REPORT: DVWA for SQL InjectionTesting

AISWARYA.D

Contents

Installation of DVWA using Docker 3
Performing SQL Injection on DVWA 6
SQL Injection (Low Security Level) 6
SQL Injection (Medium Security Level) 7
SQL Injection (High Security Level) 10
Conclusion 12

1. Installing DVWA using Docker

In order to set up Damn Vulnerable Web Application (DVWA) installation, I used Docker in order to ensure it was well and neatly organized. To complete the installation, follow these steps:

1.1 Clone Repository

I cloned the DVWA repository from pentestlab.github.io, using the following command:

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

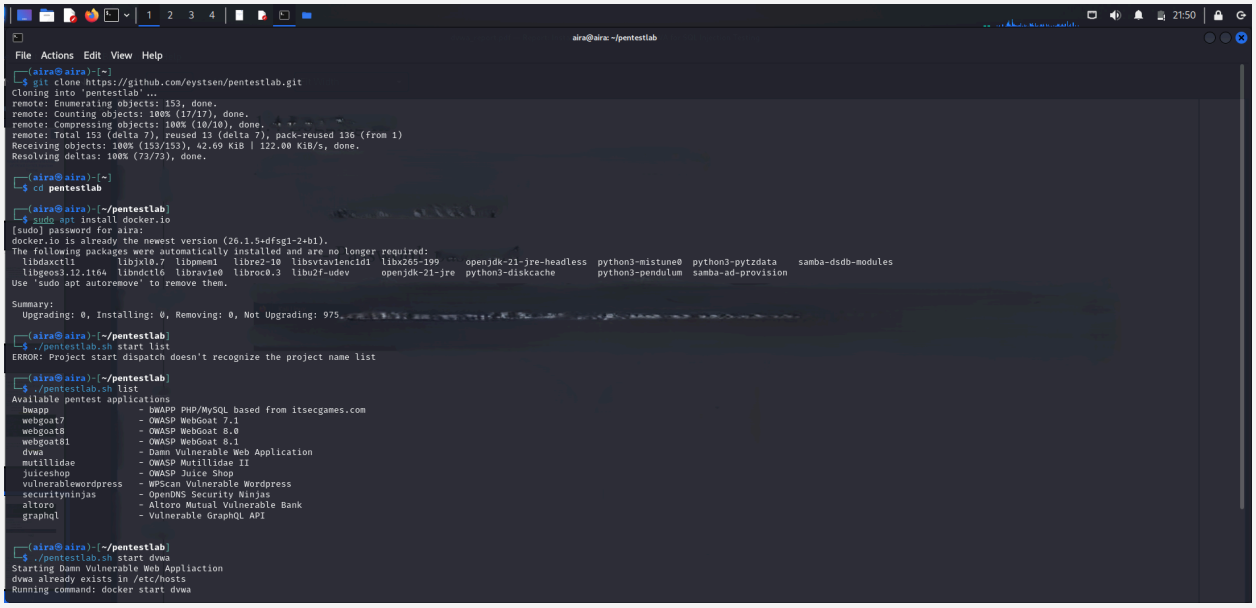
1.2 Start Docker Container

After cloning the repository, I entered the DVWA folder. I then proceeded to run the Docker commands in

Launch the web application. Activities involved:

1. Opened terminal and moved into pentestlab folder that had been cloned
2. executed the following command to install Docker container:

```
sudo apt install docker.io
```

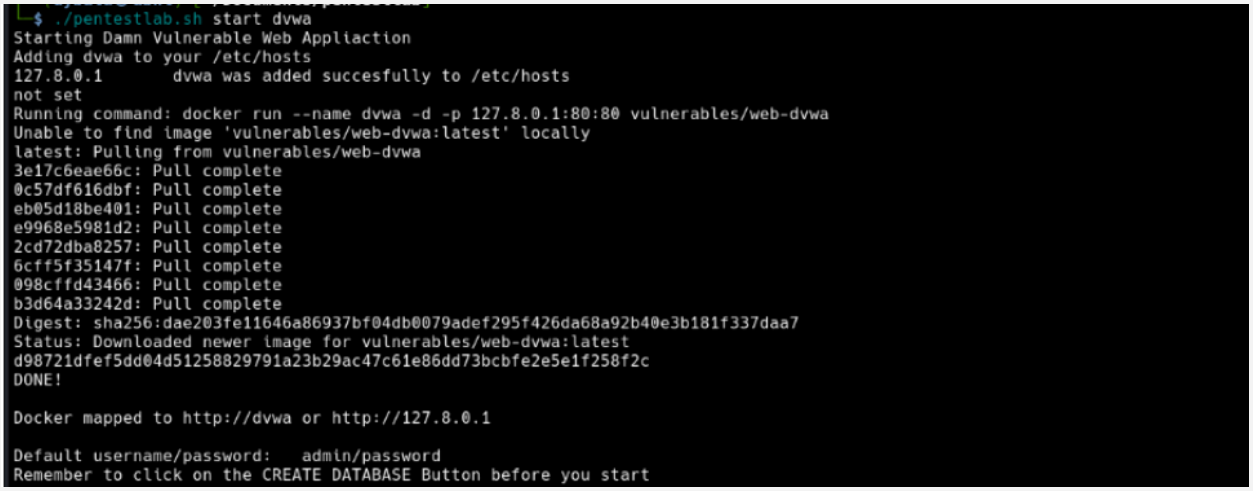


Screenshot 1

1.3 Access to the DVWA Web Page

Command used to access the dvwa webpage, when Docker was running.

./pentestlab.sh start dvwa

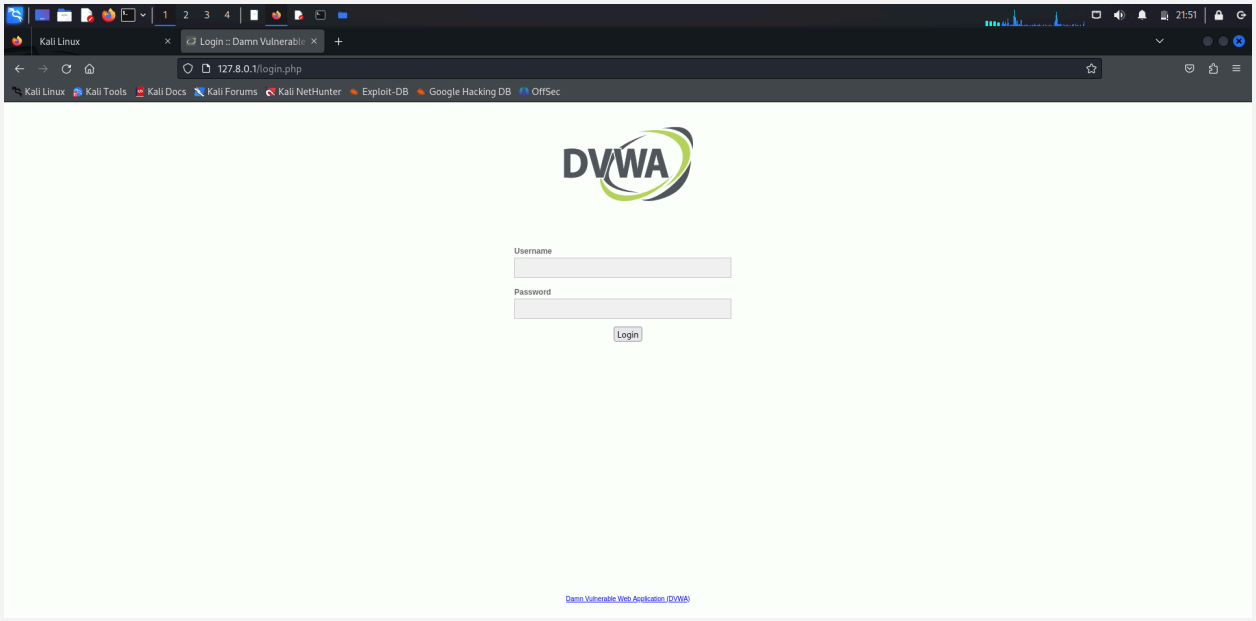


Screenshot 2

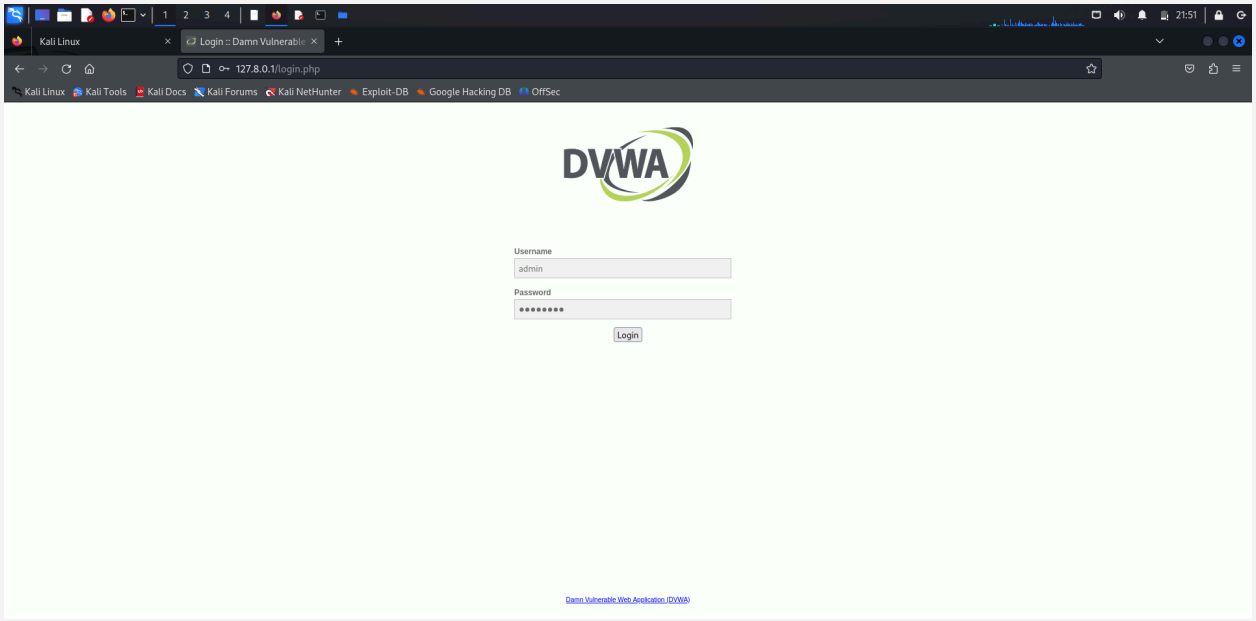
1.4 Logging In

Default credentials were applied at the login page:

Username: admin
Password: password



Screenshot 3



Screenshot 4

1.5 Resetting the Database

When I logged in for the very first time with the default admin login credentials, I got a prompt to reset the database. I clicked the "Reset

Database" button (I didn't take a screenshot of this step). When the reset was over, the system took me back to the login page.

1.6 Login Again

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

1.7 Completion

At this point, the DVWA was in a stage that would be ideal for configuration into something great for vulnerability testing.

2. SQL Injection on DVWA

2.1 SQL Injection (Low Security Level)

First, I began practicing SQL injection on the Low security level.

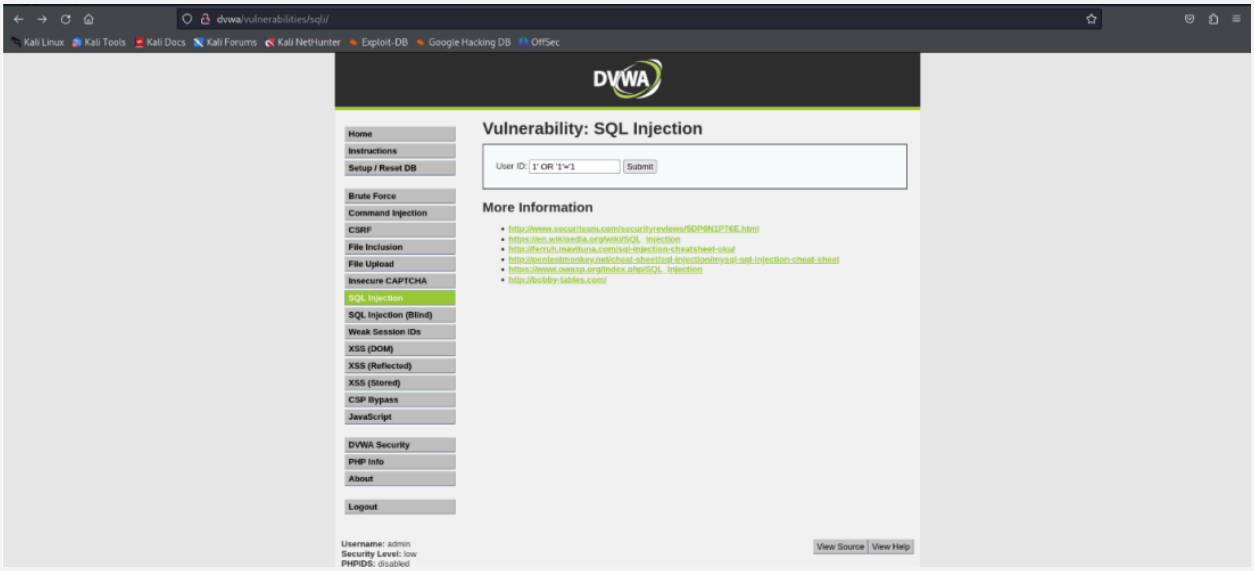
2.1.1 Injection

After accessing the page of SQL injection, it is easy to find where to inject the SQL code.

2.1.2 SQL Payload

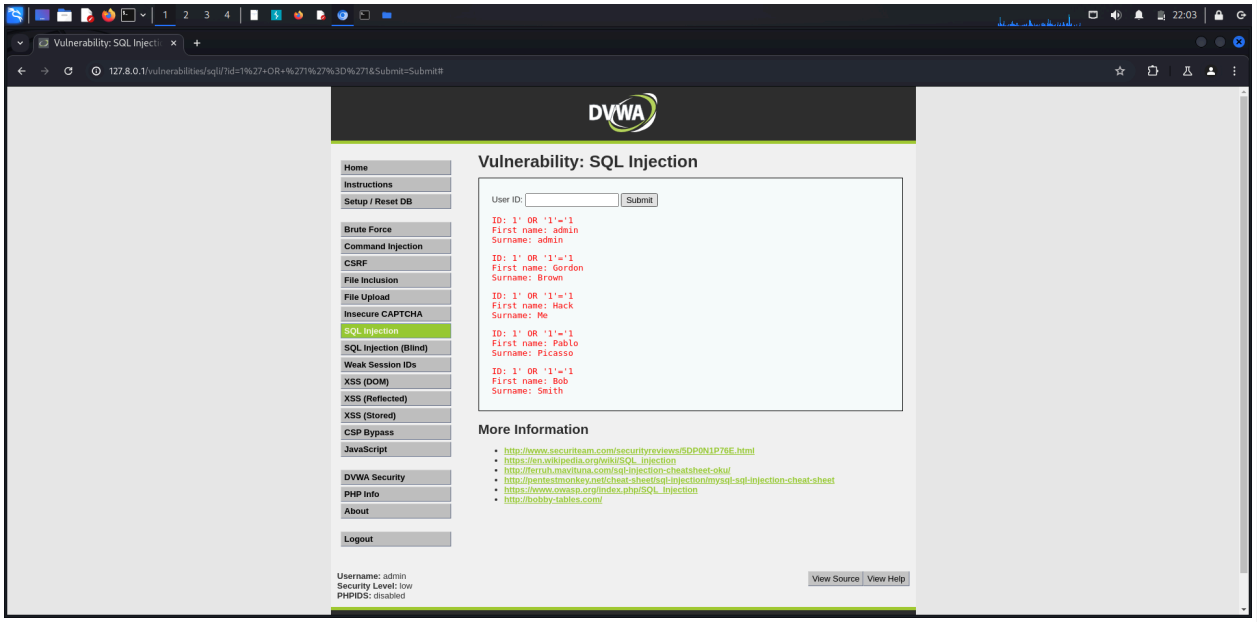
This is a primitive SQL string injection:

1' OR '1'='1



Screenshot 5

This payload eliminated the need for a valid input, and it printed out the first and last names of all users.



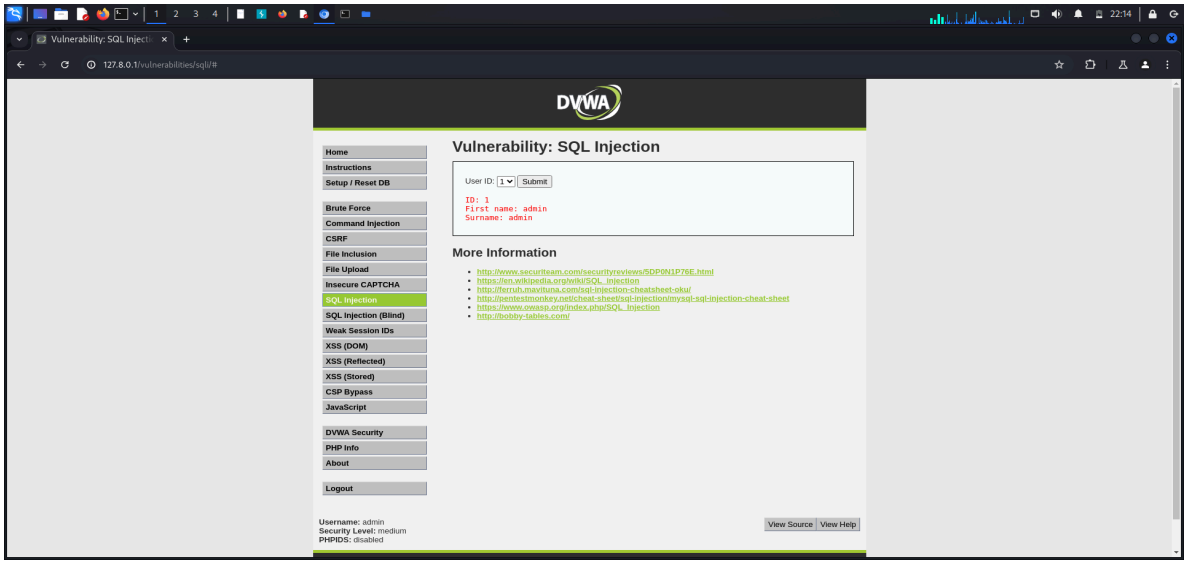
Screenshot 6

2.2 SQL Injection (Medium Security Level)

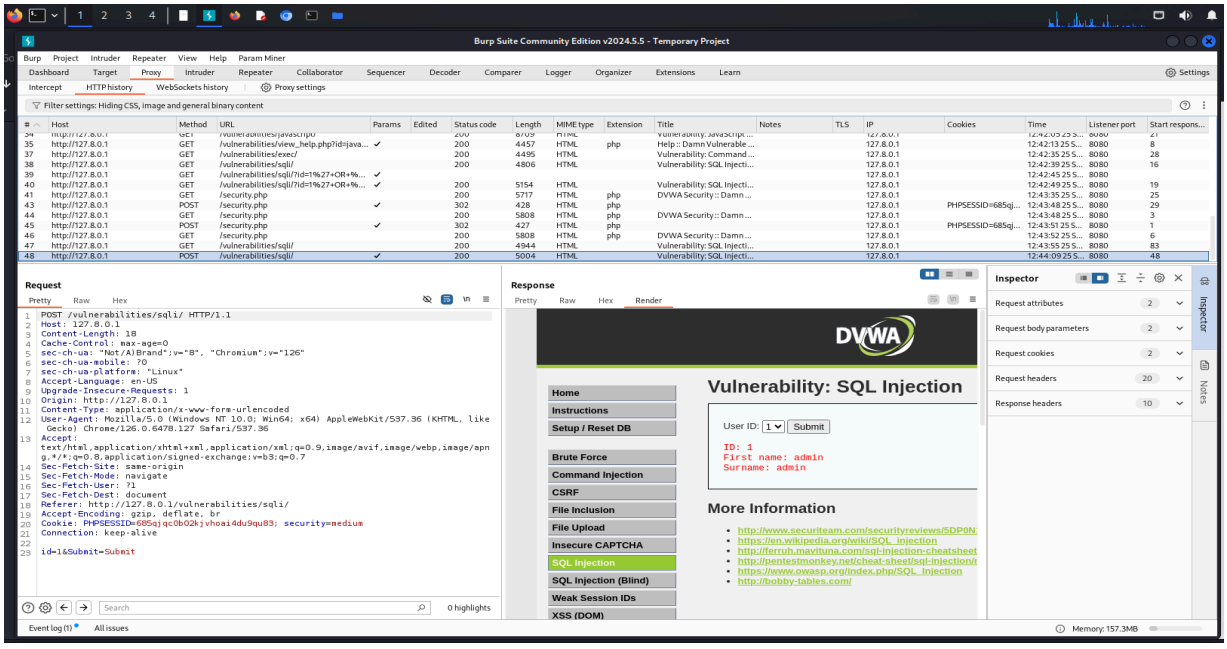
I next set the DVWA security level to Medium and attempted the test with an advanced payload.

2.2.1 Using Burp Suite

I captured the HTTP request using Burp Suite. I modified the `id` parameter in the request
to introduce a more complex SQL injection string



Screenshot 7

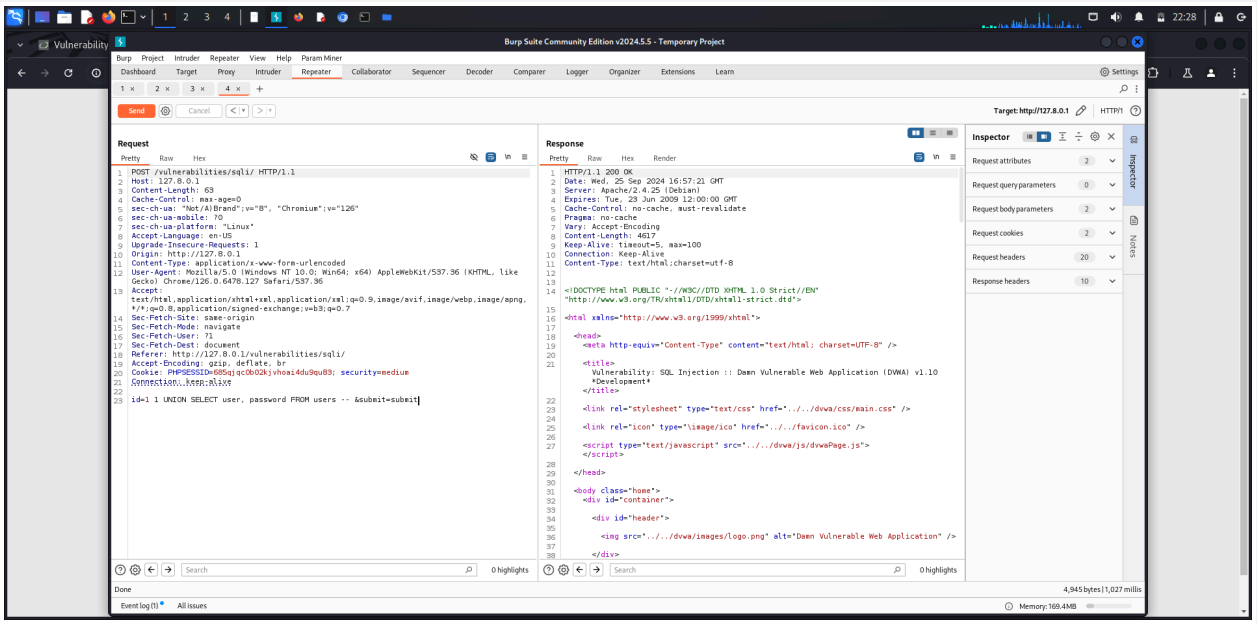


Screenshot 8

2.2.2 SQL Injection String

For example, the following payload was injected in to the `id` field :

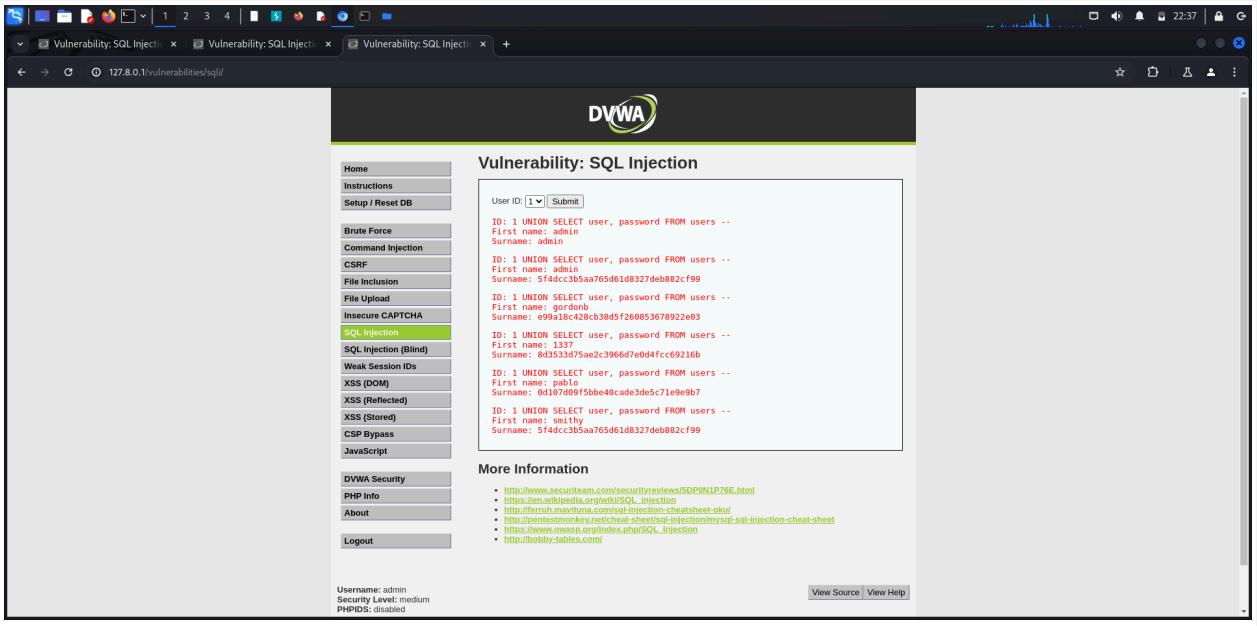
```
1 UNION SELECT user, password FROM users--
```



Screenshot 9

2.2.3 Execution

After editing the request in Burp Suite, I sent it to the server. From this, I received usernames and passwords as reflected from the system's response



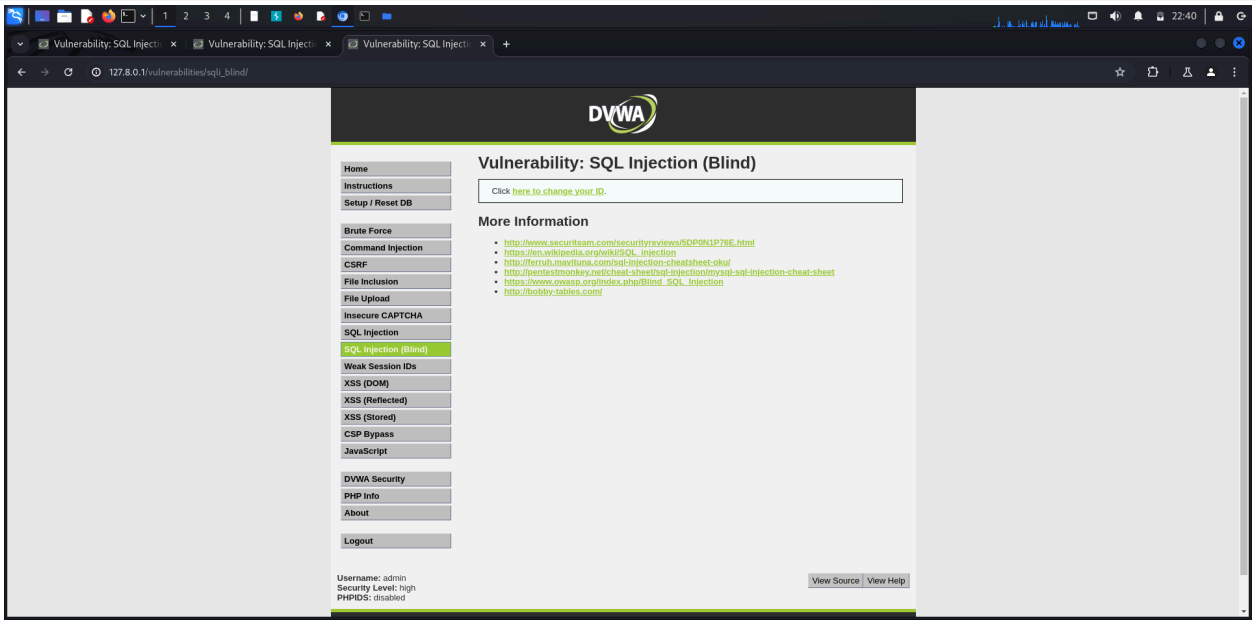
Screenshot 10

2.3 SQL Injection (High Security Level)

Last of all, I tried SQL injection at the High level of security.

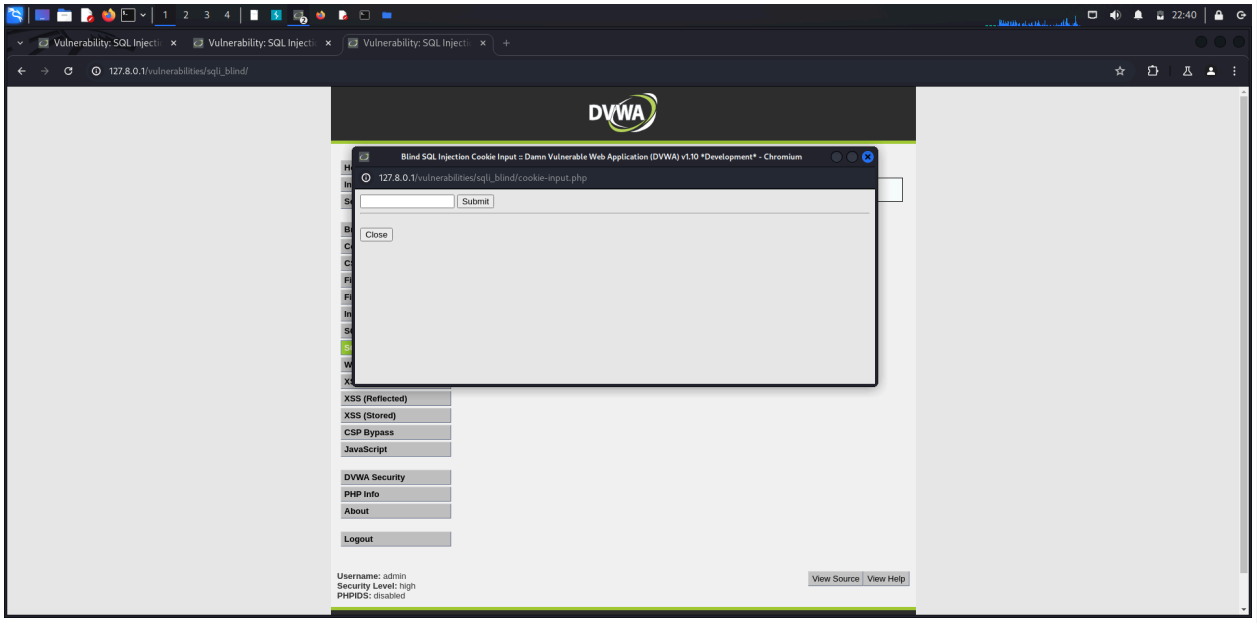
2.3.1: Find the injection point

There is a slight difference in the interface at High security level. On clicking on "Here to
"change your ID" button



Screenshot 11

a new window appeared where I could input SQL command.

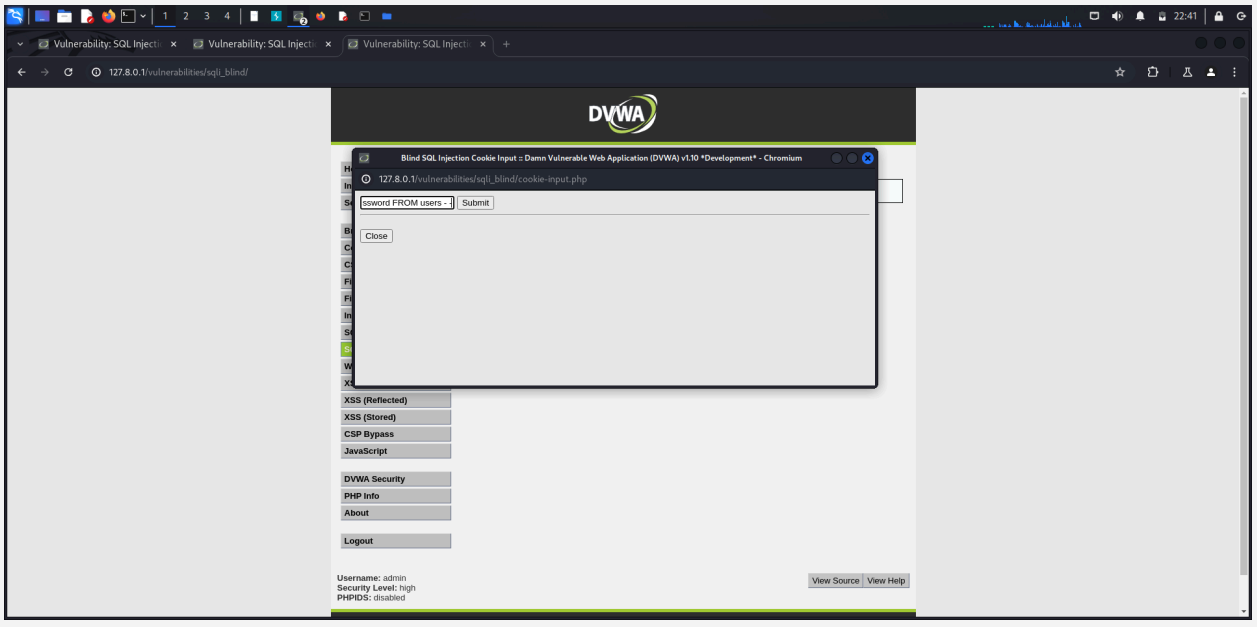


Screenshot 12

2.3.2 Injection Payload

I then inserted the following SQL injection string :

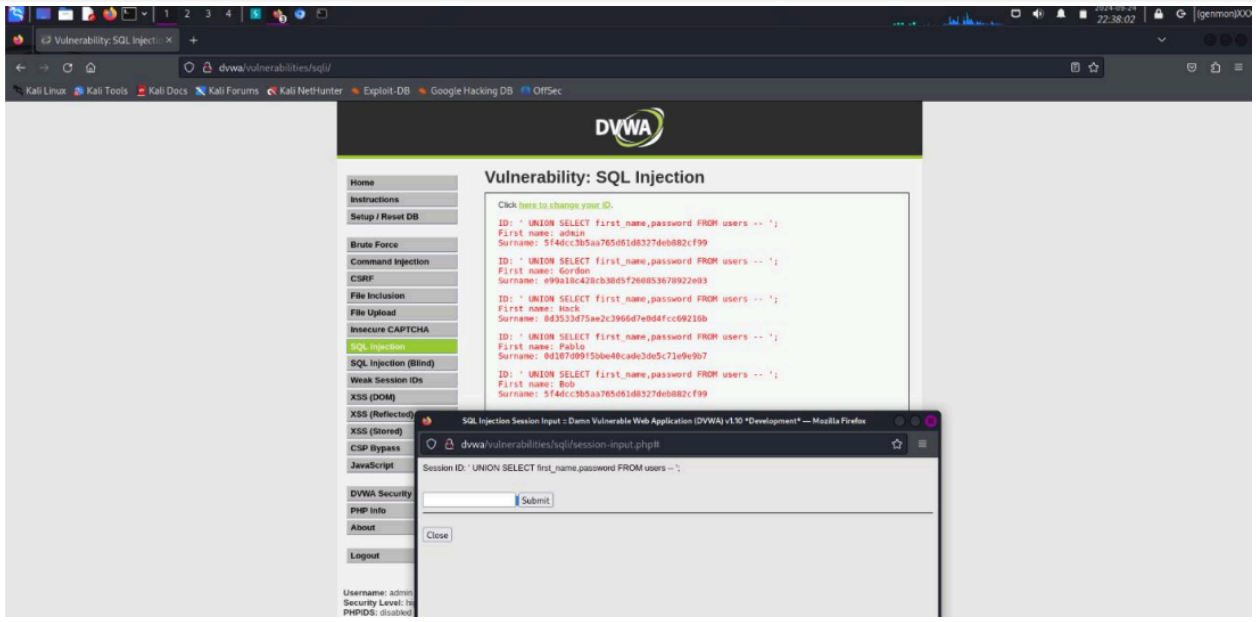
' UNION SELECT user, password FROM users –



Screenshot 13

2.3.3 Results

The system returned a list of usernames and passwords
after I injected it, signifying confirmation of vulnerability at the highest security setting.



Screenshot 14

Conclusion

Using Docker to successfully deploy DVWA, I tested for different types of SQL injection vulnerabilities across various security levels. Through basic and advanced forms of SQL injection techniques assisted by the use of a request interceptor in Burp Suite, I was able to extract sensitive data within the database for each security setting. From these experiments, I noticed the potential impact and effectiveness of SQL injection attacks for any security level.