Final Report
Inventory-managment
Security Testing
2019/2020

Francesco Penasa

January 15, 2020

1 Introduction

This is the report for the final project of the course Security Testing, 2019/2020. The project consisted to analyse the PHP code of a simple website running in local, find its XSS vulnerabilities and fix them. In the following sections we will show the procedures done and the results obtained.

2 Taint Analysis

In order to discover the XSS vulnerabilities present in the code, the tool ”Pixy” has been used. Pixy execute a Taint Analisys on the file given in input. A total of 76 possible XSS vulnerabilities have been discovered, all of them have been analysed in depth to understand their classification as false positive or true positive.

Table 1: Graphs produced by the Pixy execution.

Pixy file	**Description**	**Evaluation**
xss_changeBio.php_1_min	`changeBio.php line 22` displayed in an input tag that does not interpret the html code.	False Positive
xss_changePassword.php_1_min	`changePassword.php line 43` displayed in an input tag that does not interpret the html code.	False Positive
xss_changeUsername.php_1_min	`changeUsername.php line 23` displayed in an input tag that does not interpret the html code.	False Positive
xss_createBrand.php_1_min	`createBrand.php line 25` displayed in an input tag that does not interpret the html code.	False Positive
xss_createCategories.php_1_min	`createCategories.php line 25` displayed in an input tag that does not interpret the html code.	False Positive

xss_createOrder.php_1_min	`createOrder.php line 70` displayed in an input tag that does not interpret the html code.	False Positive
xss_createProduct.php_1_min	`createProduct.php line 43` displayed in an input tag that does not interpret the html code.	False Positive
xss_createUser.php_1_min	`createUser.php line 30` displayed in an input tag that does not interpret the html code.	False Positive
xss_dashboard.php_3_min	`dashboard.php line 75` the variable echoed can contain only a number, since it is create counting the rows of the a query.	False Positive
xss_dashboard.php_4_min	`dashboard.php line 87` the variable echoed can contain only a number, since it is create counting the rows of the a query.	False Positive
xss_dashboard.php_5_min	`dashboard.php line 101` the variable echoed can contain only a number, since it is create counting the rows of the a query.	False Positive
xss_dashboard.php_10_min	`dashboard.php line 153` display the username of the logged user without any sanitization.	**True Positive**
xss_dashboard.php_11_min	`dashboard.php line 154` the variable echoed can contain only a number, since it is create counting the rows of the a query.	False Positive
xss_editBrand.php_1_min	`editBrand.php line 25` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive
xss_editCategories.php_1_min	`editCategories.php line 25` same as the previous cell.	False Positive
xss_editOrder.php_1_min	`editOrder.php line 87` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive
xss_editPayment.php_1_min.dot	`editPayment.php line 31` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive
xss_editProduct.php_1_min.dot	`editProduct.php line 31` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive

xss_editProductImage.php_1_min	`editProductImage.php line 35` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive
xss_editUser.php_1_min.dot	`editUser.php line 27` the content of the echoed variable `$valid` is fixed. Only a specific message written by the developer can be displayed.	False Positive
xss_fetchBrand.php_1_min	`fetchBrand.php line 48` Display the brand name without any sanitization.	**True Positive**
xss_fetchCategories.php_1_min	`fetchCategories.php line 48` Display the category name without any sanitization.	**True Positive**
xss_fetchOrder.php_1_min	`fetchOrder.php line 71` Display the client name and the client contact without any sanitization.	**True Positive**
xss_fetchOrderData.php_1_min	`fetchOrderData.php line 19` As it is at the moment it is not a vulnerability, but it could become one if some other element of the query are used in the order page.	False Positive
xss_fetchProduct.php_1_min	`fetchProduct.php line 83` Display the product name, the brand name and the category name without any sanitization.	**True Positive**
xss_fetchProductData.php_1_min	`fetchProductData.php line 12` Display the product name without any proper sanitization.	**True Positive**
xss_fetchProductImage.php_1_min	`fetchProductImage.php line 13` displayed in an input tag that does not interpret the html code	False Positive
xss_fetchSelectedBrand.php_1_min	`fetchSelectedBrand.php line 16` displayed in an input tag that does not interpret the html code	False Positive
xss_fetchSelectedCategories.php_1_min	`fetchSelectedCategories.php line 16` displayed in an input tag that does not interpret the html code	False Positive
xss_fetchSelectedProduct.php_1_min	`fetchSelectedProduct.php line 16` displayed in an input tag that does not interpret the html code	False Positive
xss_fetchSelectedUser.php_1_min	`fetchSelectedUser.php line 16` displayed in an input tag that does not interpret the html code	False Positive
xss_fetchUser.php_1_min	`fetchUser.php line 47` Display the username without any sanitization.	**True Positive**
xss_getOrderReport.php_1_min	`getOrderReport.php line 49` Display the client name and client contact without any proper sanitization.	**True Positive**

xss_index.php_2_min	`index.php line 100` `$PHP_SELF` is a variable that returns the current script being executed.	False Positive
xss_orders.php_6_min.	`orders.php line 37` The variable echoed contains the ID which cannot be modified by the attacker.	False Positive
xss_orders.php_11_min	`orders.php line 111` Display the product name without any sanitization.	**True Positive**
xss_orders.php_20_min	`orders.php line 287` The date comes from a field of the DB that can contain only objects of date type.	False Positive
xss_orders.php_21_min	`orders.php line 293` displayed in an input tag that does not interpret the html code.	False Positive
xss_orders.php_22_min	`orders.php line 299` displayed in an input tag that does not interpret the html code.	False Positive
xss_orders.php_27_min	`orders.php line 345` Display the product name without any sanitization.	**True Positive**
xss_orders.php_29_min	`orders.php line 353` In order to be used in the web page rate has to be a number.	False Positive
xss_orders.php_31_min	`orders.php line 354` In order to be used in the web page rate has to be a number.	False Positive
xss_orders.php_32_min	`orders.php line 365` In order to be used in the web page quantity has to be a number.	False Positive
xss_orders.php_35_min	`orders.php line 380` In order to be used in the web page quantity has to be a number.	False Positive
xss_orders.php_37_min	`orders.php line 384` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_39_min	`orders.php line 385` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_41_min	`orders.php line 404` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_42_min	`orders.php line 405` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_43_min	`orders.php line 412` The variable echoed is calculated through not vulnerable parameters.	False Positive

xss_orders.php_44_min	`orders.php line 413` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_45_min	`orders.php line 419` In order to be used in the web page the variable echoed has to be a number.	False Positive
xss_orders.php_46_min	`orders.php line 425` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_47_min	`orders.php line 426` The hidden variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_50_min.dot	`orders.php line 432` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_51_min.dot	`orders.php line 433` The hidden variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_52_min.dot	`orders.php line 439` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_53_min.dot	`orders.php line 448` In order to be used in the web page the variable echoed has to be a number.	False Positive
xss_orders.php_54_min.dot	`orders.php line 454` The variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_55_min.dot	`orders.php line 455` The hidden variable echoed is calculated through not vulnerable parameters.	False Positive
xss_orders.php_64_min.dot	`orders.php line 513` The variable echoed $_GET[i] is the ID and cannot be modified by the attacker.	False Positive
xss_printOrder.php_1_min	`printOrder.php line 193` Display the client name, client contact and product name without any proper sanitization.	**True Positive**
xss_product.php_1_min	`product.php line 109` Display the brand name $row['1'] without any proper sanitization.	**True Positive**
xss_product.php_2_min	`product.php line 128` Display the category name without any proper sanitization.	**True Positive**
xss_product.php_3_min	`product.php line 267` Display the brand name without any proper sanitization.	**True Positive**

xss_product.php_4_min	`product.php line 286` Display the category name without any proper sanitization.	**True Positive**
xss_removeBrand.php_1_min	`removeBrand.php line 24` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_removeCategories.php_1_min	`removeCategories.php line 24` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_removeOrder.php_1_min	`removeOrder.php line 26` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_removeProduct.php_1_min	`removeProduct.php line 24` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_removeUser.php_1_min	`removeUser.php line 24` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_setting.php_1_min	`setting.php line 35` With some proper combination of characters it is possible to use the "change username" field for a XSS attack.	**True Positive**
xss_setting.php_2_min	`setting.php line 41` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_setting.php_3_min	`setting.php line 57` With some proper combination of characters it is possible to use the "change bio" field for a XSS attack.	**True Positive**
xss_setting.php_4_min	`setting.php line 63` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_setting.php_5_min	`setting.php line 99` The variable echoed contains an ID which cannot be modified by the attacker.	False Positive
xss_ssp.php_1_min	`ssp.php line 55` Server side process script.	False Positive

3 Security Testing

For each vulnerability evaluated as True Positive a test case has been written and a fix has been provided. To realize the test cases has been used the IDE Eclipe and the Selenium framework. In order to make the test cases cleaner are easibly readable we have used the Page Object method to organize the testing environment. All the vulnerabilities founded during the analysis are covered with a java file with the same name and every page of the website has its own file. There are two vulnerabilities that couldn't be tested properly: the first, `xss_printOrder.php_1_min` which have its own test file that requires some specific libraries to execute properly. While the second, `xss_getOrderReport.php_1_min` could not be tested since `report.php` does not work properly.

4 Fixes

In this section are displayed the fixes added to the code, to remove the XSS vulnerabilities. The last fix showed is one extra fix that has been added to the table in order to fix a XSS vulnerability that was not signaled by Pixy.

1. **dashboard.php**
   Vulnerable:
   ```
   153 echo ($orderResult[’username]’]);
   ```
   Fixed:
   ```
   153 echo (htmlentities($orderResult[’username]’]));
   ```

2. **fetchBrand.php**
   Vulnerable:
   ```
   48 $output[’data’][] = array($row[1]), $activeBrands, $button);
   ```
   Fixed:
   ```
   48 $output[’data’][] = array(htmlentities($row[1])), $activeBrands, $button);
   ```

3. **fetchCategories.php**
   Vulnerable:
   ```
   48 $output[’data’][] = array($row[1]), $activeBrands, $button);
   ```
   Fixed:
   ```
   48 $output[’data’][] = array(htmlentities($row[1])), $activeBrands, $button);
   ```

4. **fetchOrder.php**
   Vulnerable:
   ```
   50 $output[’data’][] = array($x, $row[1]), $row[2], $row[3], $itemCountRow, $paymentStatus,
   $button);
   ```
   Fixed:
   ```
   50 $output[’data’][] = array($x, $row[1]), htmlentities($row[2]), htmlentities($row[3]),
   $itemCountRow, $paymentStatus, $button);
   ```

5. **fetchProduct.php**
   Vulnerable:
   ```
   59 $output[’data’][] = array($productImage, $row[1], $row[6], $row[5], $brand, $category,
   $active,$button);
   ```
   Fixed:
   ```
   59 $output[’data’][] = array($productImage, htmlentities($row[1]), htmlentities($row[6]),
   $row[5], htmlentities($brand), htmlentities($category), $active,$button);
   ```

6. **fetchProductData.php**
   Vulnerable:
   ```
   13 echo json_encode($data);
   ```
   Fixed:
   ```
   13 $data[0][1] = htmlentities($data[0][1]);
   14 echo json_encode($data);
   ```

7. **fetchUser.php**
   Vulnerable:
   ```
   35 $output[’data’][] = array($username, $button);
   ```
   Fixed:
   ```
   35 $output[’data’][] = array(htmlentities($username), $button);
   ```

8. **getOrderReport.php**
   Vulnerable:
   ```
   31 $table .= ’<tr> <td><center>’.$result[’order_date’].’</center></td>
   <td><center>’.$result[’client_name’].’</center></td>
   <td><center>’.$result[’client_contact’].’</center></td>
   <td><center>’.$result[’grand_total’].’</center></td> </tr>’;
   ```
   Fixed:
   ```
   31 $table .= ’<tr> <td><center>’.$result[’order_date’].’</center></td>
   <td><center>’.htmlentities($result[’client_name’]).’</center></td>
   <td><center>’.htmlentities($result[’client_contact’]).’</center></td>
   <td><center>’.$result[’grand_total’].’</center></td> </tr>’;
   ```

9. **orders.php**
   Vulnerable:
   ```
   111 echo "<option value=’".$row[’product_id’]."’
   id=’changeProduct".$row[’product_id’]."’>".($row[’product_name’])."</option>";
   ```
   Fixed:
   ```
   111 echo "<option value=’".$row[’product_id’]."’
   id=’changeProduct".$row[’product_id’]."’>".htmlentities($row[’product_name’])."</option>";
   ```

10. **orders.php**
    Vulnerable:
    ```
    345 echo "<option value=’".$row[’product_id’]."’
    id=’changeProduct".$row[’product_id’]."’ ".$selected." >"
    .($row[’product_name’])."</option>";
    ```
    Fixed:
    ```
    345 echo "<option value=’".$row[’product_id’]."’
    id=’changeProduct".$row[’product_id’]."’ ".$selected." >"
    .htmlentities($row[’product_name’])."</option>";
    ```

11. **printOrder.php**
    Vulnerable:
    ```
    13 $clinetName = $orderData[1];
    14 $clinetContact = $orderData[2];
    ```

```
136 <td style="border-left:  1px solid black;height:  27px;">'.$row[4].'</td>
```
Fixed:
```
13 $clinetName = htmlentities($orderData[1]);
14 $clinetContact = htmlentities($orderData[2]);
136 <td style="border-left:  1px solid black;height:  27px;">'.htmlentities($row[4]).'</td>
```

12. **product.php**
    Vulnerable:
    ```
    109 echo "<option value='".$row[0]."'>".  $row[1]."</option>";
    ```
    Fixed:
    ```
    109 echo "<option value='".$row[0]."'>".  htmlentities($row[1])."</option>";
    ```

13. **product.php**
    Vulnerable:
    ```
    128 echo "<option value='".$row[0]."'>".$row[1]."</option>";
    ```
    Fixed:
    ```
    128 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
    ```

14. **product.php**
    Vulnerable:
    ```
    267 echo "<option value='".$row[0]."'>".$row[1]."</option>";
    ```
    Fixed:
    ```
    267 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
    ```

15. **product.php**
    Vulnerable:
    ```
    286 echo "<option value='".$row[0]."'>".$row[1]."</option>";
    ```
    Fixed:
    ```
    286 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
    ```

16. **setting.php**
    Vulnerable:
    ```
    35 <input type="text" class="form-control" id="username" name="username"
    placeholder="Usename" value="<?php echo ($result['username']); ?>"/>
    ```
    Fixed:
    ```
    35 <input type="text" class="form-control" id="username" name="username"
    placeholder="Usename" value="<?php echo (htmlentities($result['username'])); ?>"/>
    ```

17. **setting.php**
    Vulnerable:
    ```
    57 <input type="text" class="form-control" id="bio" name="bio" placeholder="Bio" value="<?php
    echo ($result['bio']); ?>"/>
    ```

    Fixed:
    ```
    57 <input type="text" class="form-control" id="bio" name="bio" placeholder="Bio" value="<?php
    echo (htmlentities($result['bio'])); ?>"/>
    ```

18. **dashboard.php**
    Vulnerable:
    ```
    50 echo ($_SESSION['username']);
    57 echo ($_SESSION['username']);
    ```
    Fixed:
    ```
    50 echo (htmlentities($_SESSION['username']));
    57 echo (htmlentities($_SESSION['username']));
    ```

5 Conclusion

On the 76 possible vulnerabilities we have found 17 True Positive and one vulnerability that was not signaled by Pixy. The test cases done and their fixes cover only xss vulnerabilities, for all the others type of vulnerabilities a deeper analysis has to be done.