Milestone4

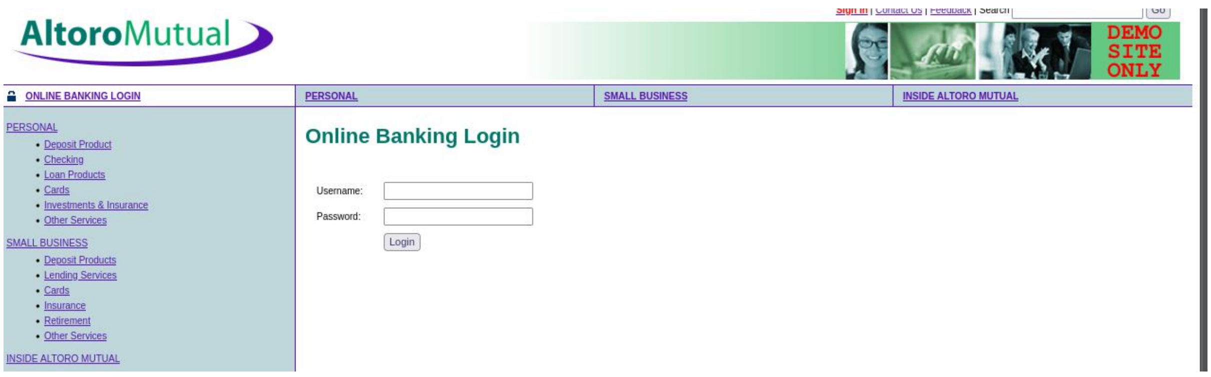
Pieter_Johannes_Swart

Table of Contents

Cloning the Altoro Mutual Sign-In Page

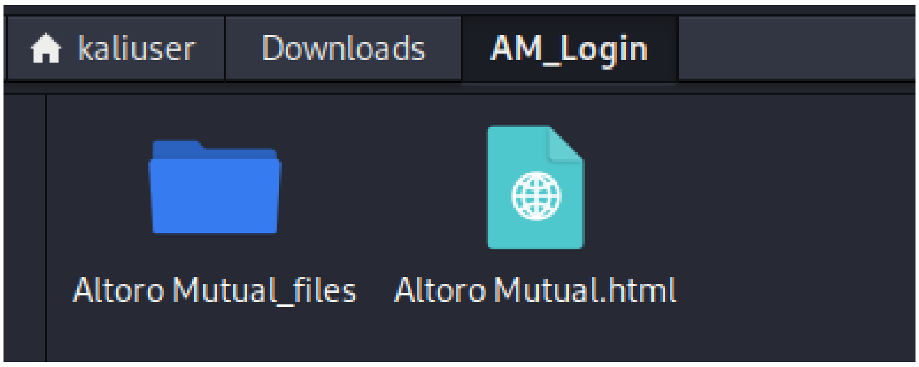
Step1: Open the Altoro Mutual website in your browser by visiting on kali:

http://altoromutual.com:8080/login.jsp

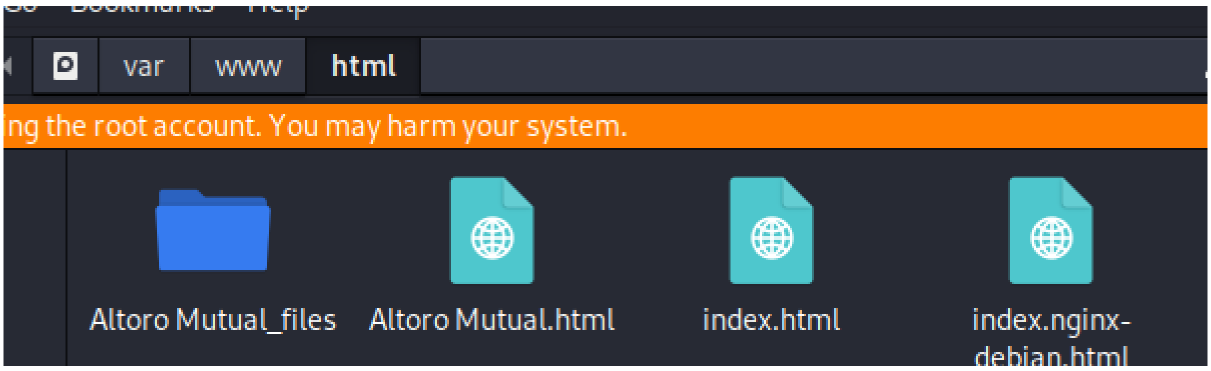


Go kali File manager and on path type:' /var/www/' => you will see a html Folder

Step2: Go back to "Altoro Mutual website" right click => click on "Save Page as..."=> create a Folder name "AM_Login" and Press "Enter => Go to file => go to downloads => dubble click on AM_Login.



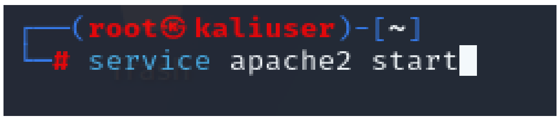
Copy the file and the .html file => on file manager, on the top left click on file => click on "Open as Root" =>go to the '/var/www/html/' => paste the two file there



Start apache2 and go to 127.0.0.1 on browser

Command:

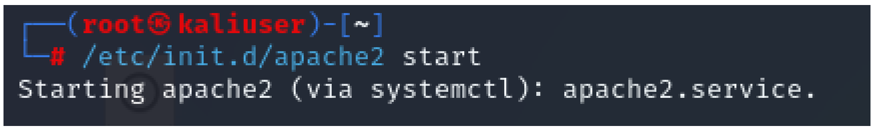
service apache2 start



```
┌──(root㉿kaliuser)-[~]
└─# service apache2 start
```

Then type the next command:

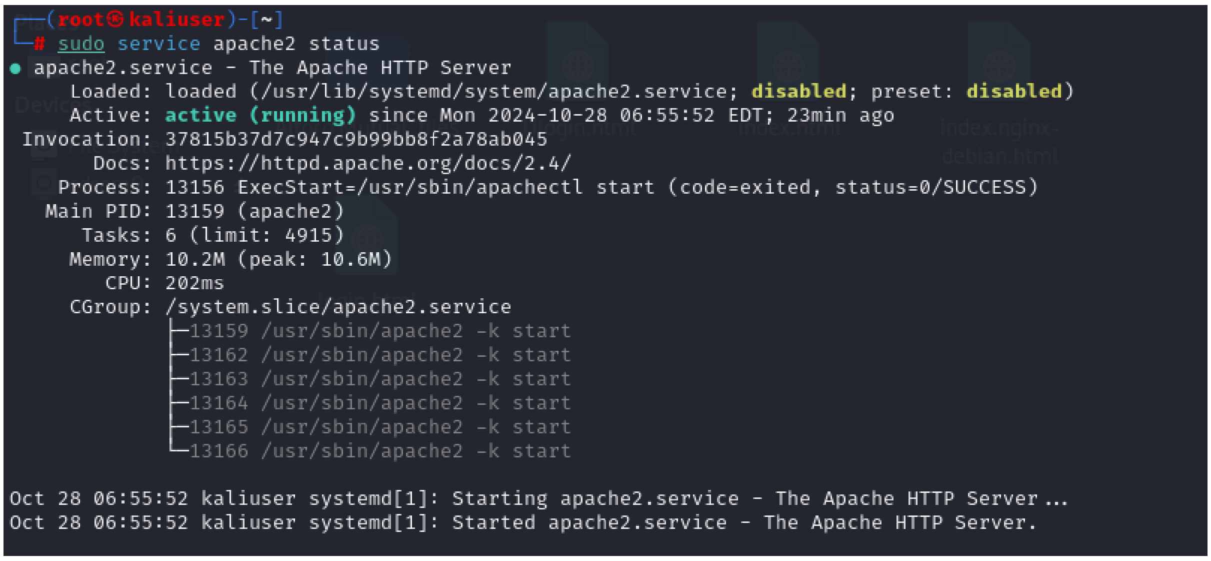
/etc/init.d/apache2 start



```
┌──(root㉿kaliuser)-[~]
└─# /etc/init.d/apache2 start
Starting apache2 (via systemctl): apache2.service.
```

If you want to see apache2 status, type this command:

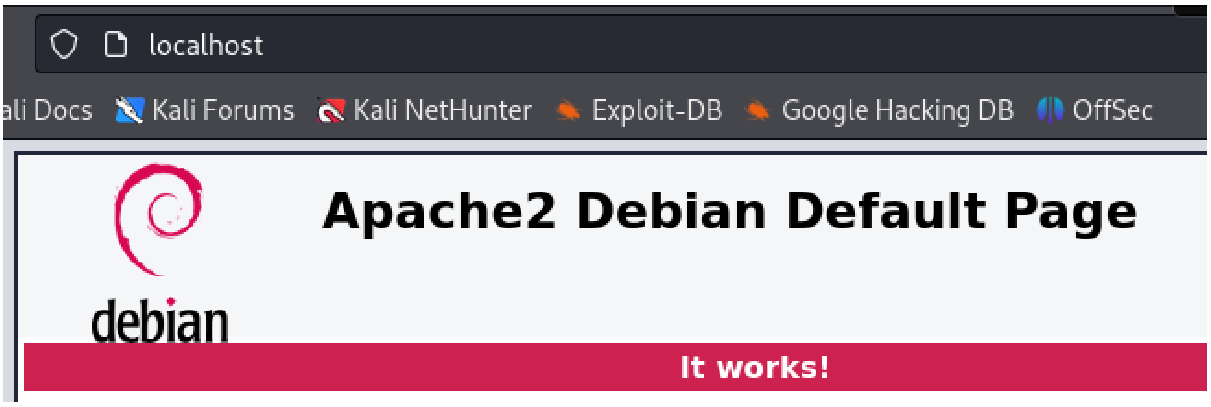
sudo service apache2 status



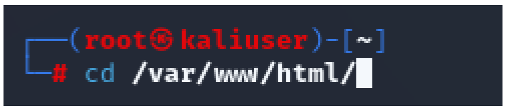
```
┌──(root㉿kaliuser)-[~]
└─# sudo service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
     Active: active (running) since Mon 2024-10-28 06:55:52 EDT; 23min ago
 Invocation: 37815b37d7c947c9b99bb8f2a78ab045
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 13156 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 13159 (apache2)
      Tasks: 6 (limit: 4915)
     Memory: 10.2M (peak: 10.6M)
        CPU: 202ms
     CGroup: /system.slice/apache2.service
             ├─13159 /usr/sbin/apache2 -k start
             ├─13162 /usr/sbin/apache2 -k start
             ├─13163 /usr/sbin/apache2 -k start
             ├─13164 /usr/sbin/apache2 -k start
             ├─13165 /usr/sbin/apache2 -k start
             └─13166 /usr/sbin/apache2 -k start

Oct 28 06:55:52 kaliuser systemd[1]: Starting apache2.service - The Apache HTTP Server...
Oct 28 06:55:52 kaliuser systemd[1]: Started apache2.service - The Apache HTTP Server.
```

Go to your browser and on the URL type: **http://localhost/**

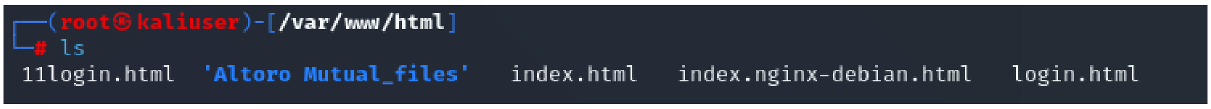


To make sure the website is running, on terminal you most go to the path by typing this command:



```
┌──(root㉿kaliuser)-[~]
└─# cd /var/www/html/
```

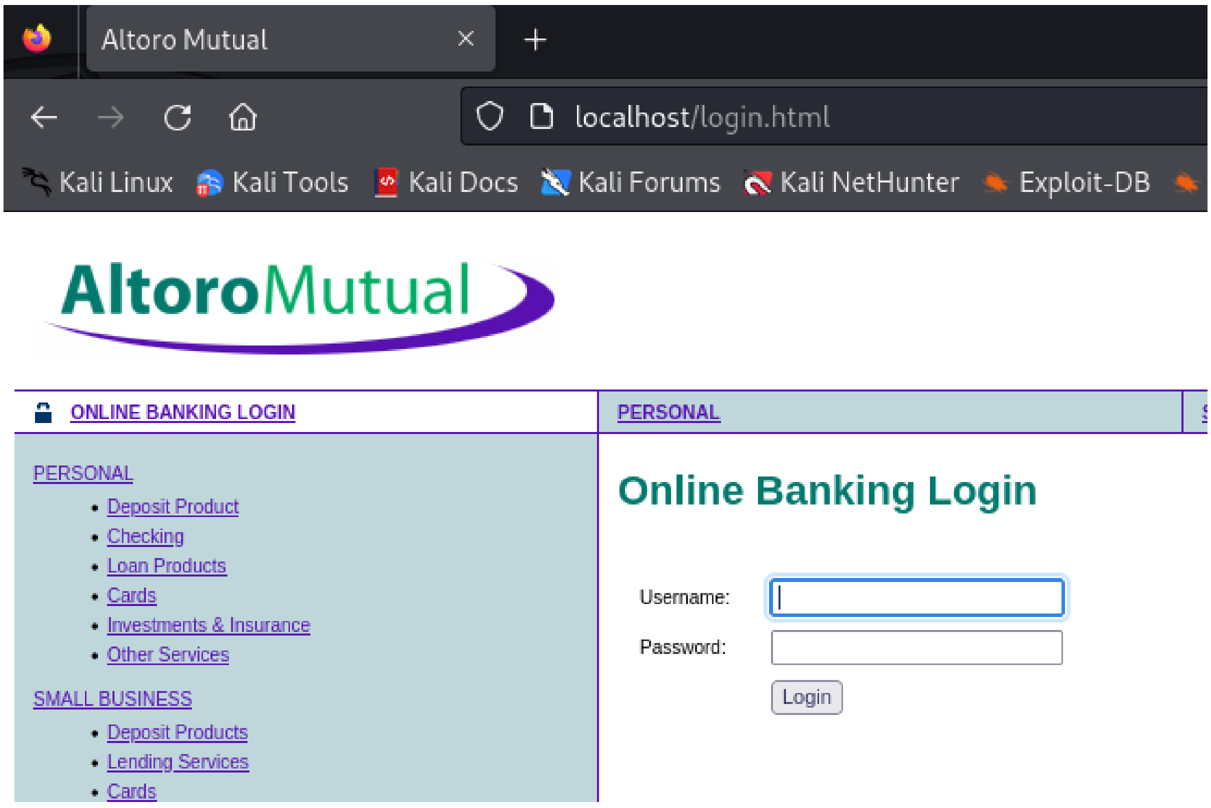
And then type: **ls**



```
┌──(root㉿kaliuser)-[/var/www/html]
└─# ls
11login.html  'Altoro Mutual_files'   index.html   index.nginx-debian.html   login.html
```

Go to your browser=> on the URL type **http://localhost/login.html**

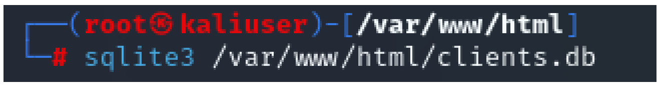
You will see the cloned page



Next step: Setting Up the Database with 5 Clients

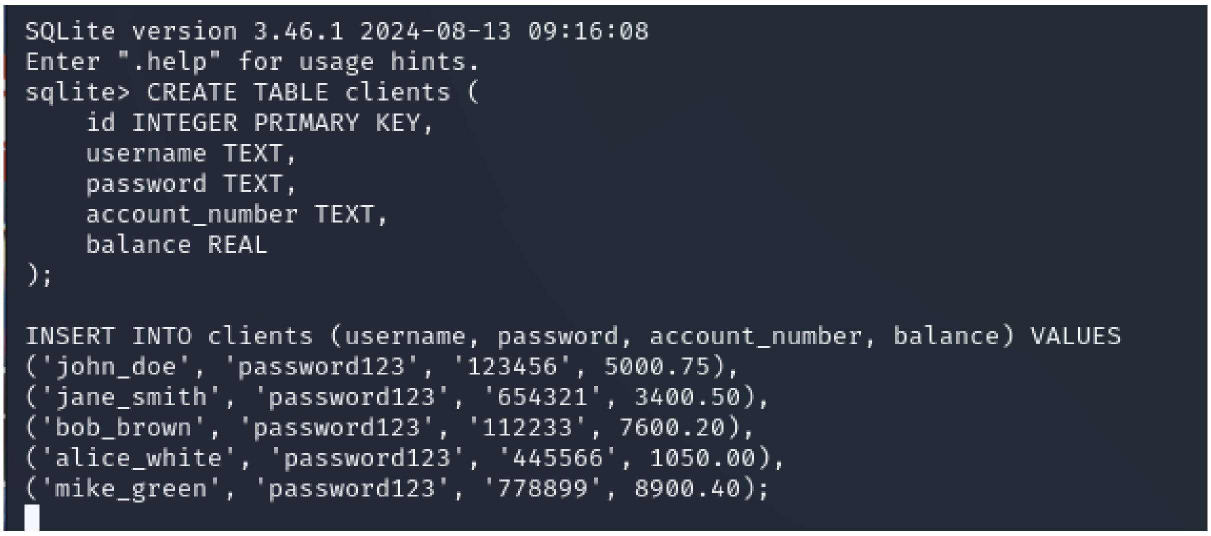
Use SQLite to create a simple database:

sqlite3 /var/www/html/clients.db



```
┌──(root㉿kaliuser)-[/var/www/html]
└─# sqlite3 /var/www/html/clients.db
```

Inside the SQLite prompt, create your table and insert 5 clients:



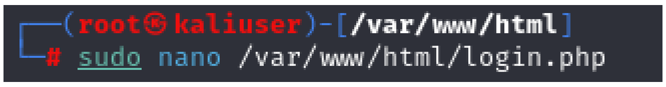
```
SQLite version 3.46.1 2024-08-13 09:16:08
Enter ".help" for usage hints.
sqlite> CREATE TABLE clients (
    id INTEGER PRIMARY KEY,
    username TEXT,
    password TEXT,
    account_number TEXT,
    balance REAL
);

INSERT INTO clients (username, password, account_number, balance) VALUES
('john_doe', 'password123', '123456', 5000.75),
('jane_smith', 'password123', '654321', 3400.50),
('bob_brown', 'password123', '112233', 7600.20),
('alice_white', 'password123', '445566', 1050.00),
('mike_green', 'password123', '778899', 8900.40);
```

Press inter => **Exit SQLite** by typing ".exit".

Next Step: **Link the Website with the Database**

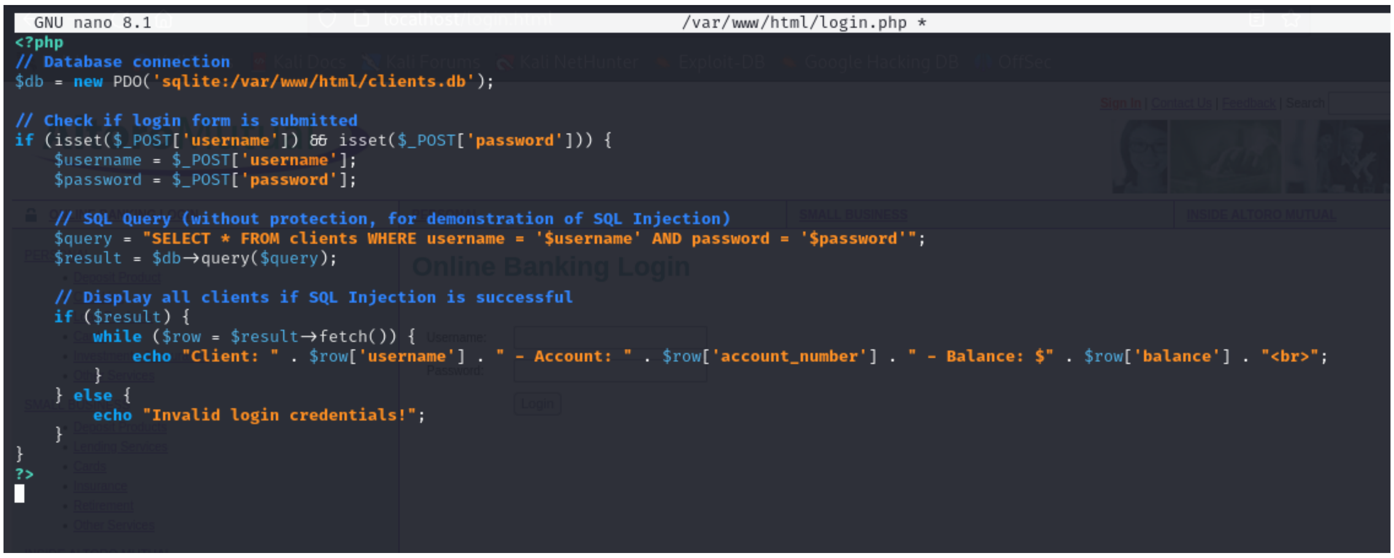
sudo nano /var/www/html/login.php



```
┌──(root㉿kaliuser)-[/var/www/html]
└─# sudo nano /var/www/html/login.php
```

Create a PHP Script for Login (login.php)

Type the the PHP script:



```
 GNU nano 8.1                                              /var/www/html/login.php *
<?php
// Database connection
$db = new PDO('sqlite:/var/www/html/clients.db');

// Check if login form is submitted
if (isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // SQL Query (without protection, for demonstration of SQL Injection)
    $query = "SELECT * FROM clients WHERE username = '$username' AND password = '$password'";
    $result = $db->query($query);

    // Display all clients if SQL Injection is successful
    if ($result) {
        while ($row = $result->fetch()) {
            echo "Client: " . $row['username'] . " - Account: " . $row['account_number'] . " - Balance: $" . $row['balance'] . "<br>";
        }
    } else {
        echo "Invalid login credentials!";
    }
}
?>
```

Demonstrate SQL Injection

What is a SQL Injection?

SQL Injection is a security flaw where a hacker can insert harmful SQL code into a query, allowing them to manipulate the database in ways not intended. This occurs as a result of mishandling of user inputs in SQL queries.

How SQL Injection Works:

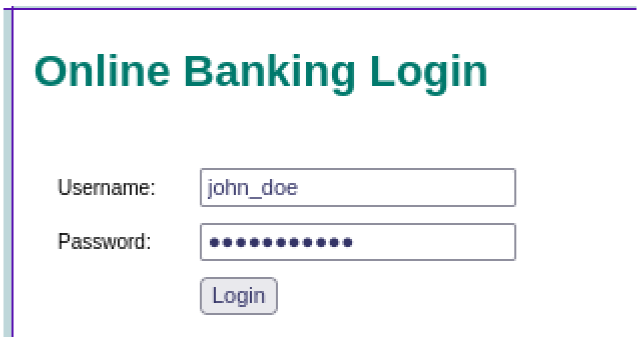
If user inputs are not adequately validated or sanitized, attackers can input SQL commands. These harmful inputs can deceive the server into performing unintended database tasks, such as accessing unauthorized data, changing information, or erasing entire databases.

Next Step:

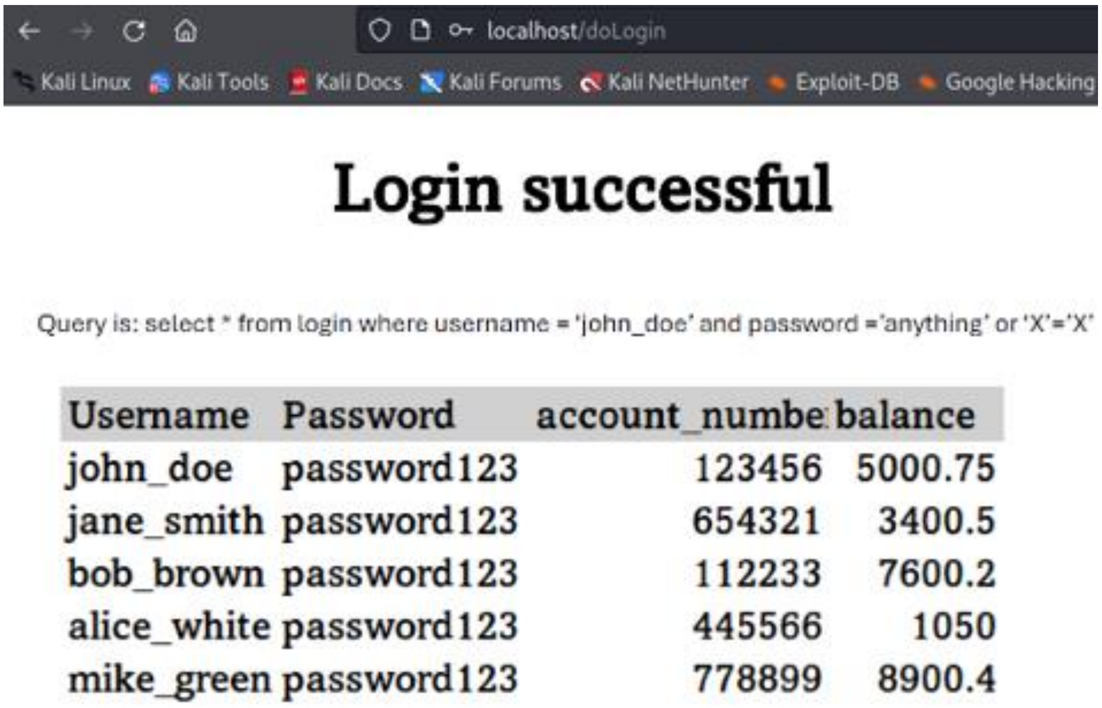
Lets test the website

- Go to http://localhost/login.html in your browser.
- Enter a simple username (e.g., john_doe) and an **SQL injection string** as the password: **'anything' or 'x'='x'**

  • '**anything'**: This is just a placeholder for any string value, it could be anything the attacker types.
  • **OR**: This is a logical operator that checks if either of the conditions on its left or right side are true.
  • **'x'='x'**: This is a condition that is **always true** because the string 'x' is always equal to itself.



If the PHP code is vulnerable, it should display all clients in the database, demonstrating successful SQL Injection.



Login successful

Query is: select * from login where username = 'john_doe' and password ='anything' or 'X'='X'

Username	Password	account_numbe	balance
john_doe	password123	123456	5000.75
jane_smith	password123	654321	3400.5
bob_brown	password123	112233	7600.2
alice_white	password123	445566	1050
mike_green	password123	778899	8900.4

---

demonstrate Cross-Site Scripting (XSS)

what is a Cross-Site Scripting?

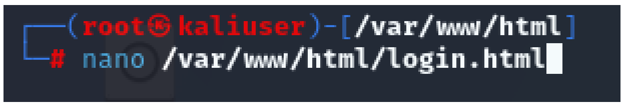
It is a security flaw that enables attackers to insert harmful scripts into webpages that are seen by other users. It happens when a program fails to validate or escape user input before incorporating it into the output. The injected code can run in the user's browser, resulting in various harmful activities such as harvesting user data, taking control of sessions, or vandalizing websites.

The goal of this step is to find an input field or URL parameter that takes user input and outputs it directly into the page without validation or sanitization.

Step1: **Modify the Cloned Login Page to Simulate XSS Vulnerability**

Open the login page HTML file you cloned earlier:

nano /var/www/html/login.html



```
┌──(root㉿kaliuser)-[/var/www/html]
└─# nano /var/www/html/login.html
```

Add a simple XSS vulnerability by directly displaying the user's input without sanitization, for example type XSS Vulnerability Code in the PHP:

xample of XSS Vulnerability Code:

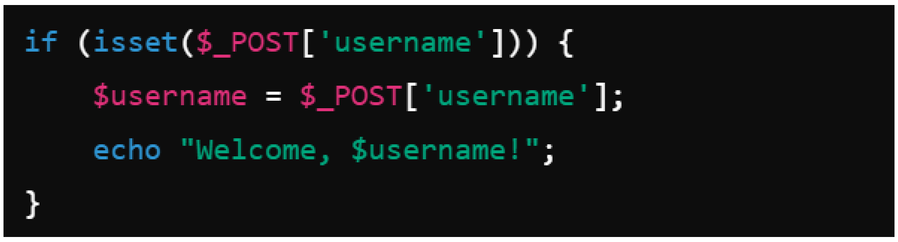
// PHP echoing unsanitized input (for XSS demonstration)

if (isset($_POST['username'])) {

$username = $_POST['username'];

echo "Welcome, $username!";

}



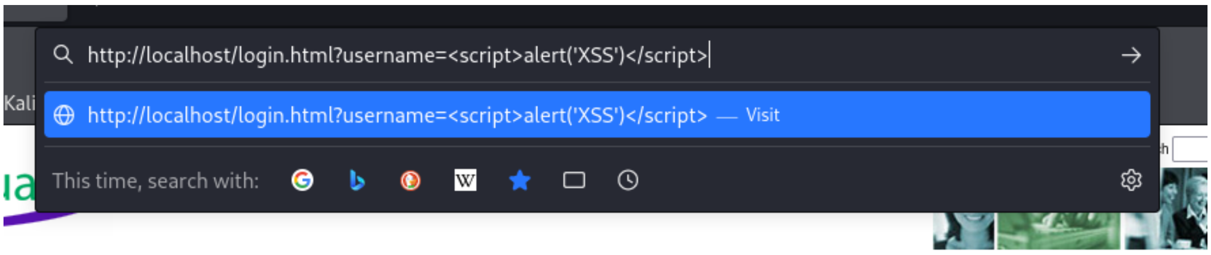
```
if (isset($_POST['username'])) {
    $username = $_POST['username'];
    echo "Welcome, $username!";
}
```

Save the file and exit

Next Step: Test for XSS vulnerability

Open a browser and go to your cloned login page => **Test the XSS vulnerability** by entering the following into the username field or by appending a query parameter to the URL like:

http://localhost/login.html?username=<script>alert('XSS')</script>

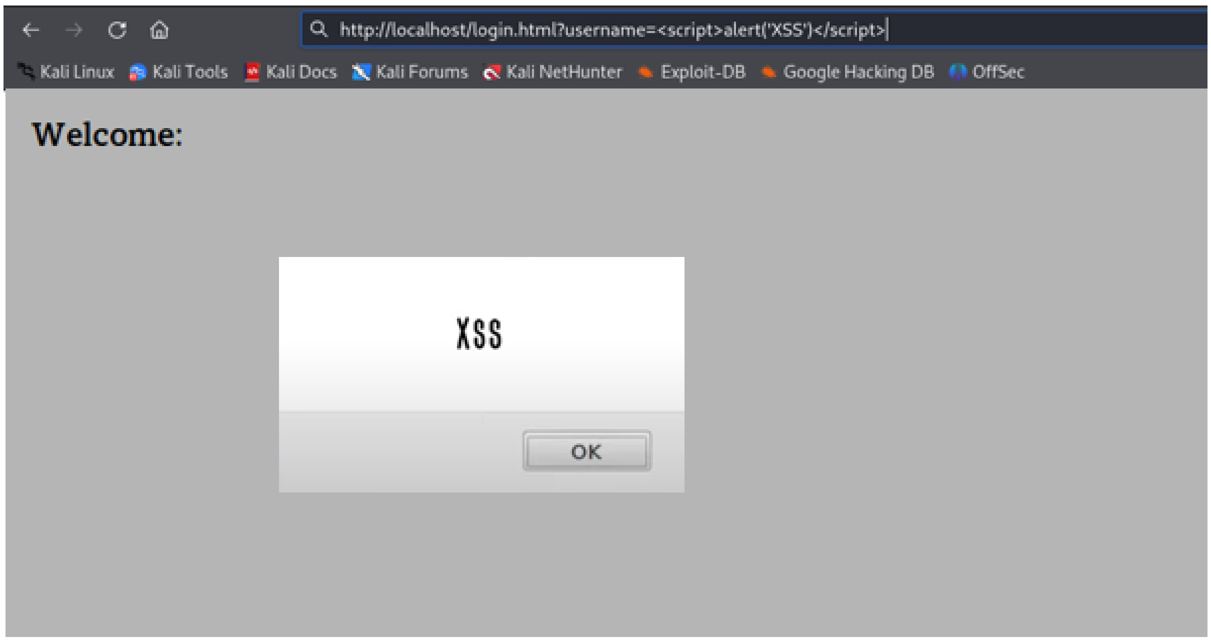


```
http://localhost/login.html?username=<script>alert('XSS')</script>
http://localhost/login.html?username=<script>alert('XSS')</script> — Visit
```

the URL Command:

- **http://localhost/login.html**: This part indicates that the user is accessing a page named login.html on their local server (localhost).
- **?username=**: This part indicates that a parameter named username is being passed to the server or the webpage. The start of the query string.
- **<script>alert('XSS')</script>**: This is the value of the username parameter, and it contains an HTML <script> tag with JavaScript code.

If the page is vulnerable, you should see a popup displaying the message XSS.



```
http://localhost/login.html?username=<script>alert('XSS')</script>
```

References

Baral, P., 2024. Cross Site Scripting (XSS) Exploitation. 7 Apr, pp. https://medium.com/@prashunbaral/cross-site-scripting-xss-exploitation-2f37c4b9010d.

Blog, I., 2018. Using SET tool kit to perform Website Cloning in Kali Linux. 28 Dec, pp. https://medium.com/@nancyjohn_95536/using-set-tool-kit-to-perform-website-cloning-in-kali-linux-67fa01c92af9.

DeVito, A., 2024. Blind SQL Injection: An Expert's Guide to Detect and Exploit. 11 July, pp. https://www.stationx.net/blind-sql-injection/.

G., B., 2018. Website cloning/phishing page. 26 May, pp. https://www.linkedin.com/pulse/website-cloningphishing-page-bhushan-ghode.

Garn, D., 2022. How to deploy an Apache web server quickly. 1 April, pp. https://www.redhat.com/en/blog/install-apache-web-server.

Kumar, V., 2023. How to Configure Apache Server in Kali Linux Guide for Beginners. 2 July, pp. https://www.cyberpratibha.com/blog/web-server-in-kali-linux/.

Sankar, S., 2023. SQL Injection using SQLMAP- Part 1.. 19 May, pp. https://medium.com/@Cyber_siva/sql-injection-using-sqlmap-77bb4c627e6a.

Unhnown, 2024. A Deep Dive Into Cross-Site Scripting and Its Significance. 13 Aug, pp. https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-cross-site-scripting-attack-and-how-to-prevent-it.