Security Testing Project Final Report

Carlo Fanciulli 198793

Introduction

This is the final report for the security testing examination of the 13th June 2020.
This project consists in analyzing the php code of an ecommerce platform called Inventory-Management-System. Then, the Pixy tool was used to detect XSS vulnerabilities and classify them True Positive or False Positive. For True Positive cases are written automated test cases using the Selenium tool to assert the presence of the vulnerabilities. Finally, the vulnerabilities have been fixed and using the already used automated test case it is asserted that the fixes are effective. The testing has been implemented using the IDE Eclipse, JUnit and Selenium.

Taint Analysis

As required by the project specification, the Taint Analysis was performed with the Pixy tool, which tries to detect two major types of web application XSS vulnerabilities and SQL injection.
In this case, the tool has been found only the XSS vulnerabilities. In fact, the following table shows all the vulnerabilities found by the tool, and they have been analysed and described in detail.

Pixy file	**Description**	**Result**
xss_changeBio.php_1	*file*: changeBio.php - *line*: 22 There is an echo of an array without any tainted variable	False Positive
xss_changePassword.php_1	*file*: changePassword.php - *line*: 43 There is an echo of an array without any tainted variable	False Positive
xss_changeUsername.php_1	*file*: changeUsername.php - *line*: 23 There is an echo of an array without any tainted variable	False Positive
xss_createBrand.php_1	*file*: createBrand.php - *line*: 25 There is an echo of an array without any tainted variable	False Positive
xss_createCategories.php_1	*file*: createCategories.php - *line*: 25 There is an echo of an array without any tainted variable	False Positive
xss_createOrder.php_1	*file*: createOrder.php - *line*: 70 There is an echo of an array without any tainted variable	False Positive
xss_createProduct.php_1	*file*: createProduct.php - *line*: 43 There is an echo of an array without any tainted variable	False Positive
xss_createUser.php_1	*file*: createUser.php - *line*: 30	False Positive

	There is an echo of an array without any tainted variable	
xss_dashboard.php_3	*file*: dashboard.php - *line*: 75 The variable is creating counting the rows of a sql table, so the echo can contain only a number	False Positive
xss_dashboard.php_4	*file*: dashboard.php - *line*: 87 The variable is creating counting the rows of a sql table, so the echo can contain only a number	False Positive
xss_dashboard.php_5	*file*: dashboard.php - *line*: 101 The variable is creating counting the rows of a sql table, so the echo can contain only a number	False Positive
xss_dashboard.php_10	*file*: dashboard.php - *line*: 153 The username of the logged user is displayed without any sanitization	**True Positive**
xss_dashboard.php_11	*file*: dashboard.php - *line*: 154 The variable is creating counting the rows of a sql table, so the echo can contain only a number	False Positive
xss_editBrand.php_1	*file*: editBrand.php - *line*: 25 There is an echo of an array without any tainted variable	False Positive
xss_editCategories.php_1	*file*: editCategories.php - *line*: 25 There is an echo of an array without any tainted variable	False Positive
xss_editOrder.php_1	*file*: editOrder.php - *line*: 87 There is an echo of an array without any tainted variable	False Positive
xss_editPayment.php_1	*file*: editPayment.php - *line*: 31 There is an echo of an array without any tainted variable	False Positive
xss_editProduct.php_1	*file*: editProduct.php - *line*: 31 There is an echo of an array without any tainted variable	False Positive
xss_editProductImage.php_1	*file*: editProductImage.php - *line*: 35 There is an echo of an array without any tainted variable	False Positive
xss_editUser.php_1	*file*: editUser.php - *line*: 27 There is an echo of an array without any tainted variable	False Positive
xss_fetchBrand.php_1	*file*: fetchBrand.php - *line*: 48 There is an echo which displays the brand name without any sanitization	**True Positive**
xss_fetchCategories.php_1	*file*: fetchCategories.php - *line*: 48 There is an echo which displays the category name without any sanitization	**True Positive**
xss_fetchOrder.php_1	*file*: fetchOrder.php - *line*: 71 There is an echo which displays the client contact without any sanitization	**True Positive**

xss_fetchOrderData.php_1	*file*: fetchOrderData.php - *line*: 19 There is an echo of an array without any tainted variable	False Positive
xss_fetchProduct.php_1	*file*: fetchProduct.php - *line*: 83 There is an echo which displays the product name, the brand name and category name without any sanitization	**True Positive**
xss_fetchProductData.php_1	*file*: fetchProductData.php - *line*: 12 There is an echo which displays the product name without any sanitization	**True Positive**
xss_fetchProductImageUrl.php_1	*file*: fetchProductImageUrl.php - *line*: 13 Product image is selected from the database which is not a potential risk since it is a file	False Positive
xss_fetchSelectedBrand.php_1	*file*: fetchSelectedBrand.php - *line*: 16 There is an echo of an array without any tainted variable	False Positive
xss_fetchSelectedCategories.php_1	*file*: fetchSelectedCategories.php - *line*: 16 There is an echo of an array without any tainted variable	False Positive
xss_fetchSelectedProduct.php_1	*file*: fetchSelectedProduct.php - *line*: 16 There is an echo of an array without any tainted variable	False Positive
xss_fetchSelectedUser.php_1	*file*: fetchSelectedUser.php - *line*: 16 There is an echo of an array without any tainted variable	False Positive
xss_fetchUser.php_1	*file*: fetchUser.php - *line*: 47 There is an echo which display the username without any sanitization	**True Positive**
xss_getOrderReport.php_1	*file*: getOrderReport.php - *line*: 49 There is an echo which display informations without any sanitization	**True Positive**
xss_index.php_2	*file*: index.php - *line*: 100 $PHP SELF is a variable that returns the current script being executed	False Positive
xss_orders.php_6	*file*: orders.php - *line*: 37 There is an echo of GET variable `i`, which can be exploited by the user	**True Positive**
xss_orders.php_11	*file*: orders.php - *line*: 111 There is an echo which displays the product name without any sanitization	**True Positive**
xss_orders.php_20	*file*: orders.php - *line*: 287 The variable contains only objects of Date type	False Positive
xss_orders.php_21	*file*: orders.php - *line*: 293 There is an echo which displays the client name without any sanitization	**True Positive**
xss_orders.php_22	*file*: orders.php - *line*: 299 There is an echo which displays the client contact without any sanitization	**True Positive**
xss_orders.php_27	*file*: orders.php - *line*: 345 There is an echo which displays the product name without any sanitization	**True Positive**

xss_orders.php_29	*file*: orders.php - *line*: 353 Rate is updated through javascript when user click on the product	False Positive
xss_orders.php_31	*file*: orders.php - *line*: 354 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_32	*file*: orders.php - *line*: 365 Quantity is displayed only if greater than 0, so an implicit cast is done	False Positive
xss_orders.php_35	*file*: orders.php - *line*: 380 Quantity is displayed only if greater than 0, so an implicit cast is done	False Positive
xss_orders.php_37	*file*: orders.php - *line*: 383 The input total can’t be manipulated by the user	False Positive
xss_orders.php_39	*file*: orders.php - *line*: 385 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_41	*file*: orders.php - *line*: 404 The input subtotal can’t be manipulated by the user	False Positive
xss_orders.php_42	*file*: orders.php - *line*: 405 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_43	*file*: orders.php - *line*: 412 The input totalAmount can’t be manipulated by the user	False Positive
xss_orders.php_44	*file*: orders.php - *line*: 413 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_45	*file*: orders.php - *line*: 419 The input discount is editable by the user	**True Positive**
xss_orders.php_46	*file*: orders.php - *line*: 425 The input grandTotal can’t be manipulated by the user	False Positive
xss_orders.php_47	*file*: orders.php - *line*: 426 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_50	*file*: orders.php - *line*: 432 The input vat can’t be manipulated by the user	False Positive
xss_orders.php_51	*file*: orders.php - *line*: 433 There is an echo of a hidden variable without vulnerable parameters	False Positive
xss_orders.php_52	*file*: orders.php - *line*: 439 The input gstn is editable by the user	**True Positive**
xss_orders.php_53	*file*: orders.php - *line*: 448 The input paid is editable by the user	**True Positive**
xss_orders.php_54	*file*: orders.php - *line*: 454 The input due can’t be manipulated by the user	False Positive
xss_orders.php_55	*file*: orders.php - *line*: 455	False Positive

	There is an echo of a hidden variable without vulnerable parameters	
xss_orders.php_64	*file*: orders.php - *line*: 513 There is an echo of GET variable \`i\`, which can be exploited by the user	**True Positive**
xss_printOrder.php_1	*file*: printOrder.php - *line*: 193 There is an echo which displays the client and contact name and product name without any sanitization	**True Positive**
xss_product.php_1	*file*: product.php - *line*: 109 There is an echo which displays the brand name without any sanitization	**True Positive**
xss_product.php_2	*file*: product.php - *line*: 128 There is an echo which displays the category name without any sanitization	**True Positive**
xss_product.php_3	*file*: product.php - *line*: 267 There is an echo which displays the brand name without any sanitization	**True Positive**
xss_product.php_4	*file*: product.php - *line*: 286 There is an echo which displays the category name without any sanitization	**True Positive**
xss_removeBrand.php_1	*file*: removeBrand.php - *line*: 24 There is an echo of an array without any tainted variable	False Positive
xss_removeCategories.php_1	*file*: removeCategories.php - *line*: 24 There is an echo of an array without any tainted variable	False Positive
xss_removeOrder.php_1	*file*: removeOrder.php - *line*: 26 There is an echo of an array without any tainted variable	False Positive
xss_removeProduct.php_1	*file*: removeProduct.php - *line*: 24 There is an echo of an array without any tainted variable	False Positive
xss_removeUser.php_1	*file*: removeUser.php - *line*: 24 There is an echo of an array without any tainted variable	False Positive
xss_setting.php_1	*file*: setting.php - *line*: 35 There is an echo of the username which can be exploited by the user	**True Positive**
xss_setting.php_2	*file*: setting.php - *line*: 41 There is an echo of an id variable which cannot be modified by the attacker	False Positive
xss_setting.php_3	*file*: setting.php - *line*: 57 There is an echo of the user bio which can be exploited by the user	**True Positive**
xss_setting.php_4	*file*: setting.php - *line*: 63 There is an echo of an id variable which cannot be modified by the attacker	False Positive
xss_setting.php_5	*file*: setting.php - *line*: 99 There is an echo of an id variable which cannot be modified by the attacker	False Positive

Fixes

This chapter will list the fixes added to the code to remove XSS vulnerabilities.

dashboard.php

Vulnerability:
```
 echo $_SESSION['username'];
```
Fix:
```
 echo htmlentities($_SESSION['username']);
```

Vulnerability:
```
 echo ($orderResult['username]']);
```
Fix:
```
 echo (htmlentities($orderResult['username]']));
```

fetchBrand.php

Vulnerability:
```
 $output['data'][] = array($row[1]), $activeBrands, $button);
```
Fix:
```
 $output['data'][] = array(htmlentities($row[1])), $activeBrands, $button);
```

fetchCategories.php

Vulnerability:
```
 $output['data'][] = array($row[1]), $activeCategories, $button);
```
Fix:
```
 $output['data'][] = array(htmlentities($row[1])), $activeCategories, $button);
```

fetchOrder.php

Vulnerability:
```
 $output['data'][] = array($x, $row[1]), $row[2], $row[3], $itemCountRow, $paymentStatus,
$button);
```
Fix:
```
 $output['data'][] = array($x, $row[1]), htmlentities($row[2]), htmlentities($row[3]),
$itemCountRow, $paymentStatus, $button);
```

fetchProduct.php

Vulnerability:
```
 $output['data'][] = array($productImage, $row[1], $row[6], $row[5], $brand, $category,
$active,$button);
```
Fix:
```
 $output['data'][] = array($productImage, htmlentities($row[1]), floatval($row[6]),
$row[5], htmlentities($brand), htmlentities($category), $active,$button);
```

fetchProductData.php

Vulnerability:
```
 echo json encode($data);
```
Fix:
```
 foreach($data as $datum){$datum[1] = htmlentities($datum[1])}
 echo json encode($data);
```

fetchUser.php

Vulnerability:

```
 $output['data'][] = array($username, $button);
```

Fix:

```
 $output['data'][] = array(htmlentities($username), $button);
```

orders.php

Vulnerability:

```
 echo "Edit Order " . $_GET['i'];
```

Fix:

```
 echo "Edit Order " . intval($_GET['i']);
```

Vulnerability:

```
id='changeProduct".$row['product_id']."'>".$row['product_name']."</option>";id='changePro
duct".$row['product id']."' ".$selected." >"
.($row['product name'])."</option>";
```

Fix:

```
id='changeProduct".$row['product_id']."'>".htmlentities($row['product_name'])."</option>";
```

Vulnerability:

```
<input type="text" class="form-control" id="orderDate" name="orderDate"
autocomplete="off" value="<?php echo $data[1] ?>" />
<input type="text" class="form-control" id="clientName" name="clientName"
placeholder="Client Name" autocomplete="off" value="<?php echo $data[2] ?>" />
<input type="text" class="form-control" id="clientContact" name="clientContact"
placeholder="Contact Number" autocomplete="off" value="<?php echo $data[3] ?>" />
```

Fix:

```
<input type="text" class="form-control" id="orderDate" name="orderDate"
autocomplete="off" value="<?php echo htmlentities($data[1]) ?>" />
<input type="text" class="form-control" id="clientName" name="clientName"
placeholder="Client Name" autocomplete="off" value="<?php echo htmlentities($data[2])
?>" />
<input type="text" class="form-control" id="clientContact" name="clientContact"
placeholder="Contact Number" autocomplete="off" value="<?php echo
htmlentities($data[3]) ?>" />
```

Vulnerability:

```
 echo "<option value='".$row['product_id']."' id='changeProduct".$row['product_id']."'
".$selected." >".$row['product_name']."</option>";
```

Fix:

```
 echo "<option value='".$row['product_id']."' id='changeProduct".$row['product_id']."'
".$selected." >".htmlentities($row['product_name'])."</option>";
```

Vulnerability:

```
<input type="hidden" name="orderId" id="orderId" value="<?php echo $_GET['i']; ?>" />
```

Fix:

```
 <input type="hidden" name="orderId" id="orderId" value="<?php echo intval($_GET['i']);
?>" />
```

printOrder.php

Vulnerability:
```
 $clientName = $orderData[1];
 $clientContact = $orderData[2];
 $gstn = $orderData[11];
```
Fix:
```
 $clientName = htmlentities($orderData[1]);
 $clientContact = htmlentities($orderData[2]);
 $gstn = htmlentities($orderData[11]);
```

product.php

Vulnerability:
```
 echo "<option value='".$row[0]."'>". $row[1]."</option>";
```
Fix:
```
 echo "<option value='".$row[0]."'>". htmlentities($row[1])."</option>";
```

Vulnerability:
```
 echo "<option value='".$row[0]."'>".$row[1]."</option>";
```
Fix:
```
 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
```

Vulnerability:
```
 echo "<option value='".$row[0]."'>".$row[1]."</option>";
```
Fix:
```
 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
```

Vulnerability:
```
 echo "<option value='".$row[0]."'>".$row[1]."</option>";
```
Fix:
```
 echo "<option value='".$row[0]."'>".htmlentities($row[1])."</option>";
```

setting.php

Vulnerability:
```
 <input type="text" class="form-control" id="username" name="username"
placeholder="Usename" value="<?php echo ($result['username']); ?>"/>
```
Fix:
```
 <input type="text" class="form-control" id="username" name="username"
placeholder="Usename" value="<?php echo (htmlentities($result['username'])); ?>"/>
```

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

Conclusion

This analysis allowed to discover 76 vulnerabilities of which only 24 were true positive and then fixed. It is right to remember that this analysis found only the xss vulnerabilities because this was the purpose of the project and that a more detailed analysis could have found other internal issues.