Ain Shams University
Faculty of Engineering

Assignment 3

Stored XSS Attack On Mad Pharma

Under Supervision of

Dr. Ayman Bahaa

&

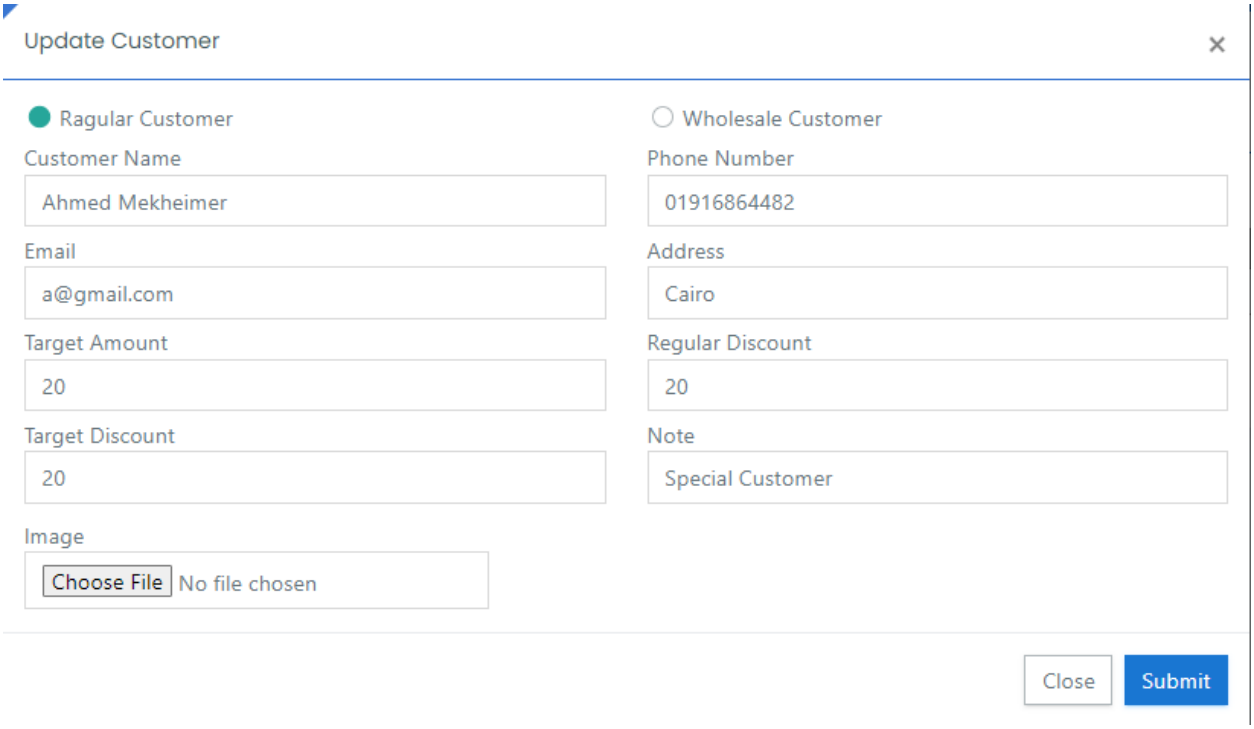
Eng. Amaal Khalid

Submitted By:

Ahmed Khaled Saad Ali Mekheimer
ID: 1809799

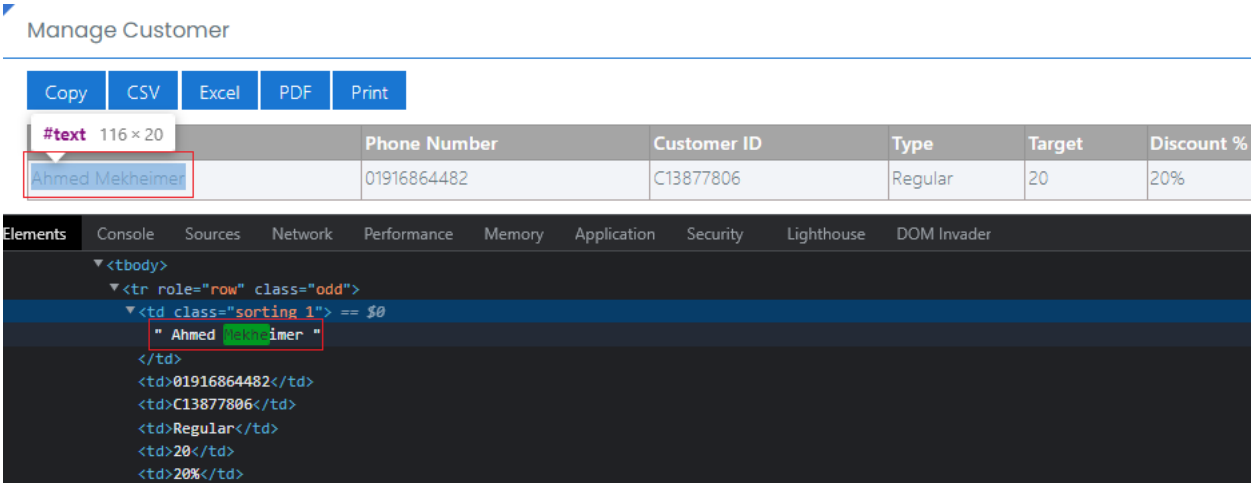
1. Stored XSS Vulnerability

In “Manage Customer” page, we will update one of the customer’s information. Then click Submit.

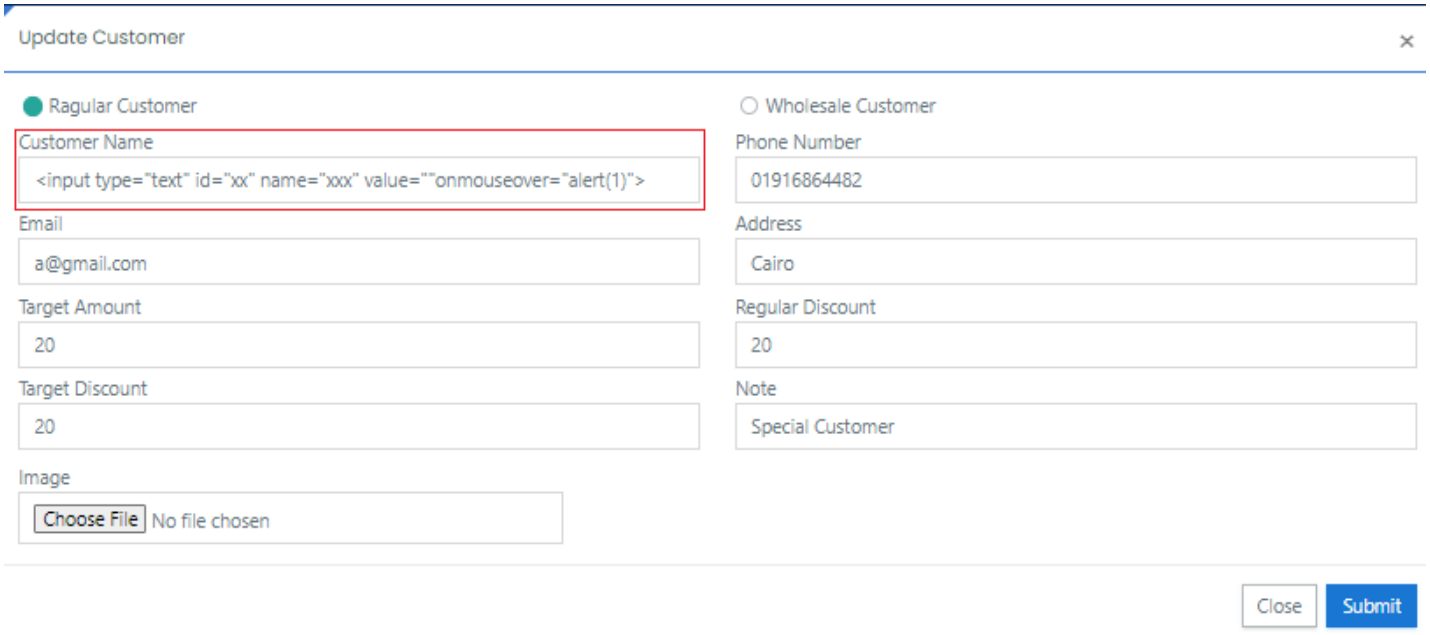


We will see we have multiple fields from what we entered that get stored in server and get displayed on the page. These fields are: Customer Name, Phone Number, Target and Target Discount.

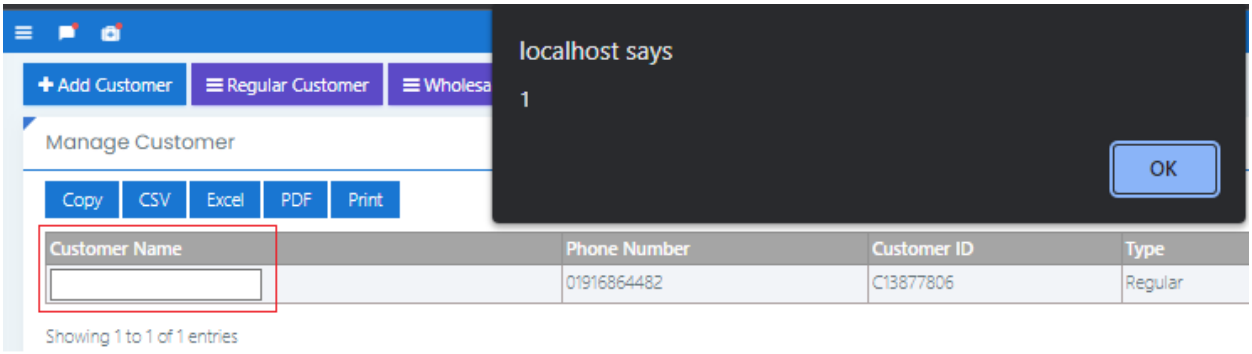
When we check “Page Source” of “Manage Customer”, we will notice that the information we entered are put on the server in HTML tags (<td>).



Let’s start with “Customer Name” field, to XSS attack it we can write an input tag to add a text field for example then make an Event Listener on it to trigger the alert() function.
We will write: <input type="text" id="xx" name="xxx" value=""onmouseover="alert(1)"> in the field below when Updating Customer’s name. Then Click Submit.



The input field is displayed in the “Customer Name” field and when we hover over it, we get an alert message. XSS attack is successful.



Here is a demo video showing the XSS attack flow.

https://drive.google.com/file/d/15AdTG0JwDpNhUkJhD0L2oaS_5z2fLO6T/view?usp=share_link

2. Fixing XSS Vulnerability

By checking the php file of this page which is “List_Customer.php”.

```
    <h4 class="m-b-0 text-white">Manage Customer </h4>

</div>

<div class="card-body">

    <div class="table-responsive ">

        <table id="myTable" class="display nowrap table table-hover table-s
            <thead>
                <tr>
                    <th>Customer Name</th>
                    <th>Phone Number</th>
                    <th>Customer ID</th>
                    <th>Type</th>
                    <th>Target</th>
                    <th>Discount %</th>
                    <th>Image </th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach($customerList as $value): ?>
                <tr>

                    <td><?php echo $value->c_name;?></td>
                    <td><?php echo $value->cus_contact; ?></td>
                    <td><?php echo $value->c_id; ?></td>
                    <td><?php echo $value->c_type; ?></td>
                    <td><?php echo $value->target_amount; ?></td>
                    <td><?php echo $value->regular_discount ."%" ?></td>
                    <td>
```

We will notice from the above snapshot that this is the table that displays the Customer’s information. In the red box, the table’s content is brought from the info that is on the database which we entered in “Update Customer”.

A common fix for such vulnerability is to HTML-encode the text entered in the field. HTML encoding means converting special characters, such as <, >, &, ", and ', into their corresponding HTML entities, so that they are treated as plain text by the browser, rather than as HTML code.

So, we shouldn't take data from "$value->c_name" directly in the page to be displayed it will get read as HTML code and cause the alert message.

Instead, we should HTML-Encode "$value->c_name" using "htmlspecialchars()", then pass it to the page to display it using "echo".

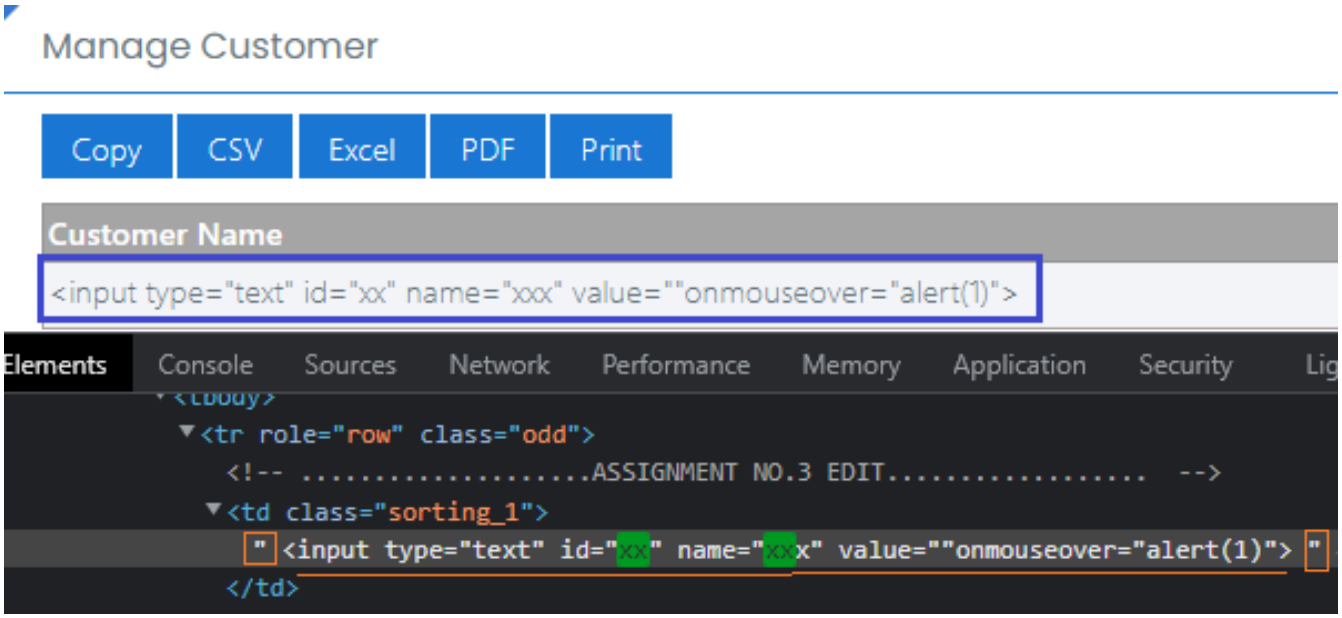
```
64                  <?php foreach($customerList as $value): ?>
65                   <tr>
66                              <!-- ..................ASSIGNMENT NO.3 EDIT.................  -->
67                       <td>
68                          <?php  //..................ASSIGNMENT NO.3 EDIT.................//
69                          $xss_name = htmlspecialchars($value->c_name, ENT_QUOTES, 'UTF-8'); //ASSIGNMENT NO.3 EDIT
70                          echo $xss_name;                                                    //ASSIGNMENT NO.3 EDIT
71                          ?>
72                       </td>
```

Note: put edited file in htdocs\mad-pharma master\application\views\backend

Now let's refresh the page and see if what script entered in the field still generates the input field and alert message or not.

We observe that now the page sees HTML code entered as plain text and sets it as Customer's name and no alert messages display when hovering on the field. XSS attack is countered. Vulnerability is fixed.

To see why our fix functioned, when we check page source, we observe that input tag was put as plain text (between double quotes) not as HTML code as shown below.



Here is a demo video showing the flow of the attack being fixed.

https://drive.google.com/file/d/1Kt9PUsPxWu1NEztrIGOAVzcZgt4kZFA5/view?usp=share_link