Installation and Usage of DVWA for SQL Injection Testing

ADITHYAN SHABU

CONTENTS

1. Installation of DVWA in Docker

I employed Docker to make the installation of Damn Vulnerable Web Application (DVWA) less tedious. Here is the summery of the steps involved in completing the installation:

a. Clone the Repository

First step was to clone the DVWA repository from pentestlab.github.io using the following command:

git clone https://github.com/eystsen/pentestlab.git

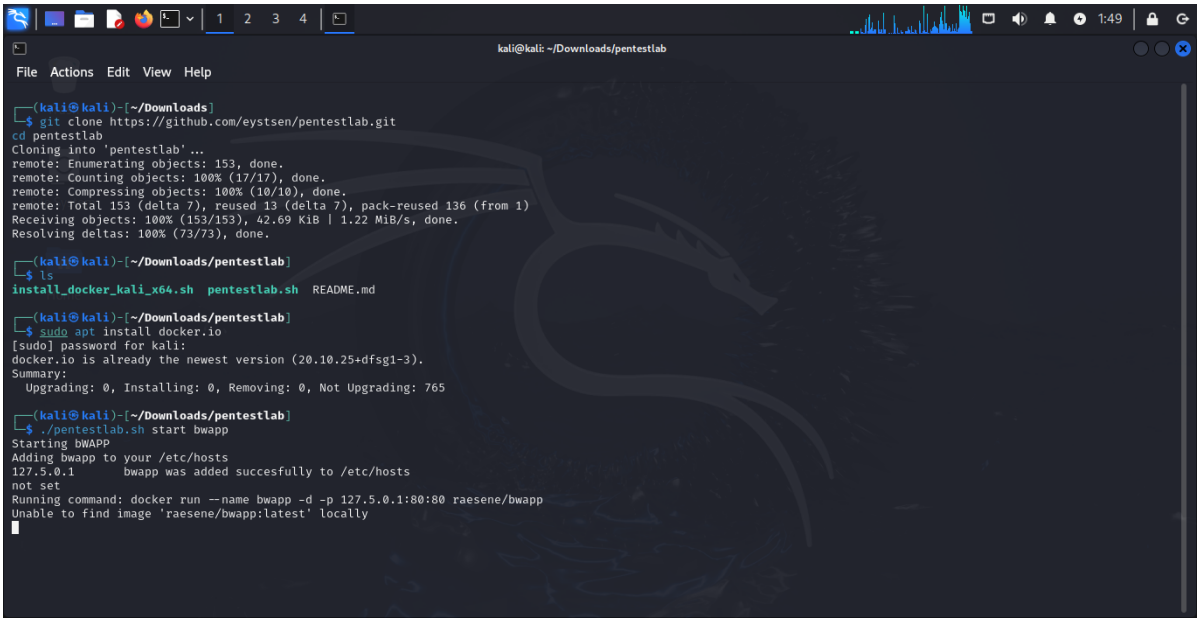
b. Start the Docker Container

I went into the DVWA folder after cloning the repository, and with a series of Docker commands, I started and fired the engines.

c. Opened the terminal and went into the cloned pentestlab folder

d. Ran the following command to install Docker container

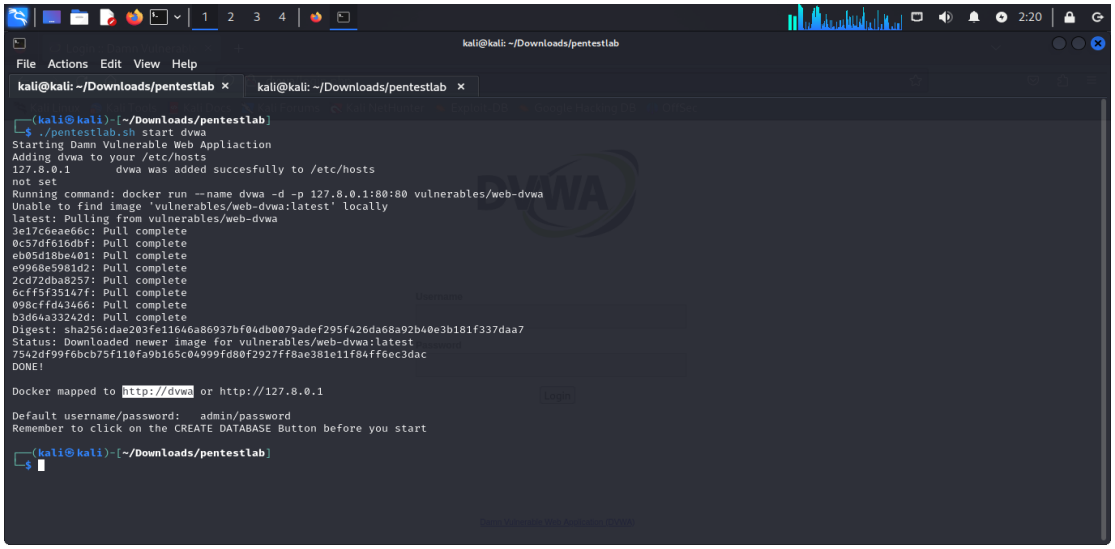
sudo apt install docker.io



e. Accessing to the DVWA Web Page

After starting the Docker container,I executes the following command to access the DVWA web page

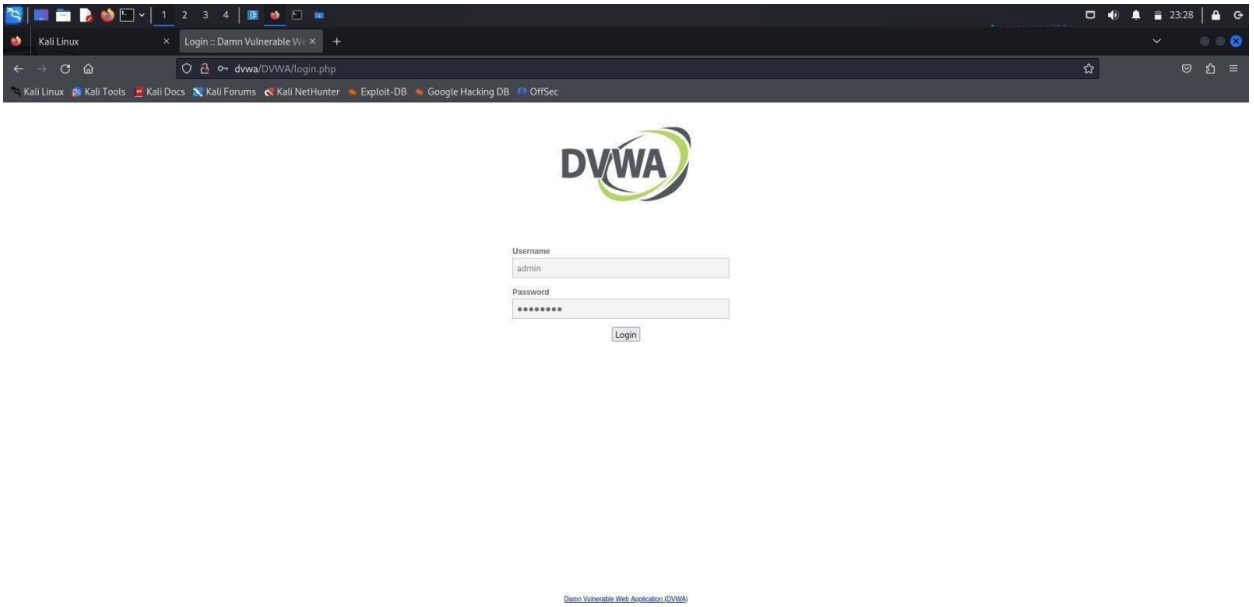
./pentestlab.sh start dvwa



f. Logging In

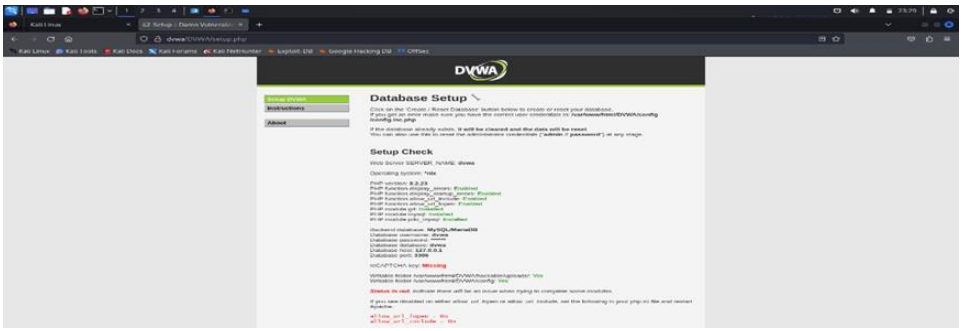
At the login page, I used the default credentials:

- Username: admin
- Password: password



g. Resetting the Database

Reset the database, after logging in for the first time. Then click the "Reset Database" button. The system redirected me back to the login page, Once the reset was completed.



h. Logging In Again

After resetting the database, I once again used the default credentials to gain access to the DVWA dashboard

i. Completion

At this point, the DVWA setup was complete, and ready for vulnerability testing.

2. Performing SQL Injection on DVWA

a. SQL Injection (Low Security Level)

I began by testing SQL injection on the Low security level.

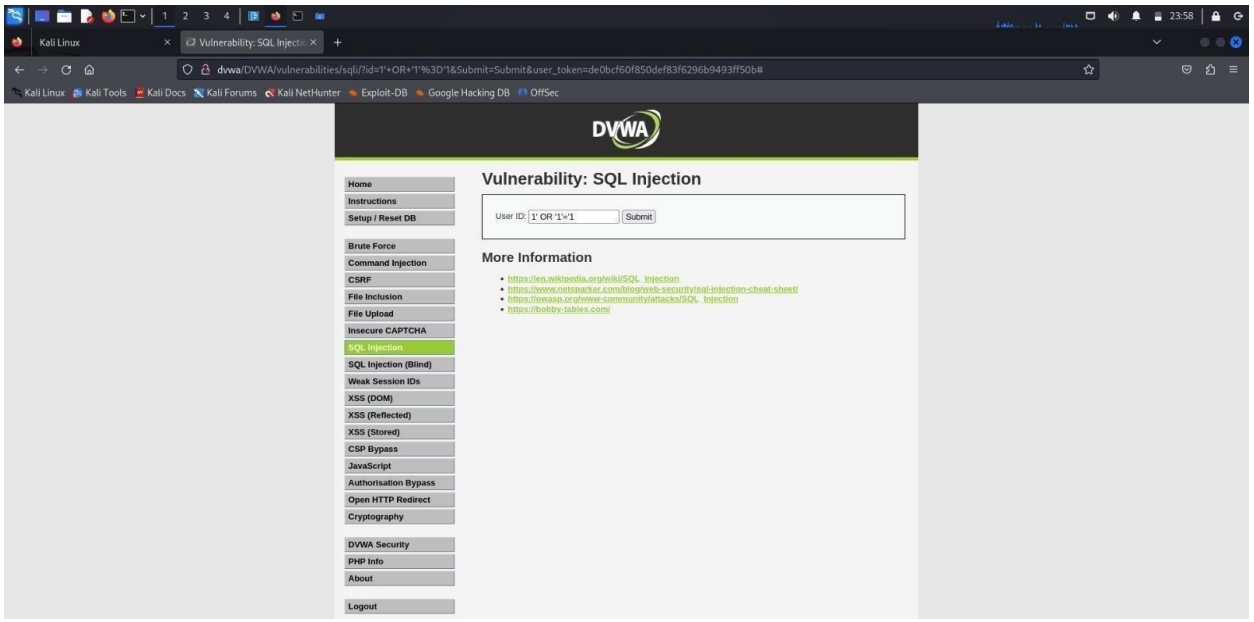
i. Initial Injection

I quickly identified the input field for injecting SQL code, after accessing the SQL injection page.

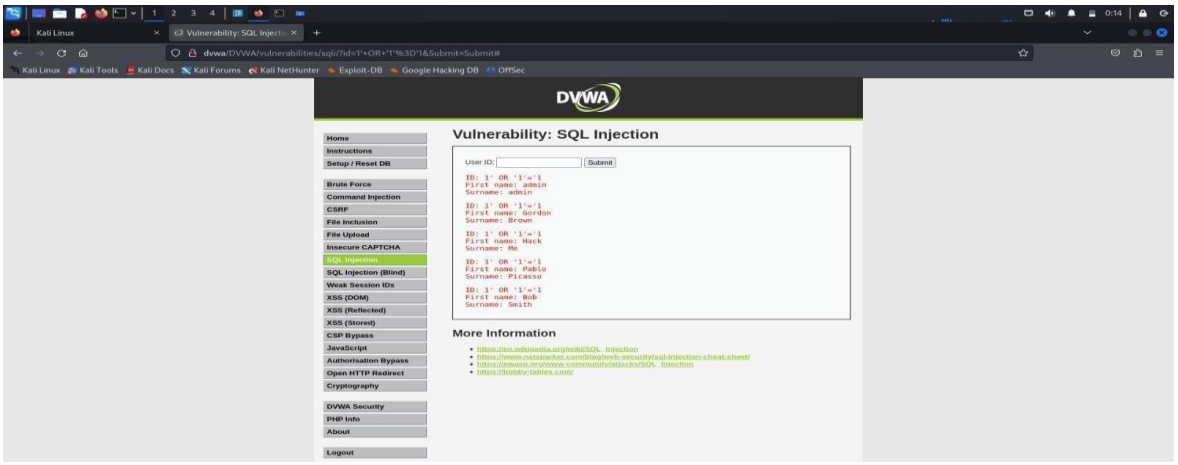
ii. SQL Payload

The following basic SQL injection string is used:

1' OR '1'='1



This clever payload circumvented the requirement for valid input and proceeded to reveal the first names and surnames of all users

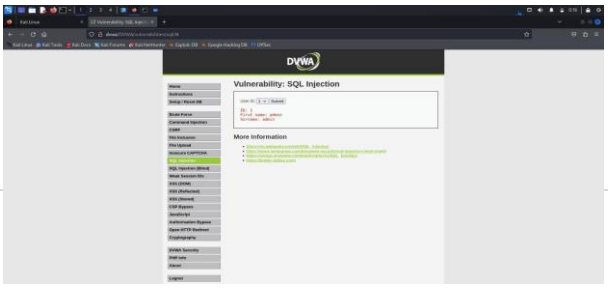


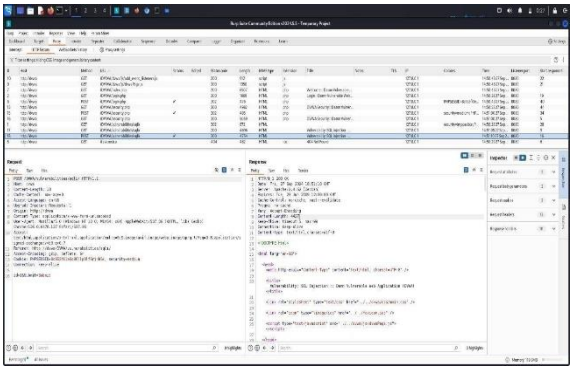
b. SQL Injection (Medium Security Level)

I changed the DVWA security setting to Medium and conducted the test with an enhanced payload.

i. Using Burp Suite

I employed Burp Suite to intercept the HTTP request and then tweaked the ID parameter in the request, injecting a a more sophisticated SQL string

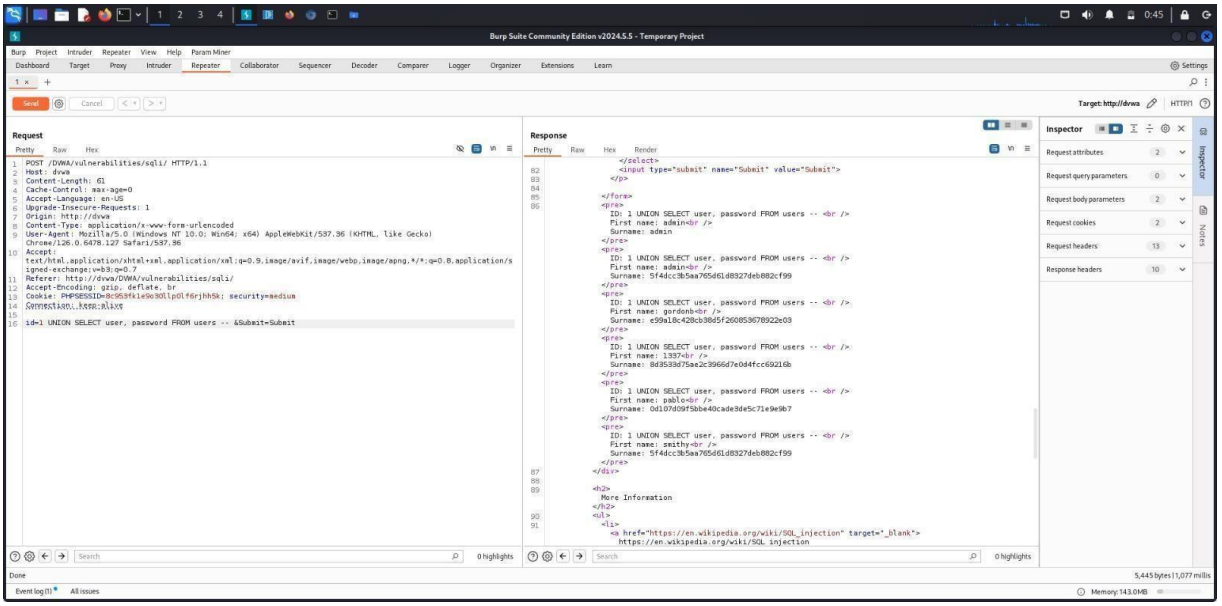




ii. SQL Injection String

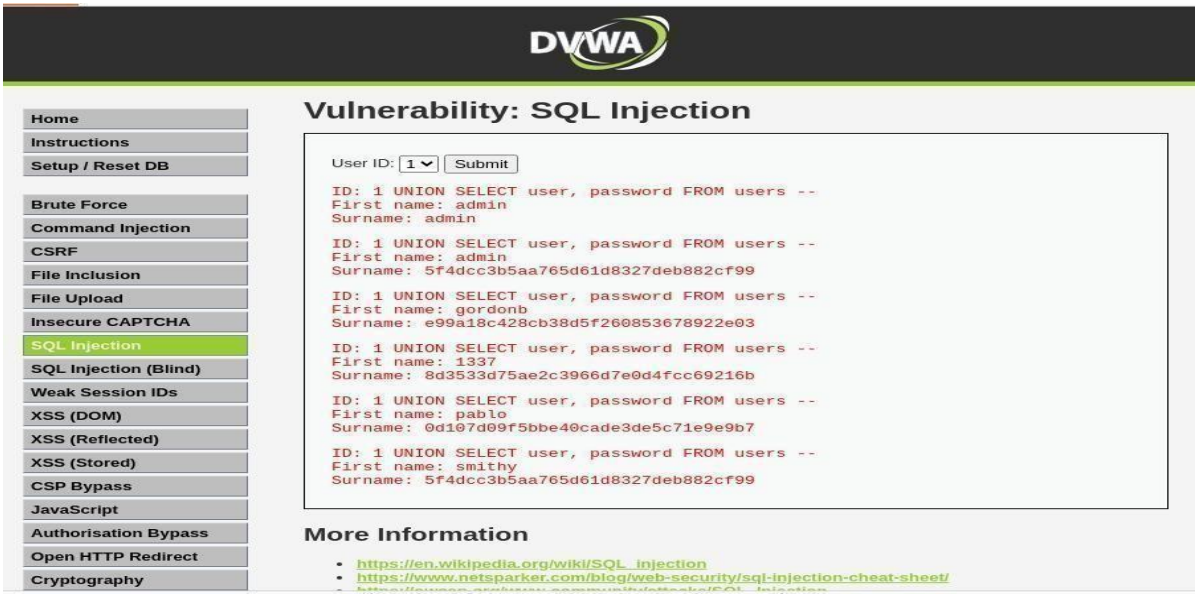
I inserted the following payload into the `id` field:

1 UNION SELECT user, password FROM users--



iii. Execution

After modifying the request in Burp Suite.I sent it to the server.As a result,I was able to extract usernames and passwords from the system's response.

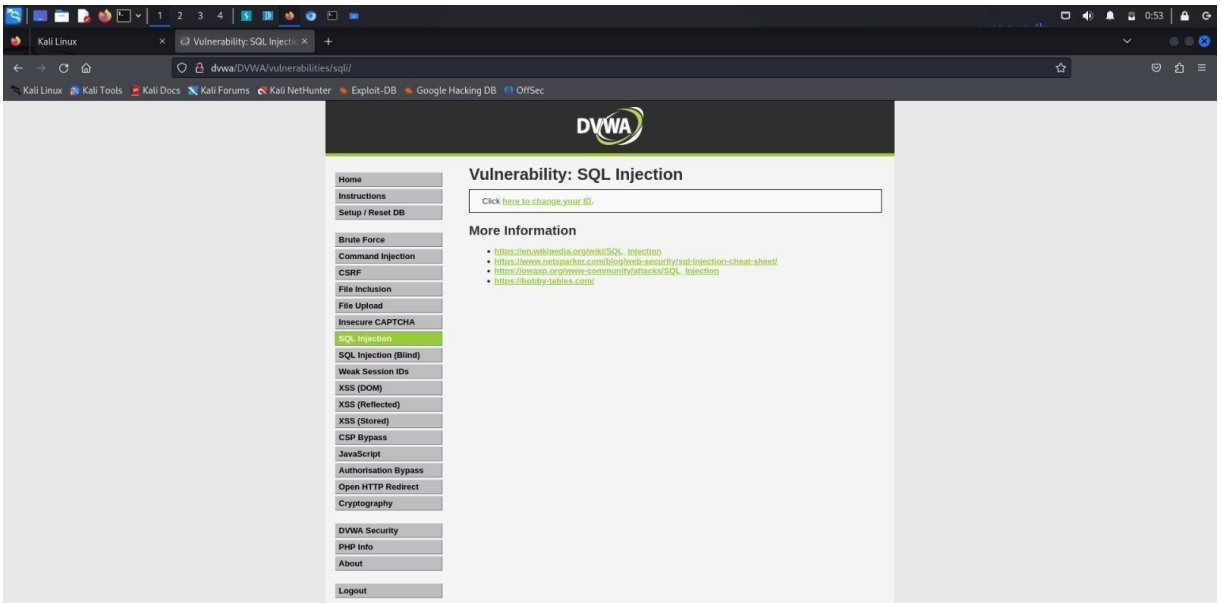


c. SQL Injection (High Security Level)

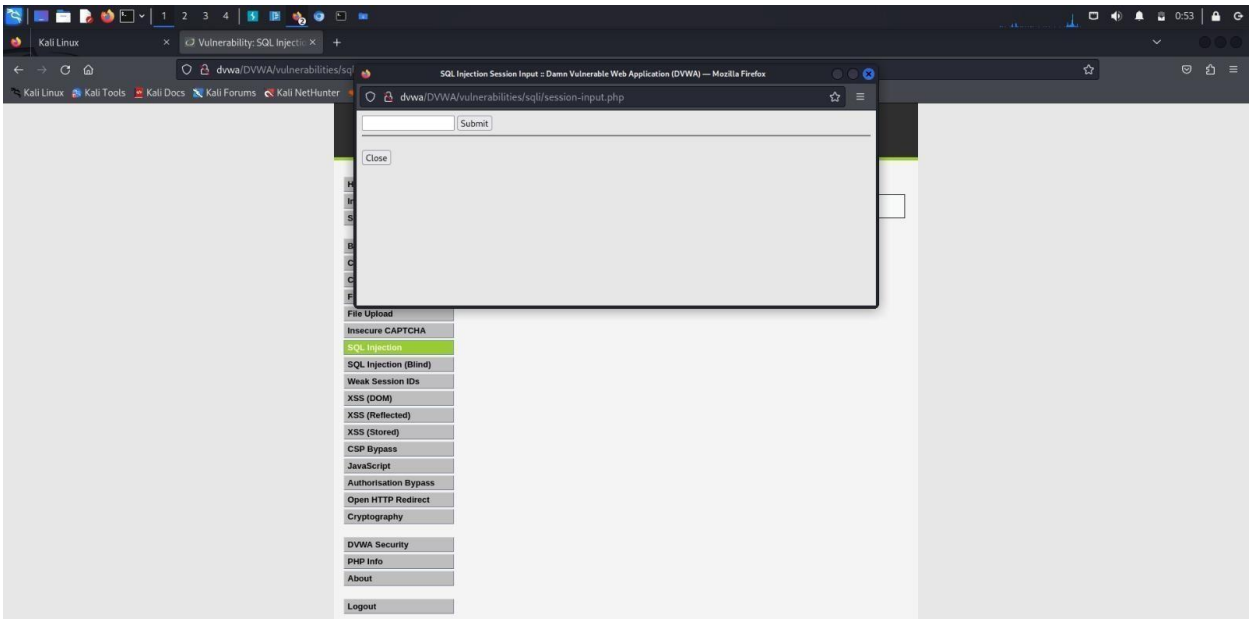
Finally, I tested SQL injection on the High security level.

i. Identifying the Injection Point

After selecting the ‘Here to Change your ID’ button,you’ll notice a slightly difference in the interface at the high security level



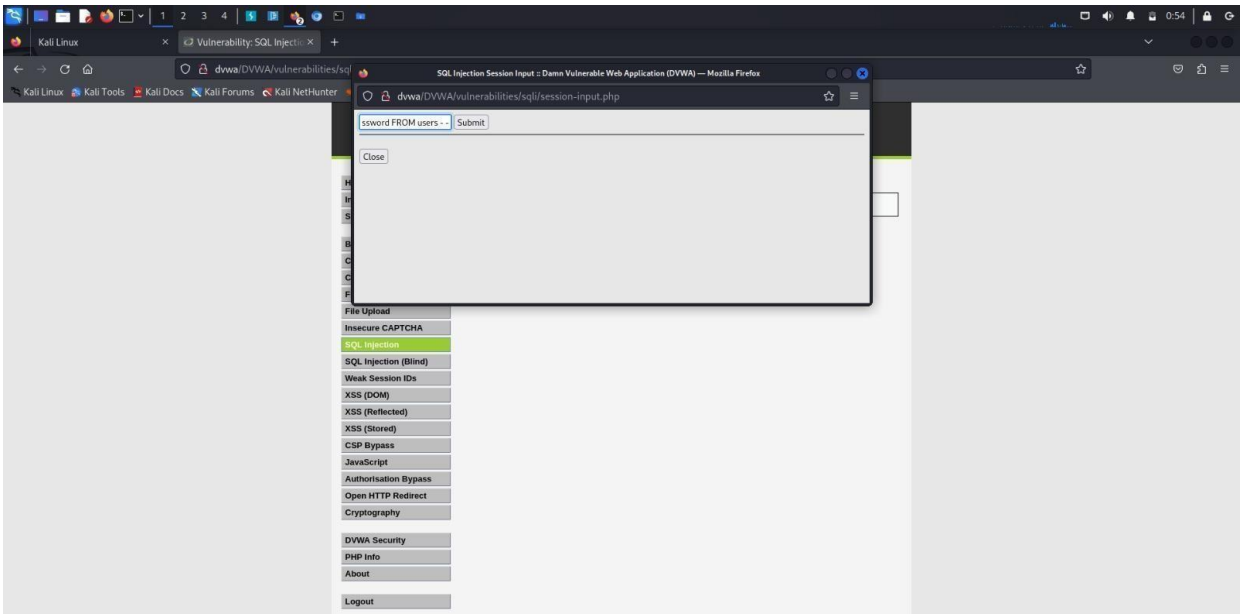
and there is a new window appeared where I could input SQL command.



i. Injection Payload

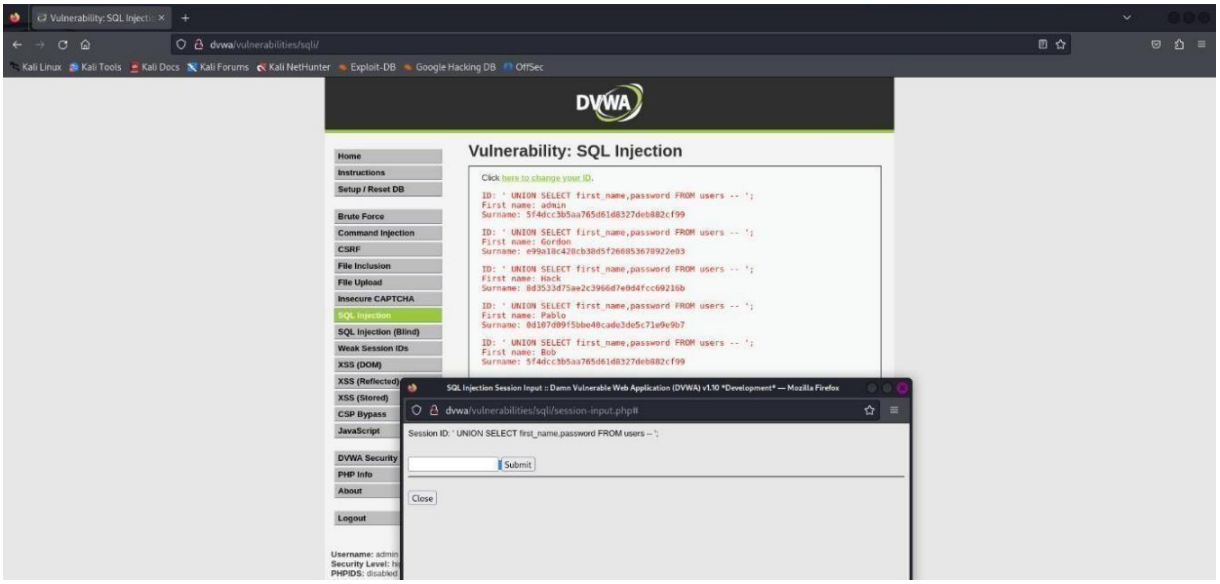
I inserted the following SQL injection string: '

UNION SELECT user, password FROM users - -



i. Results

After executing the provided code, I was able to confirm the vulnerability even under the most stringent security settings



Conclusion

After setting up DVWA via docker,I conducted SQL injection tests at varying security levels Employing both basic and advanced SQL injection payloads along with Burp Suite for request interception,I successfully retrieved sensitive data from the database across all security configurations ,effectively showcasing the vulnerabilities.