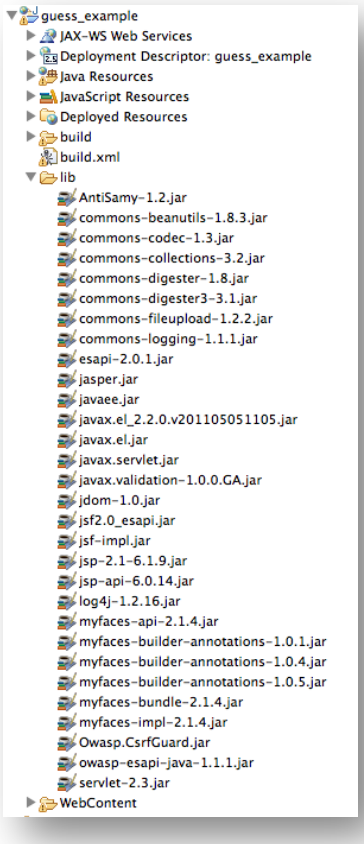


Choose ‘Existing Projects into Workspace’, then select the zip downloaded and press ‘Finish’.



3. All libraries specified in the ‘librairies’ section of the OWASP EJSF Project need to be present in the build path. Some librairies need to be copied in the lib directory located in the WEB-INF folder of the project. The libraries are normally already there.





4. Download the esapi_final.jar file in the ‘installation’ section and you need to copy this file in the WEB-INF/lib directory.

5. The next step is necessary for the login works! The users.txt file is available in the ‘Downloads’ section and need to be copied in this path:

- Mac and Linux users: \users\your_name\esapi\
- Windows users: C:\Documents and Settings\your_user\.esapi\

6. copy all config files in the WEB-INF folder (ESAPI.properties, esapi.taglib.xml, faces-config.xml, validation.properties)

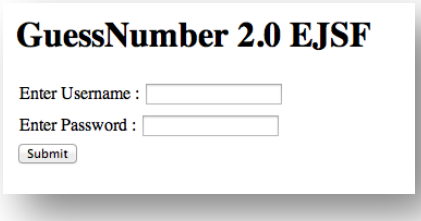
7. in the web.xml file you need to add some code :

```
<filter>
   <filter-name>CsrfFilter</filter-name>
   <filter-class>ch.security4web.esapi.csrf.OwaspCSRFSessionFilter</filter-class>
  </filter>
  <filter>
   <filter-name>AuthorizationFilter</filter-name>
   <filter-class>ch.security4web.esapi.authentication.EsapiAuthorizationFilter</filter-class>
  </filter>
<filter>

   <filter-name>POSTFilter</filter-name>
   <filter-class>ch.security4web.esapi.authentication.EsapiPOSTFilter</filter-class>
  </filter>
```

```
  <filter-mapping>
    <filter-name>POSTFilter</filter-name>
    <url-pattern>*.xhtml</url-pattern>
  </filter-mapping>

<context-param>
    <param-name>facelets.LIBRARIES</param-name>
    <param-value>/WEB-INF/esapi.taglib.xml</param-value>
  </context-param>
  <listener>
    <description>OwaspCSRFSessionListener</description>
    <listener-class>ch.security4web.esapi.csrf.OwaspCSRFSessionListener</listener-class>
  </listener>
```

3. A simple example

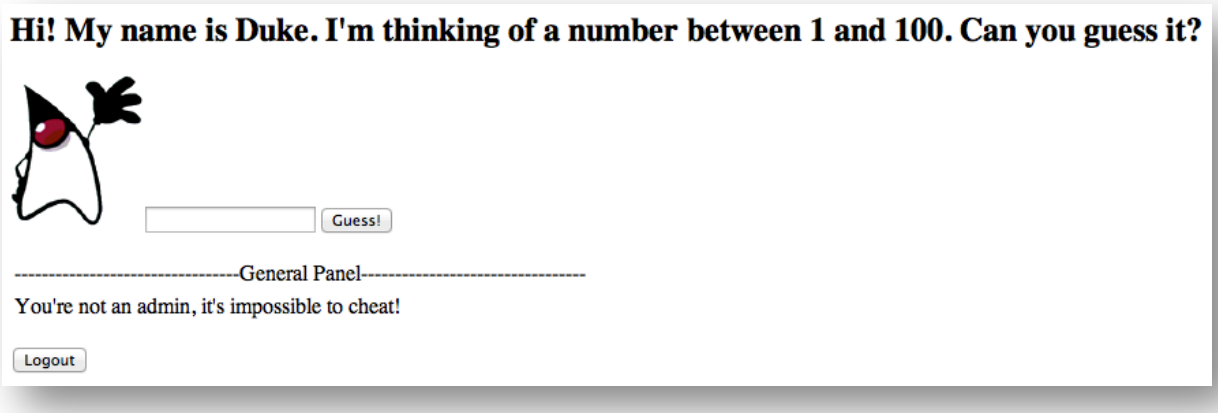
Before playing the game, you need to login on the main page. There are admins and users accounts.



3.1 Playing with a user account

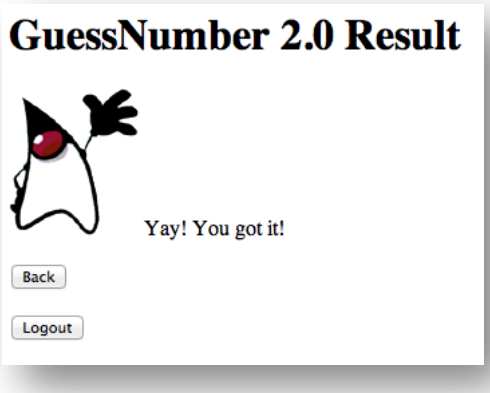
The first account is a user account.
Username: user
Password: Test12345

Please guess the number and press the 'Guess!' button.



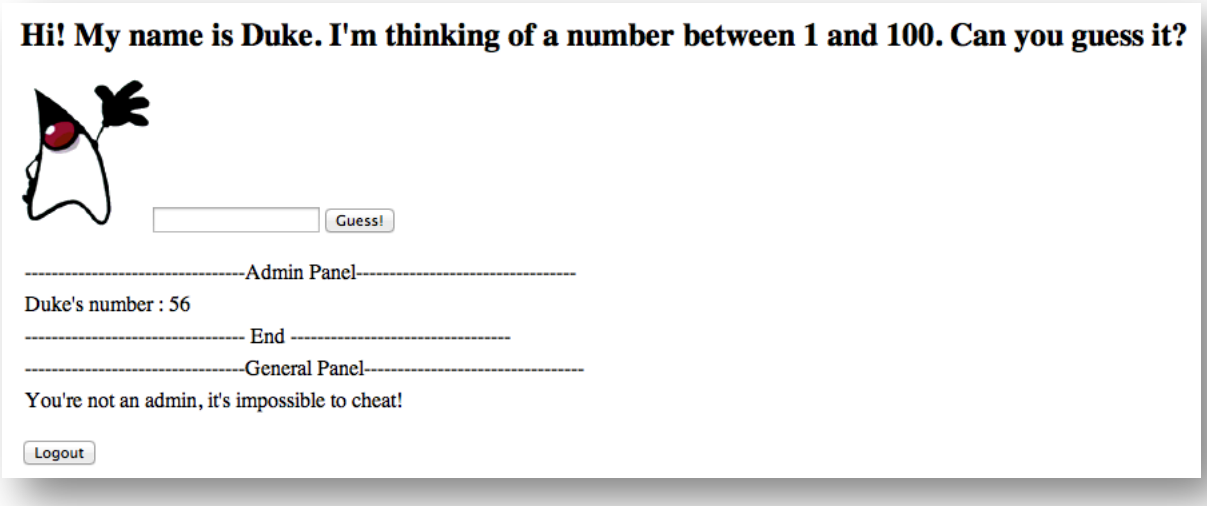
The number entered was 7, the game inform us that the number we such is larger!

We try with the 8 and we got it!



3.2 Playing with an admin account

With the admin account, the game is the same. We have just a little difference: we have the solution displayed in the 'Admin Panel'! (This panel is not visible by the user account). The admin user can see the general panel too.
Username: admin
Password: Test1234



4. Protection

An attacker (user player) can easily know the different node ID's available in the source code of the page.
In this example he knows the ID of the node who the result is displayed. He can now generate a POST request with this ID that allow to manually access this area of the page, yet, it is not allowed to view this result and he can cheat!

This solution works when the attacker generates the POST request, we verify that the ID entered in this request is on the blacklist or not. If the ID is available on the blacklist (that means that he certainly try to modify the tree node), the POST request should not be executed and an error will be generated.