

2024

# **Report: Installation and Usage of DVWA for SQL Injection Testing**

**ARUN KUMAR PK**

# CONTENTS

<b>Installation of DVWA using Docker .....</b>	<b>2</b>
<b>Performing SQL Injection on DVWA .....</b>	<b>2</b>
SQL Injection (Low Security Level) .....	4 SQL
Injection (Medium Security Level) .....	5 SQL
Injection (High Security Level) .....	6
<b>Conclusion .....</b>	<b>9</b>

# 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:

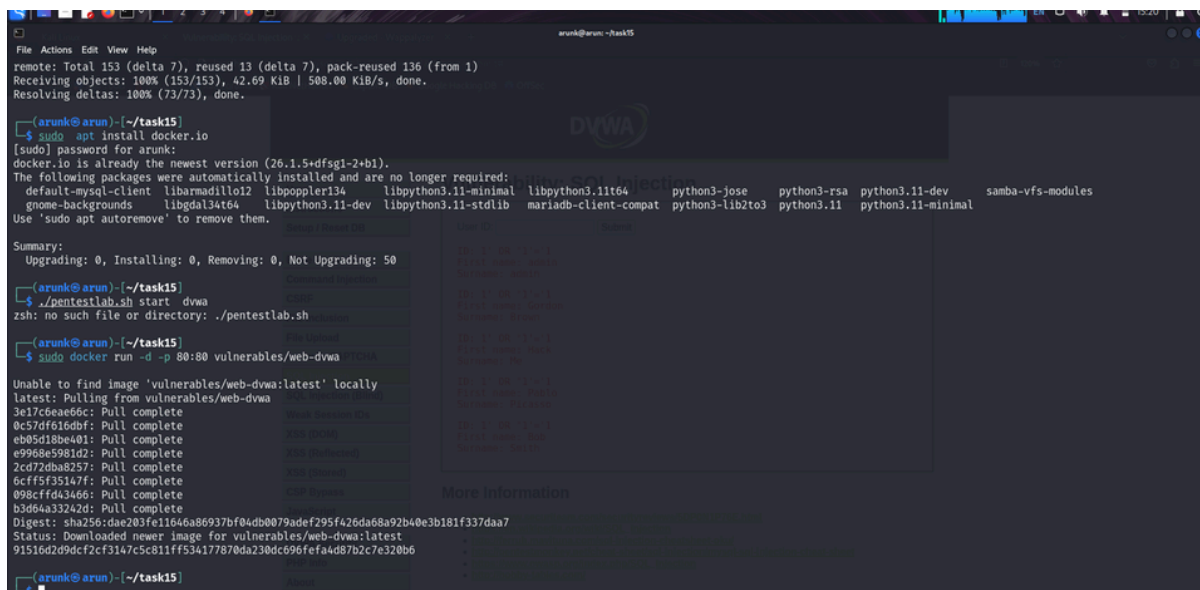
```
git clone https://github.com/eystsen/pentestlab.git
```

## 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
```



```
File Actions Edit View Help
remote: Total 153 (delta 7), reused 13 (delta 7), pack-reused 136 (from 1)
Receiving objects: 100% (153/153), 42.69 KiB | 508.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.

(arunk@arun) ~/task15
$ sudo apt install docker.io
[sudo] password for arunk:
docker.io is already the newest version (26.1.5+dfsg1-2+b1).
The following packages were automatically installed and are no longer required:
 default-mysql-client libarmadillo12 libpoppler134 libpython3.11-minimal libpython3.11t64 python3-jose python3-rsa python3.11-dev samba-vfs-modules
Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 50

(arunk@arun) ~/task15
$ ./pentestlab.sh start dvwa
zsh: no such file or directory: ./pentestlab.sh

(arunk@arun) ~/task15
$ sudo docker run -d -p 80:80 vulnerables/web-dvwa

Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6ae66c: Pull complete
0c57df61bdf: Pull complete
eb09d18b401: Pull complete
e998e981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cfd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe1164a86937bf04db0079ade295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
91516d2d9dcf2cf3147c5c811ff534177870da230dc96fefa4d87b2c7e320b6

(arunk@arun) ~/task15
$
```

Screenshot 1

## 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
```

```
└─$ cd pentestlab

(arunk@arun)-[~/task15/pentestlab]
└─$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Application
Adding dvwa to your /etc/hosts
127.0.0.1      dvwa was added succesfully to /etc/hosts
not set
Running command: docker run --name dvwa -d -p 127.0.0.1:80:80 vulnerables/web-dvwa
150723c9859c9a2b776fbf957493cd6c898def852429bd6ff228c51ed5b986b9
docker: Error response from daemon: driver failed programming external connectivity on endpoint dvwa (8417c31a3fe8baa059b5a7c16616949f7202de6326382cd486dc66e
2c6b95d44): Error starting userland proxy: listen tcp4 127.0.0.1:80: bind: address already in use.
DONE!

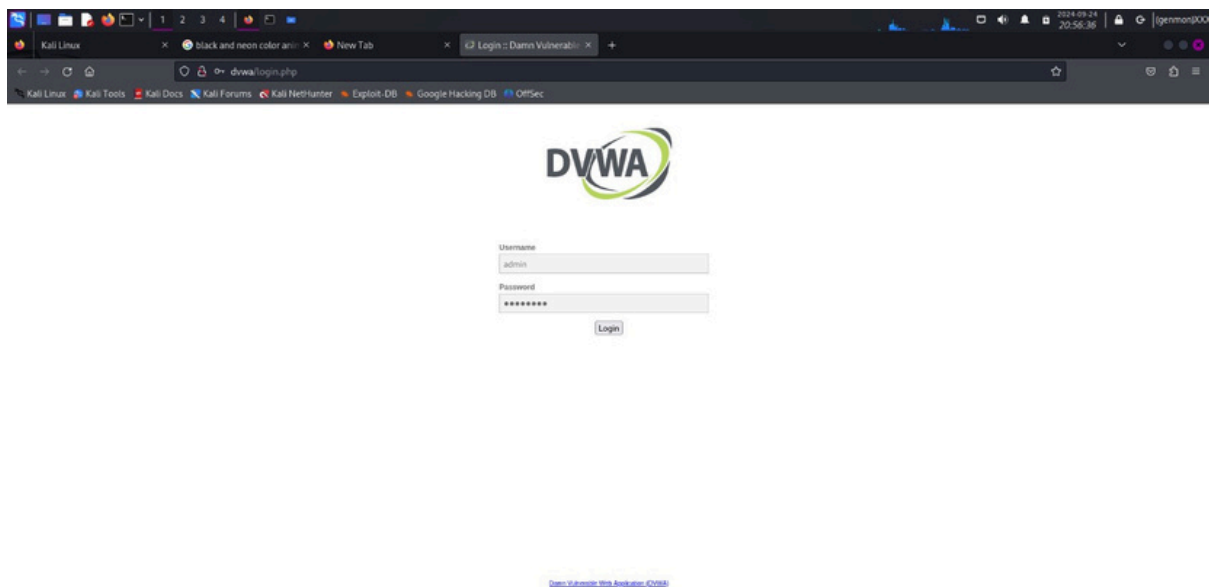
Docker mapped to http://dvwa or http://127.0.0.1

Default username/password:  admin/password
Remember to click on the CREATE DATABASE Button before you start

(arunk@arun)-[~/task15/pentestlab]
└─$
```

#### 1.4 Logging In At the login page, I used the default credentials:

- Username: admin
- Password: password



Screenshot 3

#### 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 (I missed capturing a screenshot of this step). 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

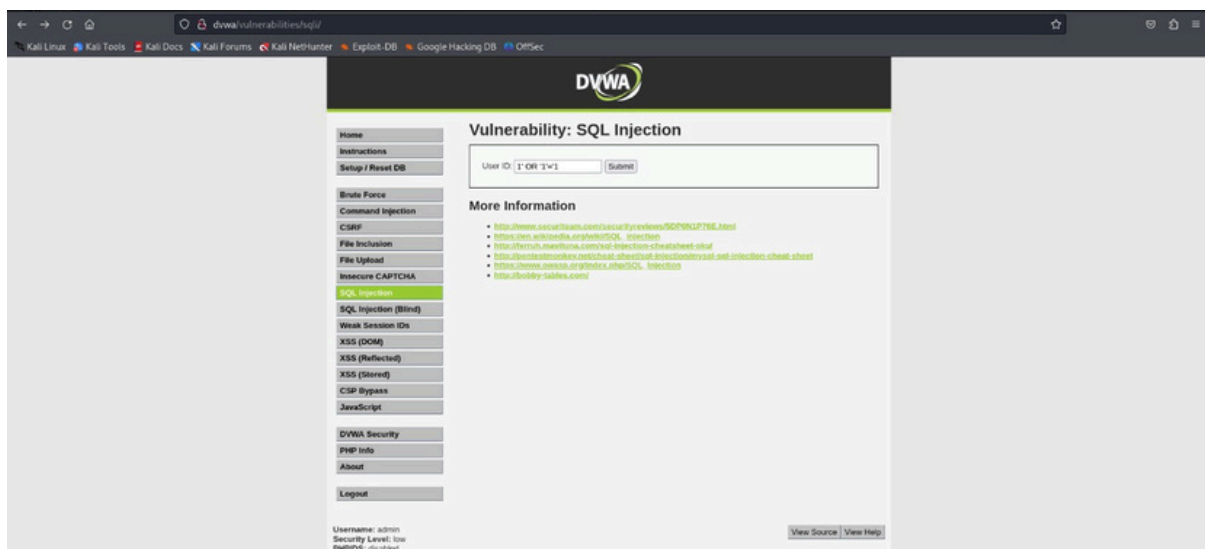
### 2.1 SQL Injection (Low Security Level)

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

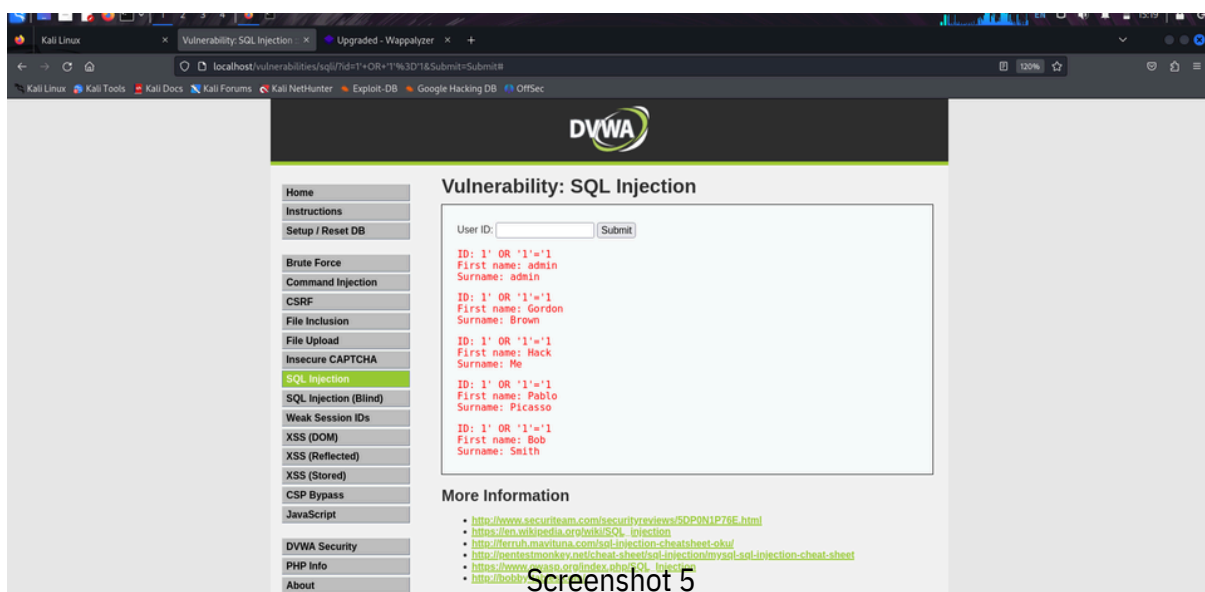
I used the following basic SQL injection string:

**1' OR '1'='1**



Screenshot 4

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



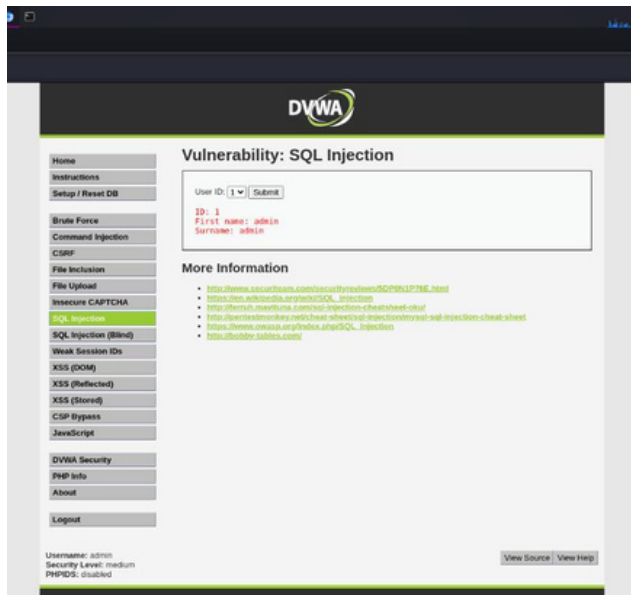
Screenshot 5

## 2.2 SQL Injection (Medium Security Level)

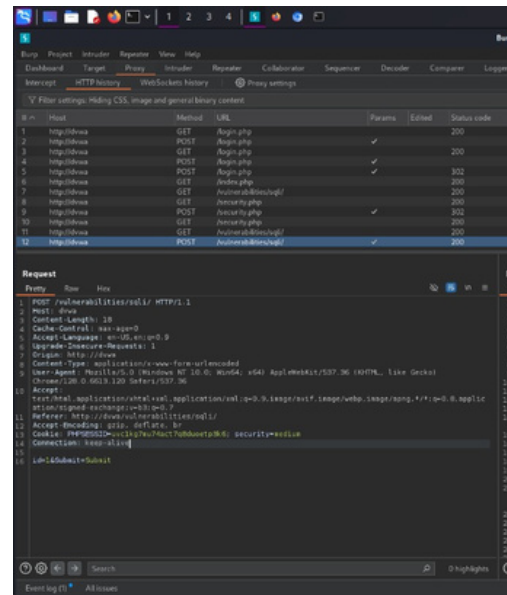
Next, I changed the DVWA security setting to Medium and conducted the test with an enhanced payload.

### 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.



Screenshot 6

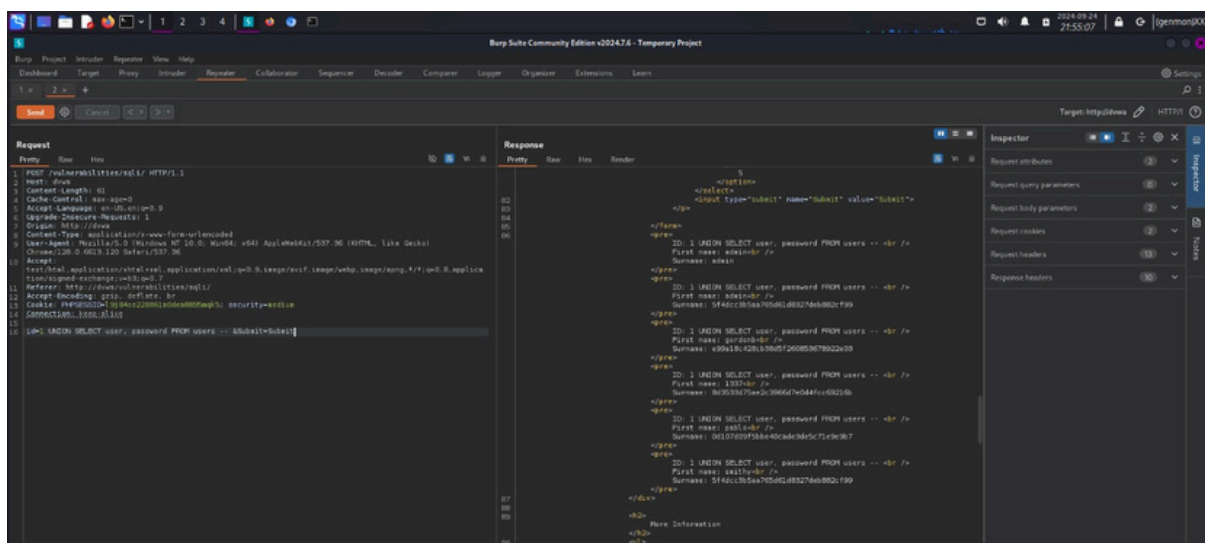


Screenshot 7

### 2.2.2 SQL Injection String

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

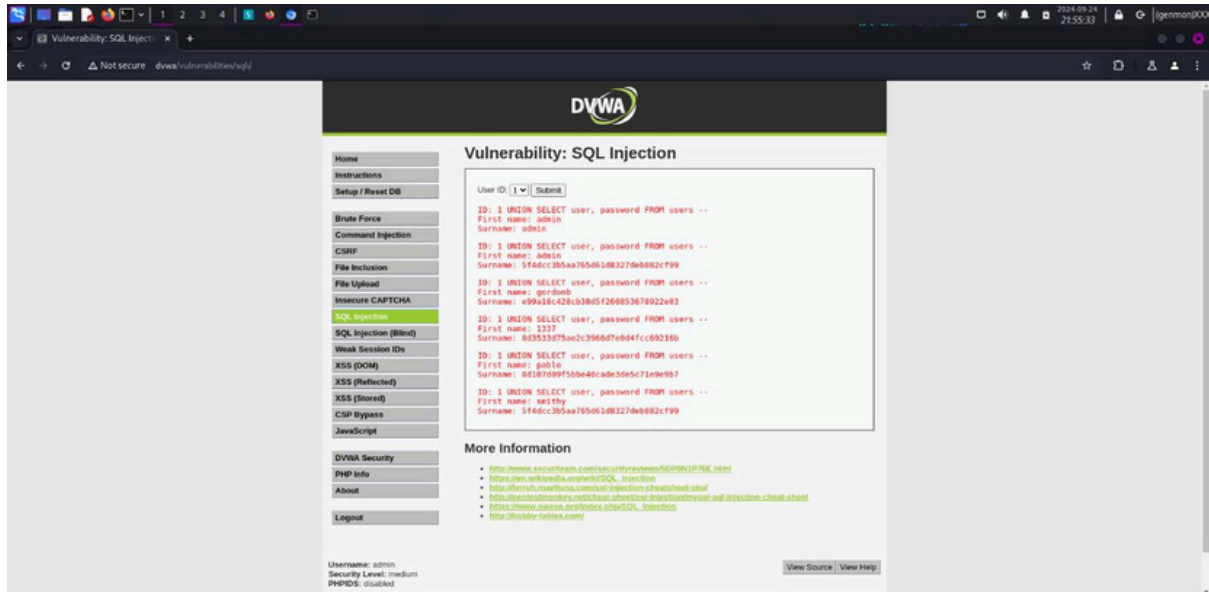
1 UNION SELECT user, password FROM users - -



Screenshot 6

### 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 (refer screenshot 6 and screenshot 7).



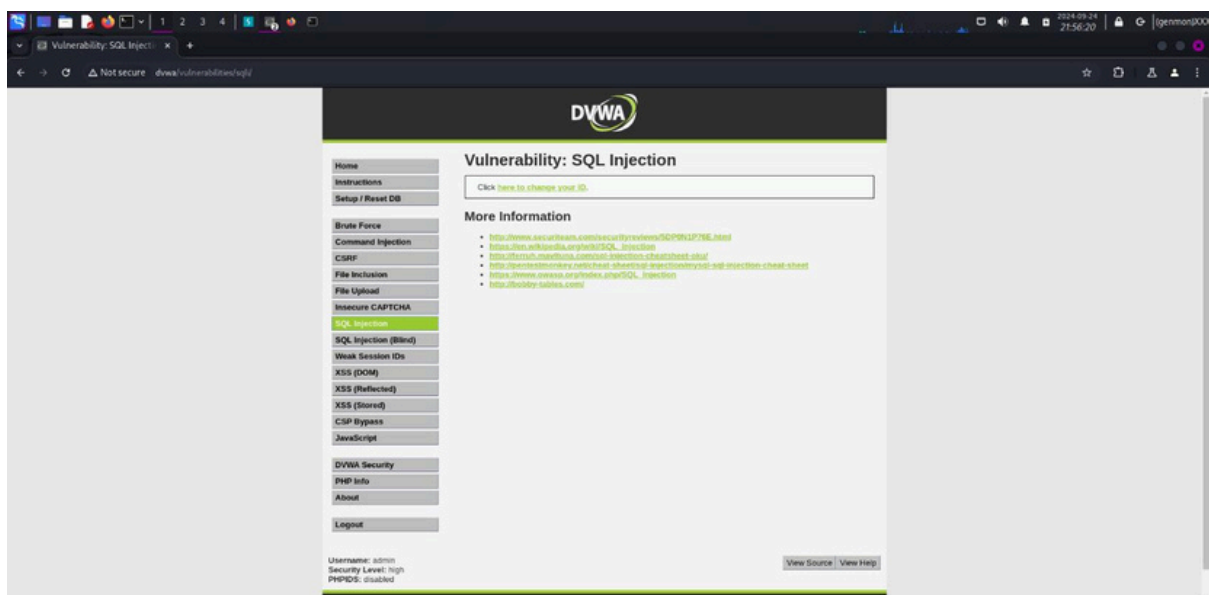
Screenshot 7

## 2.3 SQL Injection (High Security Level)

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

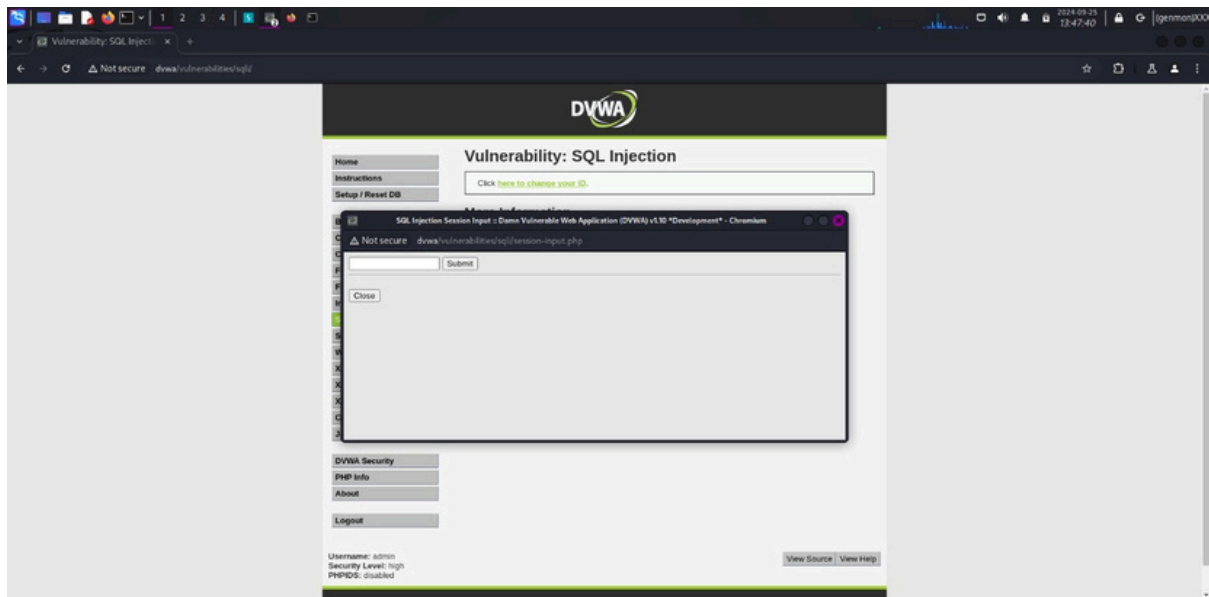
### 2.3.1 Identifying the Injection Point

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



Screenshot 8

a new window appeared where I could input SQL command.

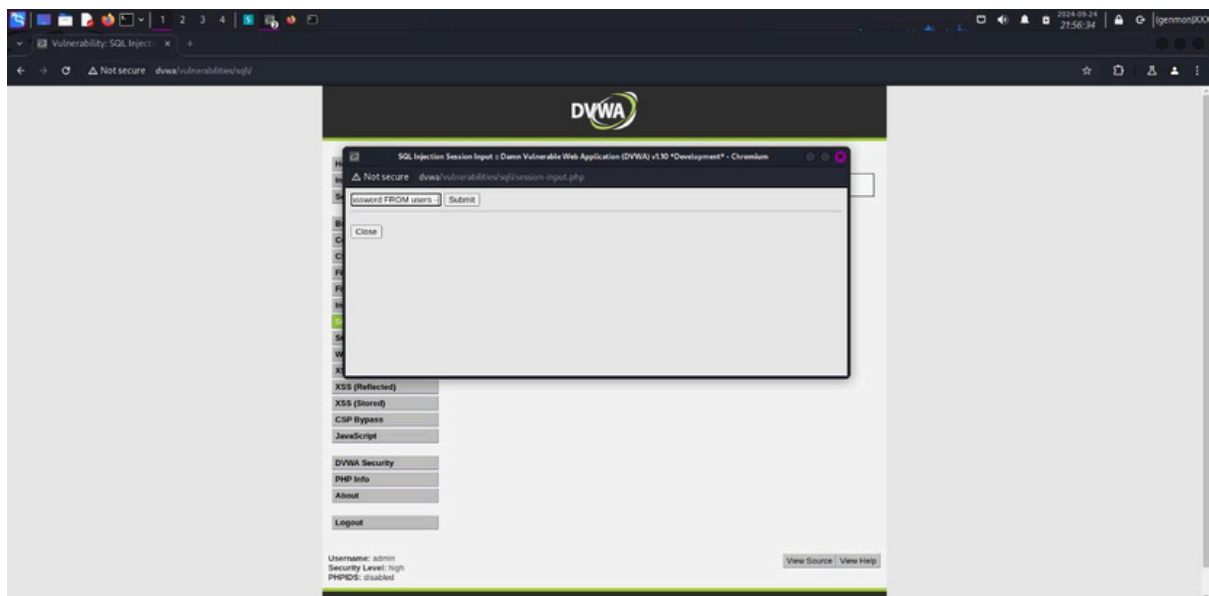


Screenshot 9

### 2.3.2 Injection Payload

I inserted the following SQL injection string:

**' UNION SELECT user, password FROM users - -**

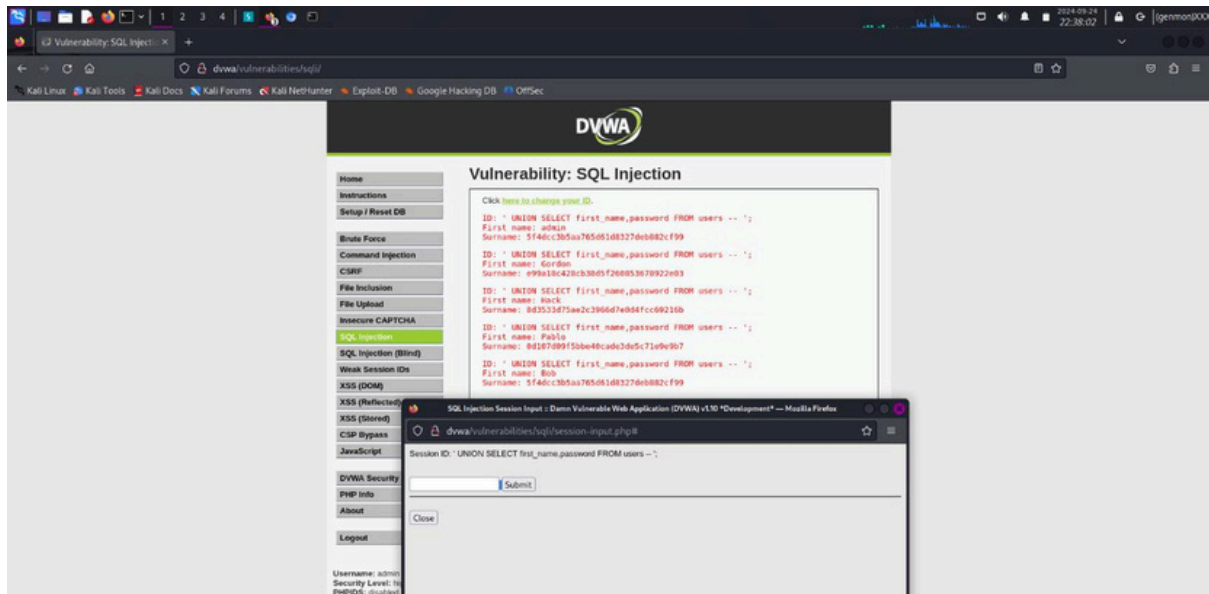


Screenshot 10



### 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.



Screenshot 11

## 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.