Report: Installation and Usage of DVWA for SQL Injection.

Hazeena khalid

9/27/2024

CONTENTS

1. Installation of DVWA using Docker

To install Damn Vulnerable Web Application (DVWA), I used Docker for a streamlined setup. Below are the steps I followed to complete the installation:

1.1 Cloning the Repository

I started by cloning the DVWA repository from pentestlab.github.io using the following command:
git clone https://github.com/eystsen/pentestlab.git

1.2 Starting the Docker Container

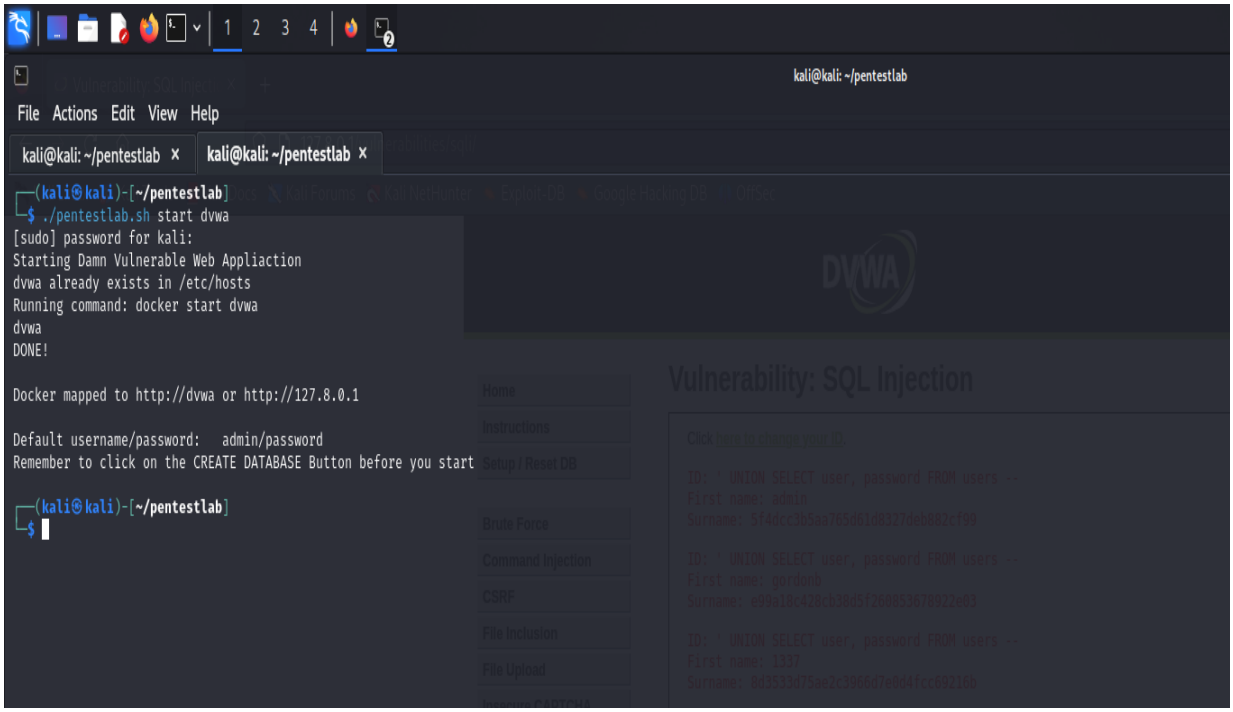
After cloning the repository, I navigated to the DVWA folder and ran Docker commands to initiate the web application. The specific steps I followed were:
1. Opened the terminal and navigated to the cloned pentestlab folder.
2. Ran the following command to install Docker container:

sudo apt install docker.io

1.3 Accessing to the DVWA Web Page

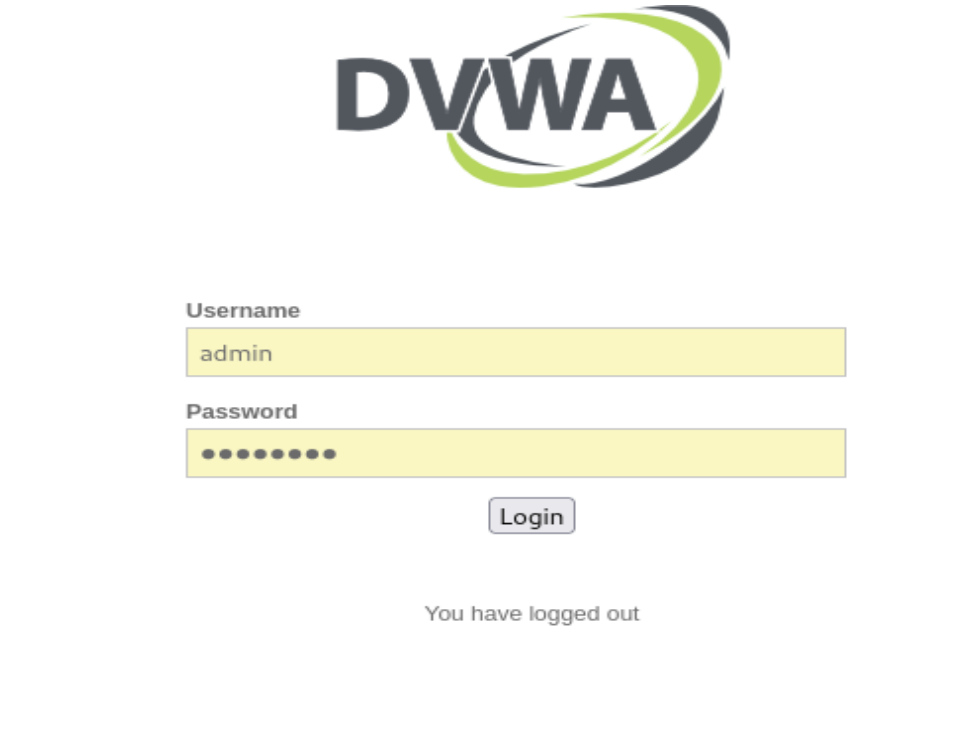
Once the Docker container was running, I run this command for accessing the dvwa web page.
Command:

./pentestlab.sh start dvwa



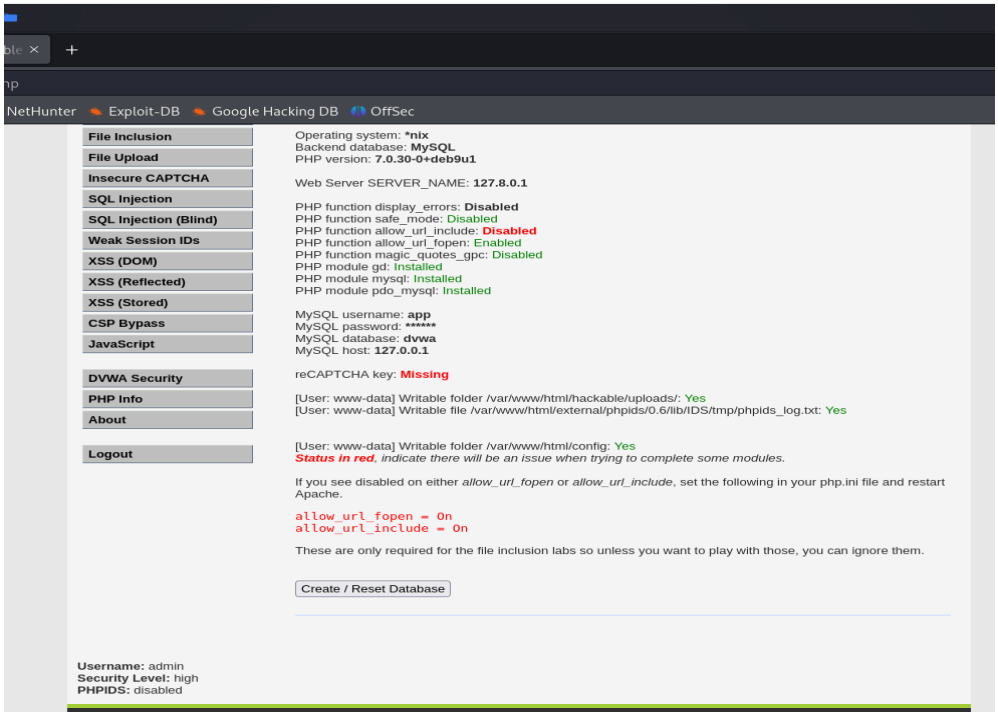
1.4 Logging In

At the login page, I used the default credentials:
- Username: admin
- Password: password



1.5 Resetting the Database

After logging in for the first time, I was prompted to reset the database. I clicked the "Reset Database" button. Once the reset was completed, the system redirected me back to the login page.



1.6 Logging In Again

After resetting the database, I logged in again with the default credentials to access the DVWA dashboard.

1.7 Completion

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

2. Performing SQL Injection on DVWA

2.1 SQL Injection (Low Security Level)

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

2.1.1 Initial Injection

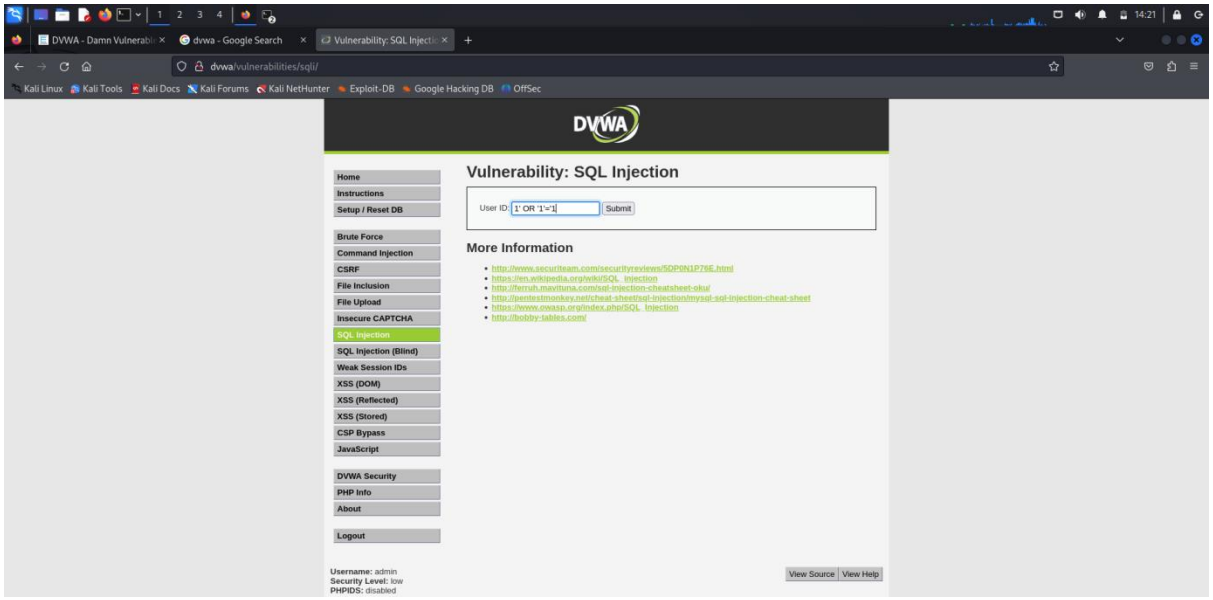
After accessing the SQL injection page, I quickly identified the input field for injecting SQL code.

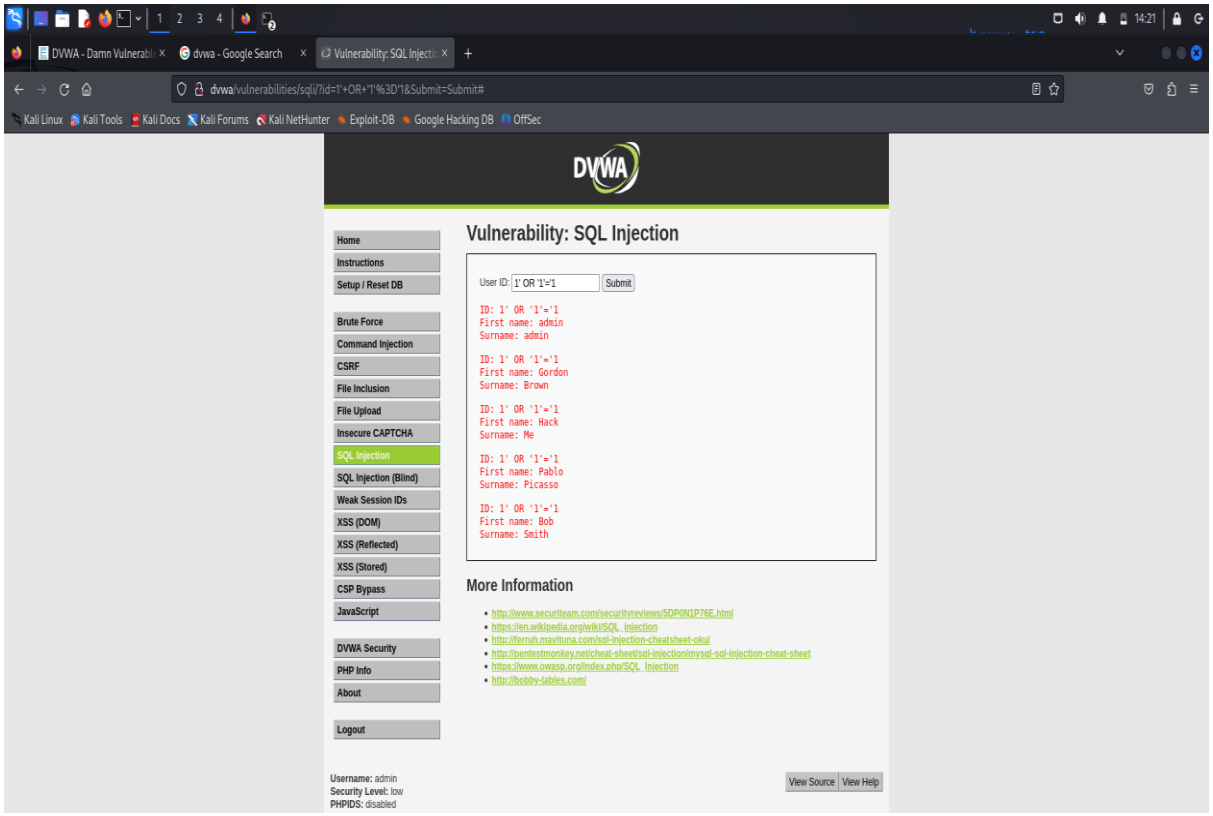
2.1.2 SQL Payload

I used the following basic SQL injection string:

```
1' OR '1'='1
```

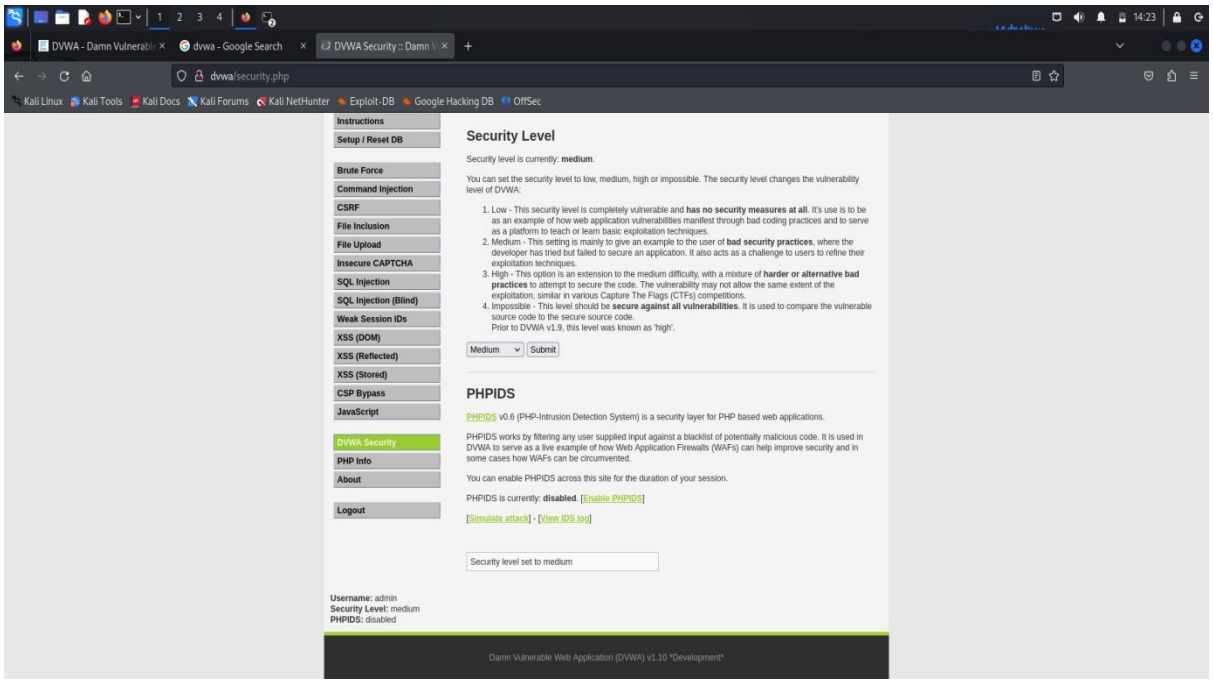
This payload bypassed the need for valid input and displayed the first name and surname of all users.





2.2 SQL Injection (Medium Security Level)

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



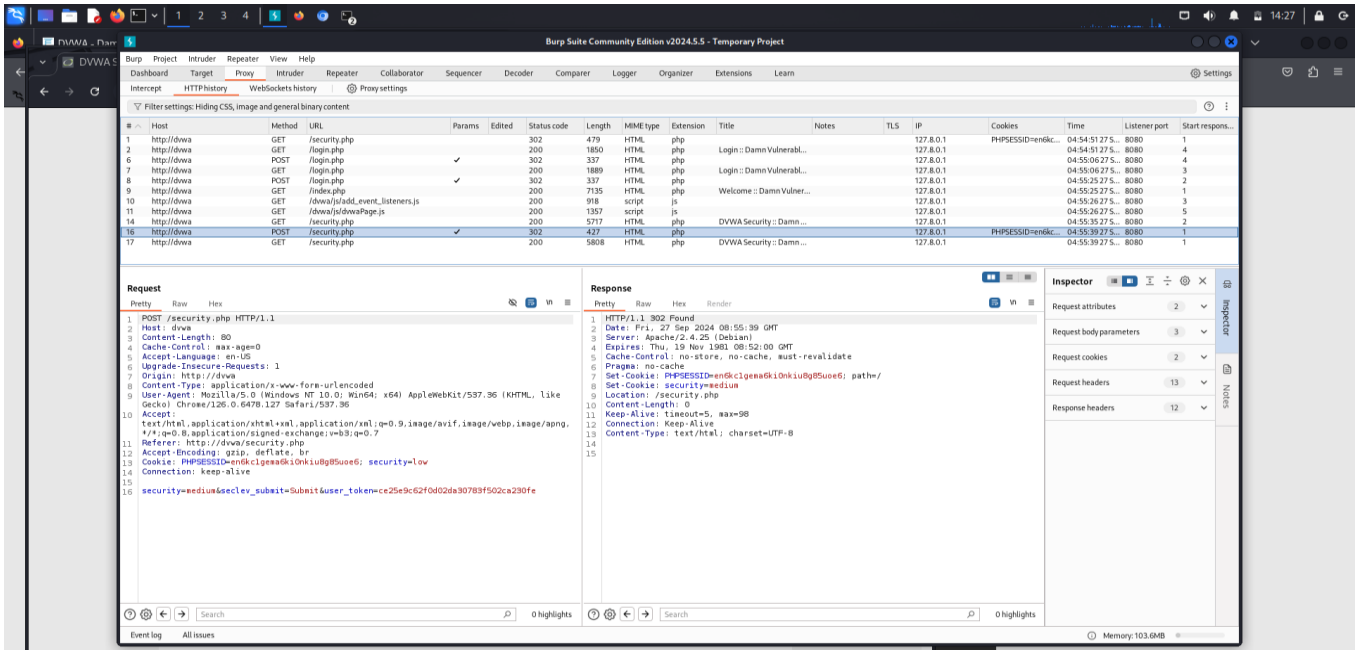
2.2.1 Using Burp Suite

I used Burp Suite to intercept the HTTP request. I modified the `id` parameter in the request to insert a more advanced SQL injection string.

2.2.2 SQL Injection String

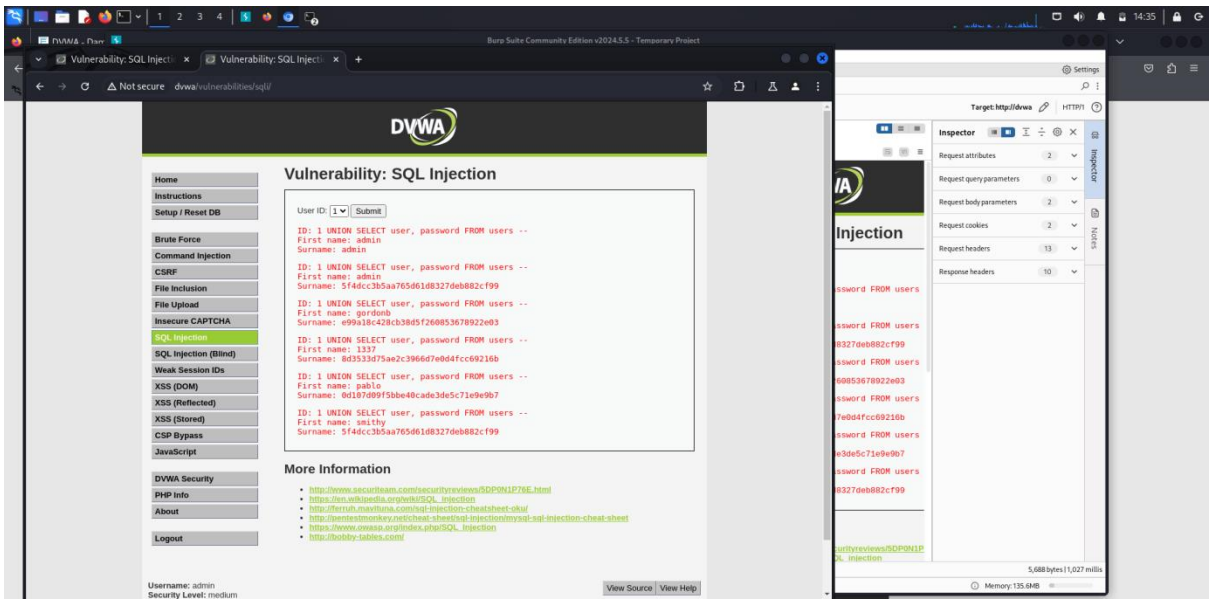
I inserted the following payload into the `id` field:

```
1 UNION SELECT user, password FROM users - -
```



2.2.3 Execution

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



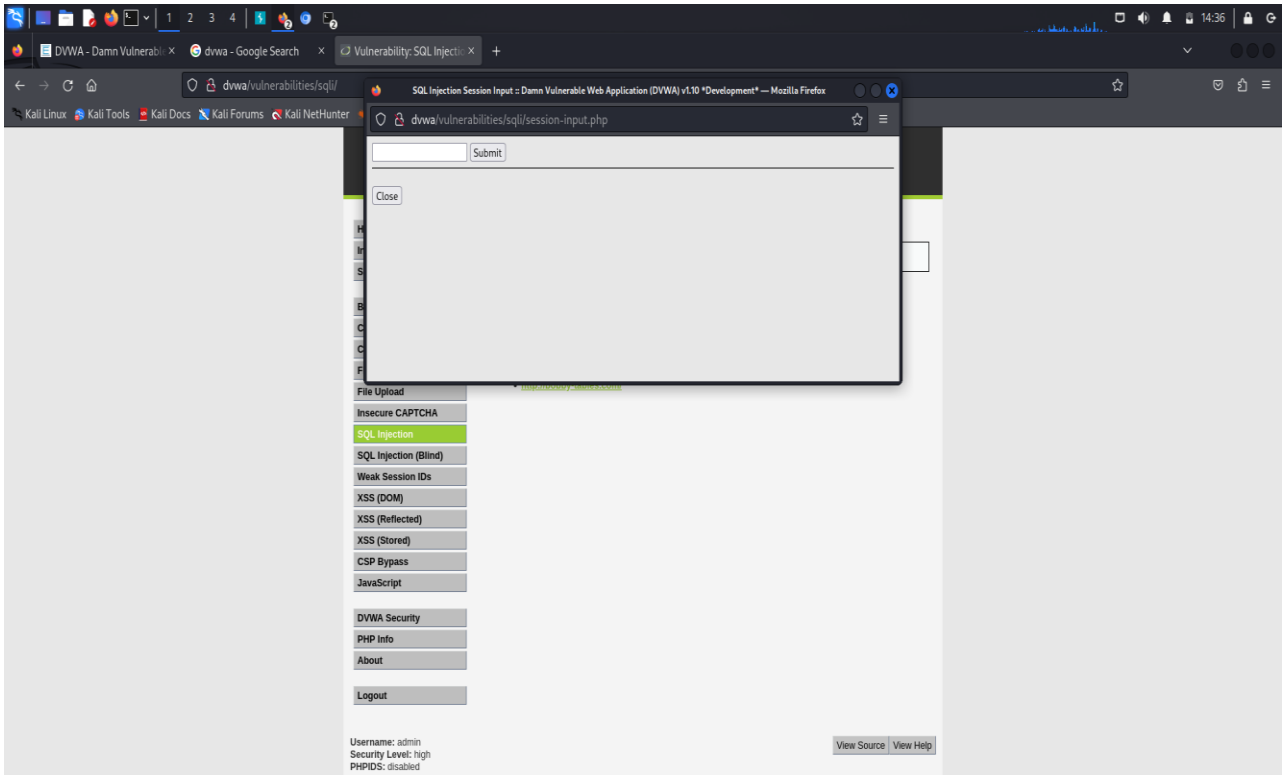
2.3 SQL Injection (High Security Level)

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

2.3.1 Identifying the Injection Point

At the High security level, the interface is slightly different. After clicking the “Here to change your ID” button,

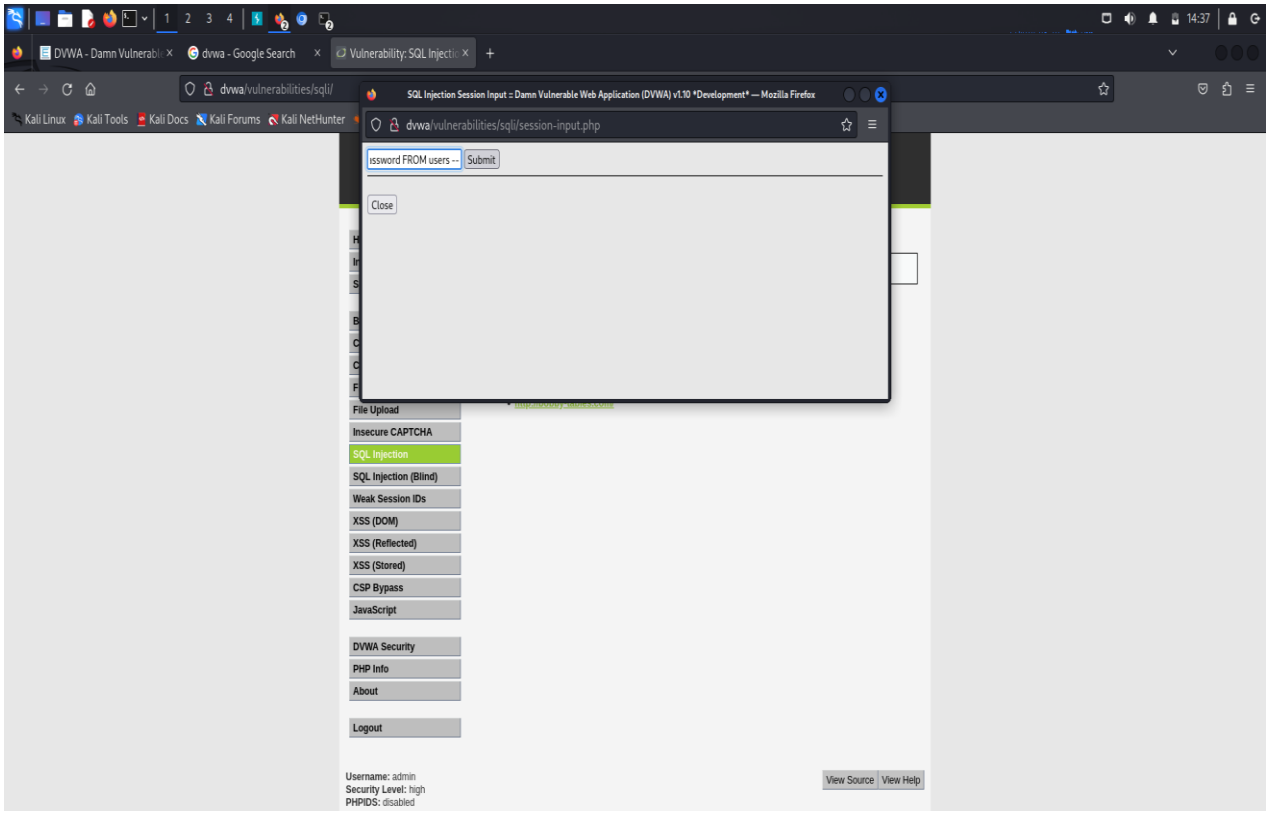
a new window appeared where I could input SQL command.



2.3.2 Injection Payload

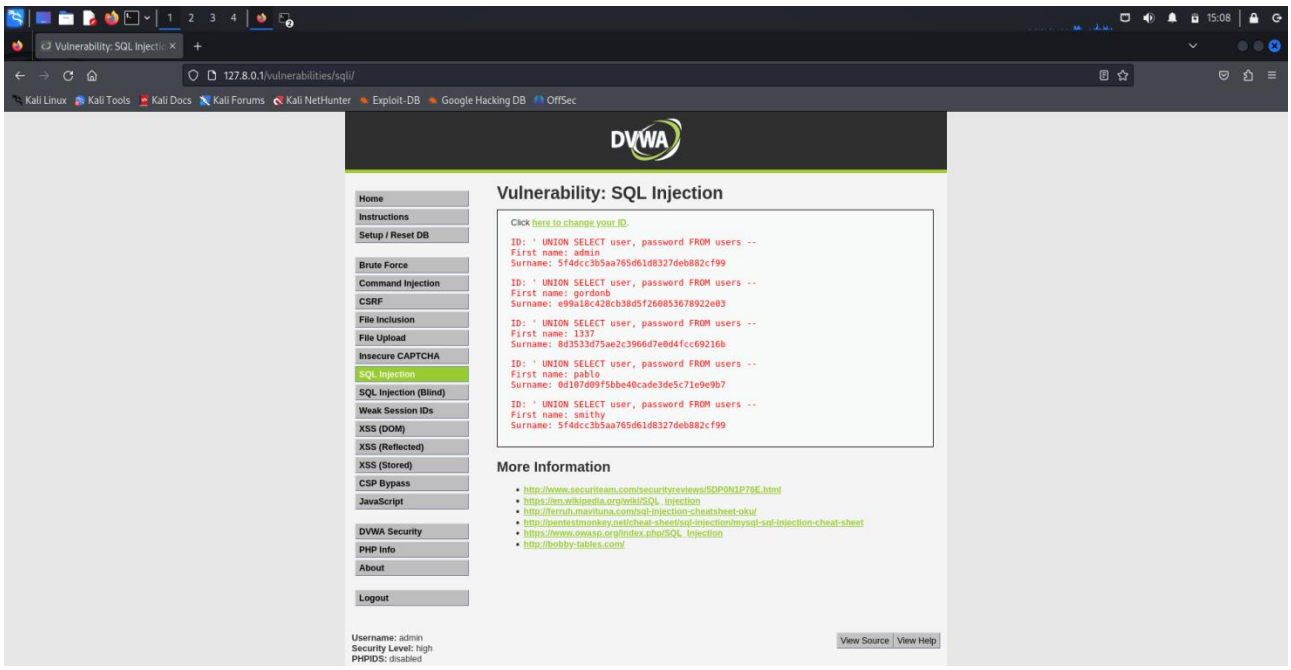
I inserted the following SQL injection string:

' UNION SELECT user, password FROM users - -



2.3.3 Results

After submitting the malicious code, the system returned a list of usernames and passwords, successfully confirming the vulnerability even at the highest security setting.



Conclusion

I successfully installed DVWA using Docker and tested SQL injection vulnerabilities at different security levels. Using simple and advanced SQL injection payloads, along with Burp Suite for request interception, I was able to extract sensitive information from the database across all security settings, demonstrating the effectiveness of these attacks.