WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Bheemreddy Vikhyath Reddy

Email: bheemrvy@mail.uc.edu

Figure 1: Vikhyath's Headshot

Repository Information

Repository's URL: https://github.com/Vikhyath-Reddy/waph-bheemrvy

This is a private repository which is used to store all the codes related to course Topics in Computer Systems. The structure of this repository is as mentioned below.

Lab's overview

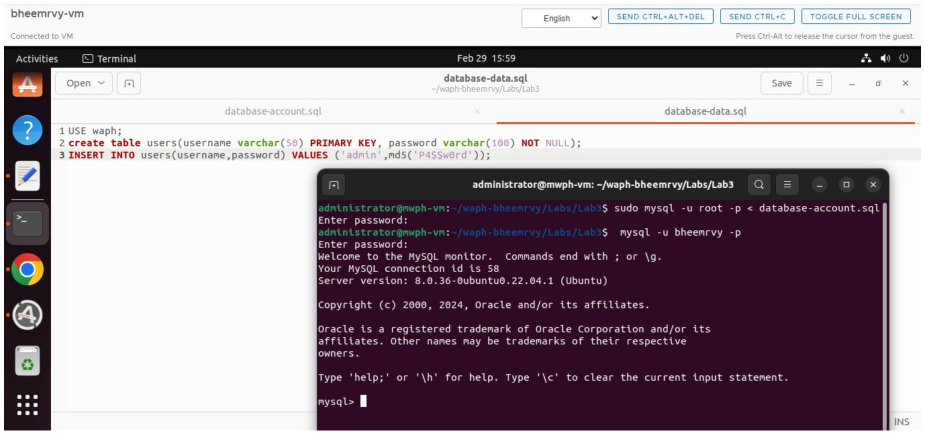
This laboratory exercise provides an instructional outline for developing a secure web application using PHP/MySQL. Initially, the database setup is performed in the terminal window, granting access to the database and establishing a connection with the root user. A new database is then created using the account setup command, aimed at facilitating user learning and permission management. Subsequently, a table is configured to enhance the functionality of the login page. The implementation of a login system utilizing PHP and MySQL is detailed, showcasing the deployment of the login page. To assess vulnerabilities, XSS and SQL Injection Attacks are simulated. Finally, measures to safeguard the login page are explored, including the utilization of SMTP requests to bolster security.

Link to Lab3 code : https://github.com/Vikhyath-Reddy/waph-bheemrvy/tree/main/Labs/Lab3

Database Setup and Management:

Installation of MySQL

The process begins with the installation of MySQL using the command `sudo apt-get install mysql-server -y`, followed by accessing MySQL with `sudo mysql -u root -p`. Subsequently, a new database is created and permissions are initiated through the `database.account.sql` command. Login credentials are then entered into a database table using MySQL, facilitating the setup of a new database account. The creation of a new table named Users is executed, with data insertion including hashed passwords. Verification of the table creation and data insertion is done with `show databases` and `SELECT * FROM users;` commands respectively. Throughout, SQL commands are utilized to define the structure of the Users table, specifying column data types and constraints. This step ensures the proper organization and security of the database, enhancing the functionality of the web application.



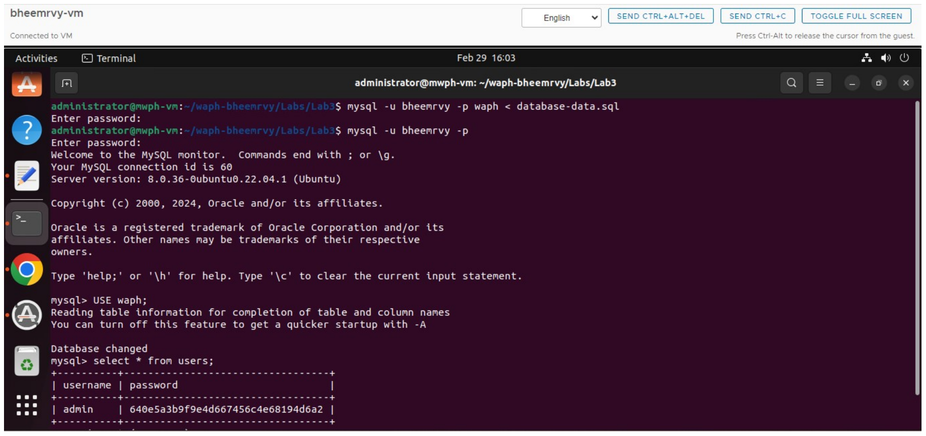
```
USE waph;
create table users(username varchar(50) PRIMARY KEY, password varchar(100) NOT NULL);
INSERT INTO users(username,password) VALUES ('admin',md5('P4$$w0rd'));
```

```
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab3$ sudo mysql -u root -p < database-account.sql
Enter password:
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab3$  mysql -u bheemrvy -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 58
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

Figure 2: Database Creation

A Simple Login System with PHP/MySQL

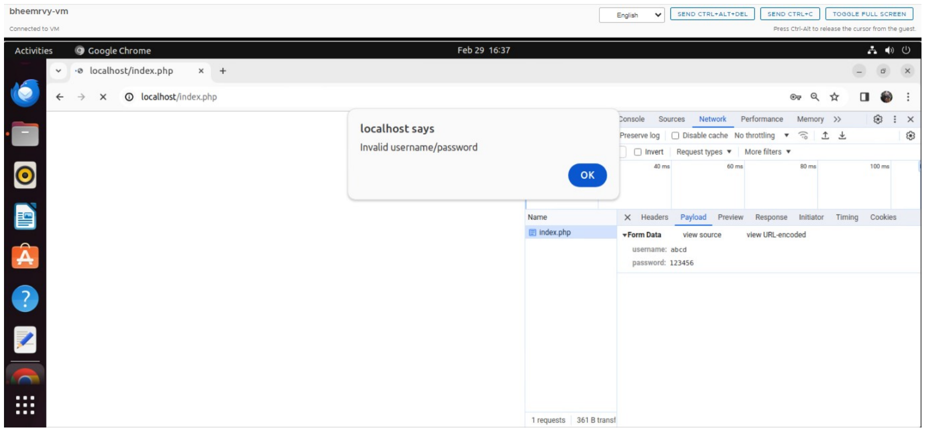
To set up the necessary PHP and MySQL drivers, begin by running `sudo apt-get install php-mysqli` and restart Apache using `sudo service apache2 restart`. Next, modify the `index.php` file as instructed by adding a `checklogin_mysql` function for database authentication, which verifies user credentials against the MySQL database. After making these changes, test and deploy the updated files (`index.php` and `form.php`) in the web server directory, ensuring accessibility by the web server. Finally, conduct login tests with both valid and invalid username/password combinations to confirm the functionality of the login system. This process ensures the proper installation of PHP and



```
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab3$ mysql -u bheemrvy -p waph < database-data.sql
Enter password:
administrator@mwph-vm:~/waph-bheemrvy/Labs/Lab3$ mysql -u bheemrvy -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 60
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE waph;
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> select * from users;
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| admin    | 640e5a3b9f9e4d667456c4e68194d6a2 |
+----------+----------------------------------+
```

Figure 3: Table Creation

MySQL drivers and validates the effectiveness of the login functionality within the web application.



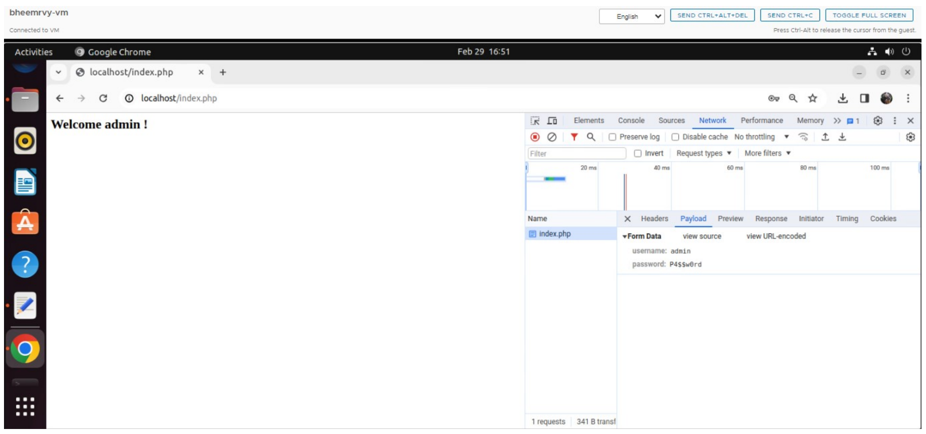
localhost says
Invalid username/password
OK

Form Data
username: abcd
password: 123456

Figure 4: Successful Login

XSS and SQL Injection Attacks

During the SQL Injection Attack phase, we initiate the test attack by injecting basic malicious SQL code. This attack involves entering harmful SQL code into the username field of a login form, aiming to surpass authentication. By manipulating the query, the injected SQL code can lead to unexpected outcomes, such as gaining unauthorized access without a valid login and password. The login form showcases the username field containing the SQL injection payload, demonstrating its successful bypass of authentication and unauthorized access to

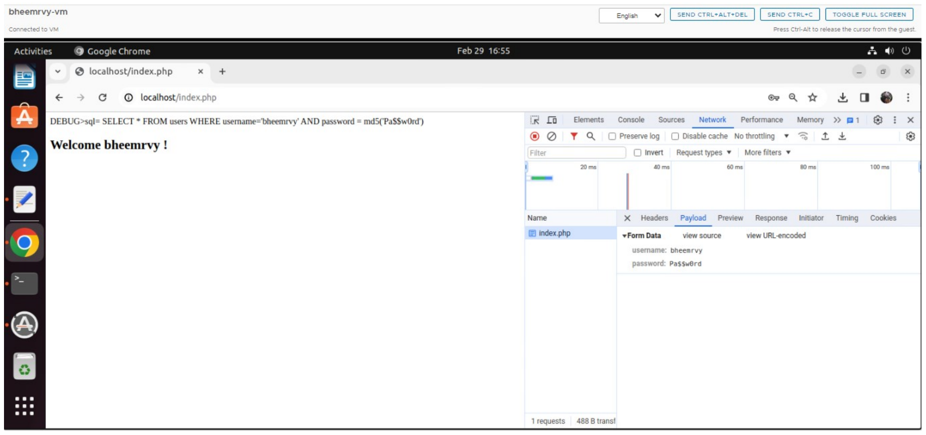


```
Welcome admin !
```

```
Form Data
username: admin
password: P4$$w0rd
```

Figure 5: Unsuccessful Login

the network. The vulnerability to SQL injection attacks arises from inadequate input validation and the absence of valid queries within the application code. When user input isn't thoroughly checked, input fields become susceptible to malicious SQL code insertion by attackers.



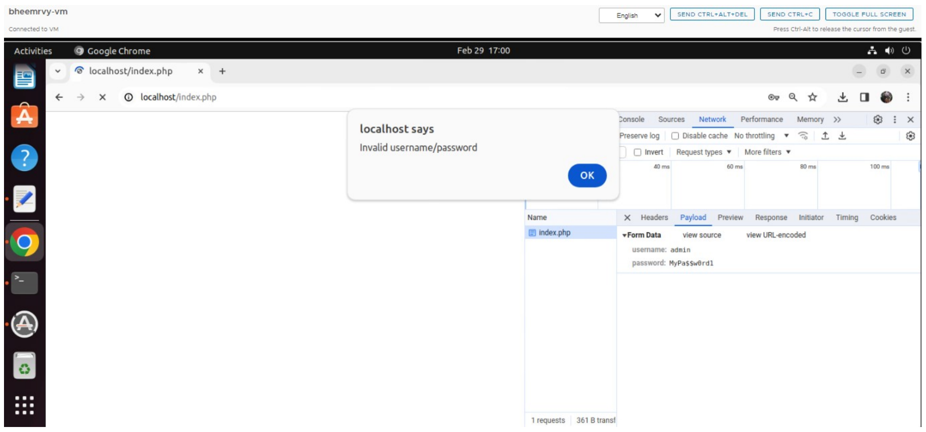
```
DEBUG>sql= SELECT * FROM users WHERE username='bheemrvy' AND password = md5('Pa$$w0rd')

Welcome bheemrvy !
```

```
Form Data
username: bheemrvy
password: Pa$$w0rd
```

Figure 6: Test injection by echo

Prepared Statement Implementation

The PHP program undergoes several security enhancements, such as the implementation of prepared statements and password hashing to prevent SQL injection attacks. Prepared statements isolate SQL code from data, making it difficult for attackers to insert malicious code. Parameterized queries further secure the system by treating input data as information rather than executable SQL code,



localhost says

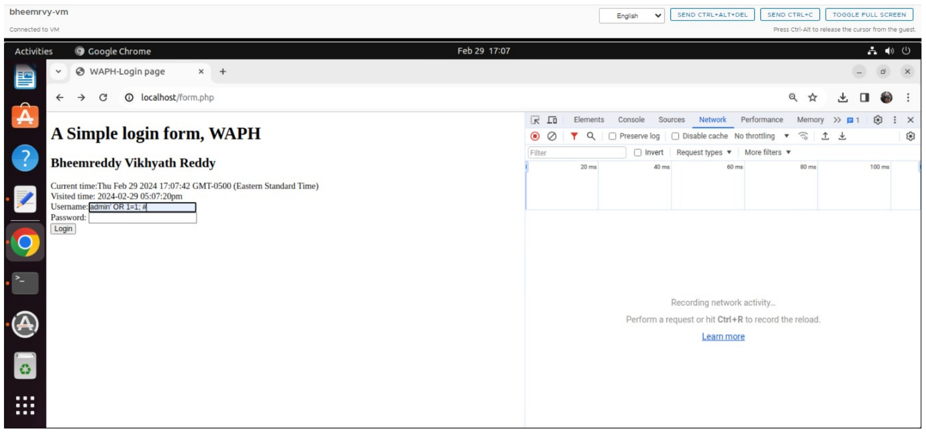
Invalid username/password

Form Data

username: admin

password: MyPa$$w0rd1

Figure 7: Unsuccessful Login

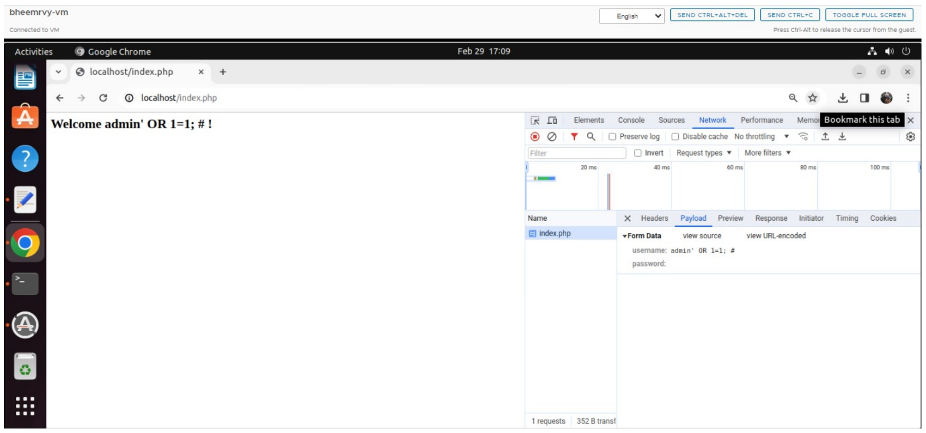


A Simple login form, WAPH

Bheemreddy Vikhyath Reddy

Current time:Thu Feb 29 2024 17:07:42 GMT-0500 (Eastern Standard Time)
Visited time: 2024-02-29 05:07:20pm
Username: admin' OR 1=1; #
Password:
Login

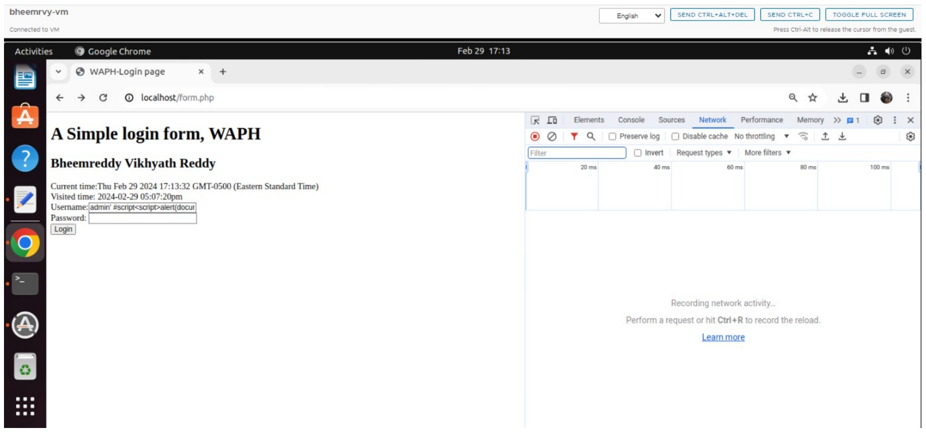
Figure 8: Injection with 1=1



Welcome admin' OR 1=1; # !

```
Form Data    view source    view URL-encoded
  username: admin' OR 1=1; #
  password:
```

Figure 9: Input for test Injection

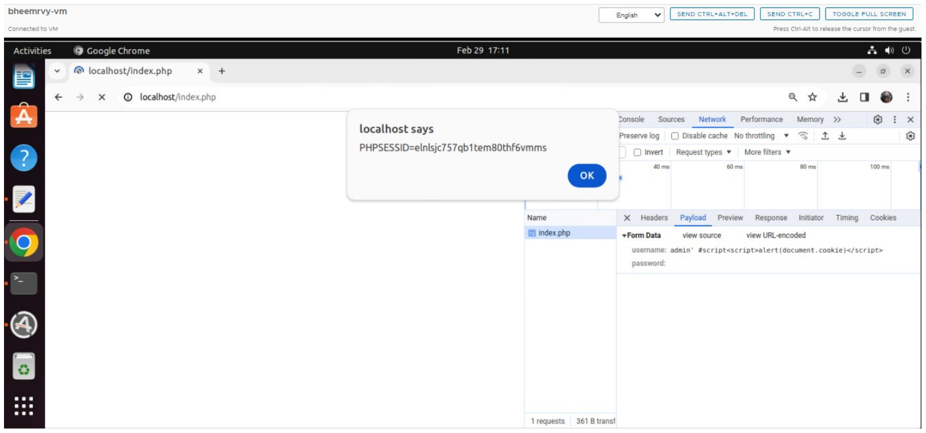


A Simple login form, WAPH

Bheemreddy Vikhyath Reddy

Current time:Thu Feb 29 2024 17:13:32 GMT-0500 (Eastern Standard Time)
Visited time: 2024-02-29 05:07:20pm
Username: admin' #script<script>alert(docur
Password:
Login

Figure 10: Input for test

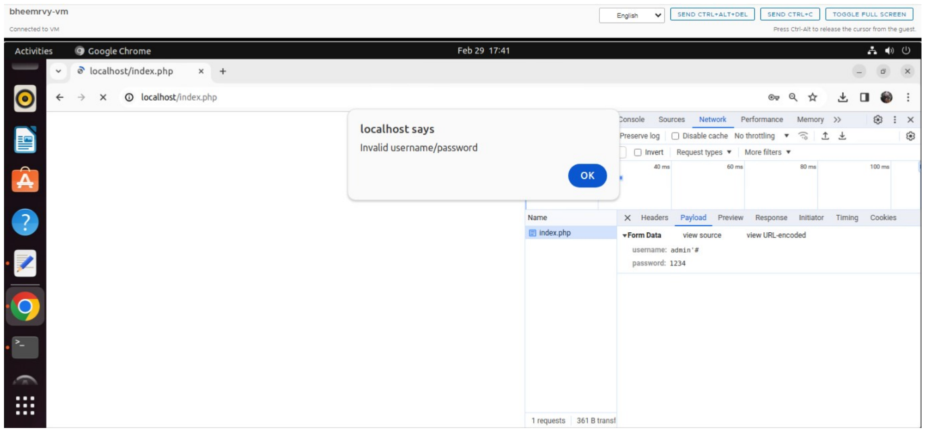


```
localhost says
PHPSESSID=elnlsjc757qb1tem80thf6vmms
```

```
Form Data
username: admin' #script<script>alert(document.cookie)</script>
password:
```

Figure 11: Cookies

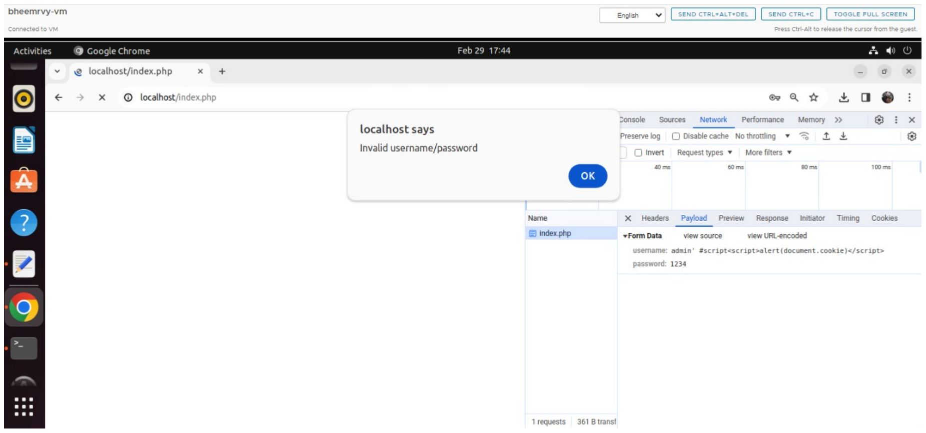
thus preventing SQL injection. Additionally, password hashing with the md5() function adds an extra layer of security by encrypting passwords stored in the database. The program also utilizes the mysqli extension for secure database connections and incorporates error handling to ensure continuous operation and minimize vulnerabilities. However, there are still some areas of concern, such as concatenating input values directly into the query string, which could lead to confusion and potential vulnerabilities. The program also lacks proper handling of empty username/password fields, potentially allowing unauthorized access. Continuous updates and vigilance regarding database errors are crucial for maintaining system security. Moreover, input validation and normalization techniques are essential for preventing flaws and ensuring data accuracy and integrity.



```
localhost says
Invalid username/password
```

```
Form Data
username: admin'#
password: 1234
```

Figure 12: Login after Code implementation



localhost says

Invalid username/password

Form Data

```
username: admin' #script<script>alert(document.cookie)</script>
password: 1234
```

Figure 13: Login after Code implementation