

# TASK 15 - DVWA REPORT

AN OVERVIEW OF THE SOLUTION TO THE SQL INJECTION LABS (LOW, MEDIUM, AND HIGH) IN DVWA.

**PREPARED BY:**  
**NAMITHA MANUVEL**



# INDEX

<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>2. DVWA INSTALLATION.....</b>	<b>2-4</b>
<b>3. LOW LEVEL.....</b>	<b>5-6</b>
<b>4. MEDIUM LEVEL.....</b>	<b>7-9</b>
<b>5. HIGH LEVEL.....</b>	<b>10-11</b>
<b>6. CONCLUSION.....</b>	<b>12</b>

# INTRODUCTION

This report provides a detailed analysis of the SQL Injection vulnerabilities identified in the **Damn Vulnerable Web Application (DVWA)**. DVWA is a deliberately insecure web application created for security professionals to test their skills and tools in a safe, legal environment. The focus of this report is to demonstrate the exploitation of SQL Injection vulnerabilities at three different security levels—Low, Medium, and High—and to outline the steps taken to achieve successful injections. Additionally, the report will highlight the security mechanisms in place at each level and propose mitigation strategies to defend against such attacks in real-world web applications.

# DVWA INSTALLATION

To access DVWA , I utilised a GitHub repository named pentestlab. By following the below steps I cloned the repository:

## 1. Create the desired directory

Create and change to the pentestlab directory where you want to clone DVWA.

- **mkdir pentestlab**
- **cd pentestlab**

## 2. Clone the DVWA Repository

Use the git clone command to clone the DVWA repository from GitHub.

**git clone <https://github.com/eystsen/pentestlab.git>**

## 3. Listing the contents

To view the available files and scripts in the pentestlab directory, the ls command was executed: **ls**  
This revealed the presence of the main script, pentestlab.sh, which is used to list and manage various vulnerable applications.

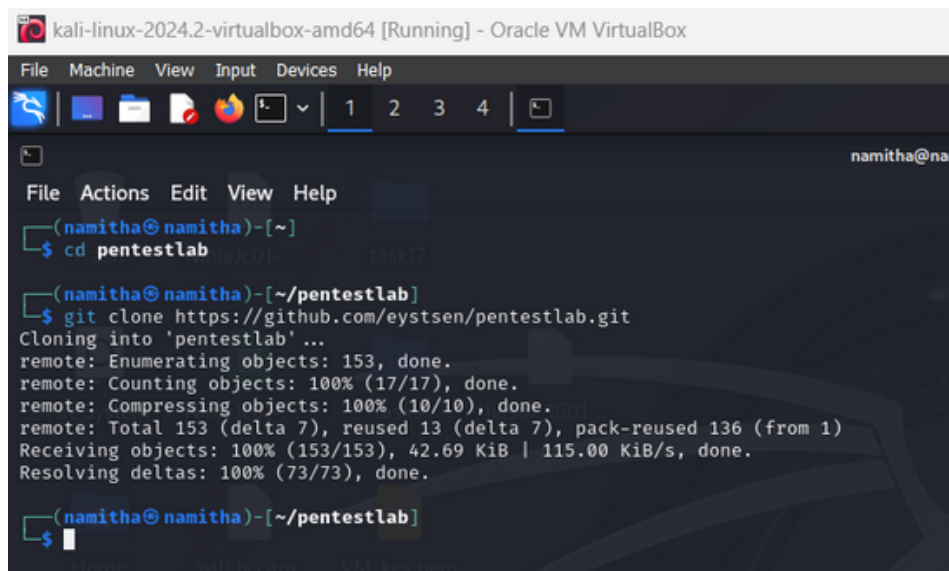
# CONT.....

## 4. Starting the DVWA Lab

To start the DVWA instance, the following command was executed:

**`./pentestlab.sh start dvwa`**

This command deployed DVWA, along with the required services (Apache, MySQL, PHP)



```
kali-linux-2024.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
namitha@namitha
File Actions Edit View Help
(namitha@namitha)-[~]
$ cd pentestlab
(namitha@namitha)-[~/pentestlab]
$ 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 | 115.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.
(namitha@namitha)-[~/pentestlab]
$
```

Repository Cloning

```
(namitha@namitha)-[~/pentestlab] module gd: installed
$ ls
install_docker_kali_x64.sh  pentestlab.sh  README.md
(namitha@namitha)-[~/pentestlab] username: app
password: *****
$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Application : 127.0.0.1
dvwa already exists in /etc/hosts
Running command: docker start dvwa
dvwa
DONE!
[User: www-data] Writable folder /var/www/html/hackable/uploads
[User: www-data] Writable file /var/www/html/external/phpids/0.6.0/
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
(namitha@namitha)-[~/pentestlab]
$
```

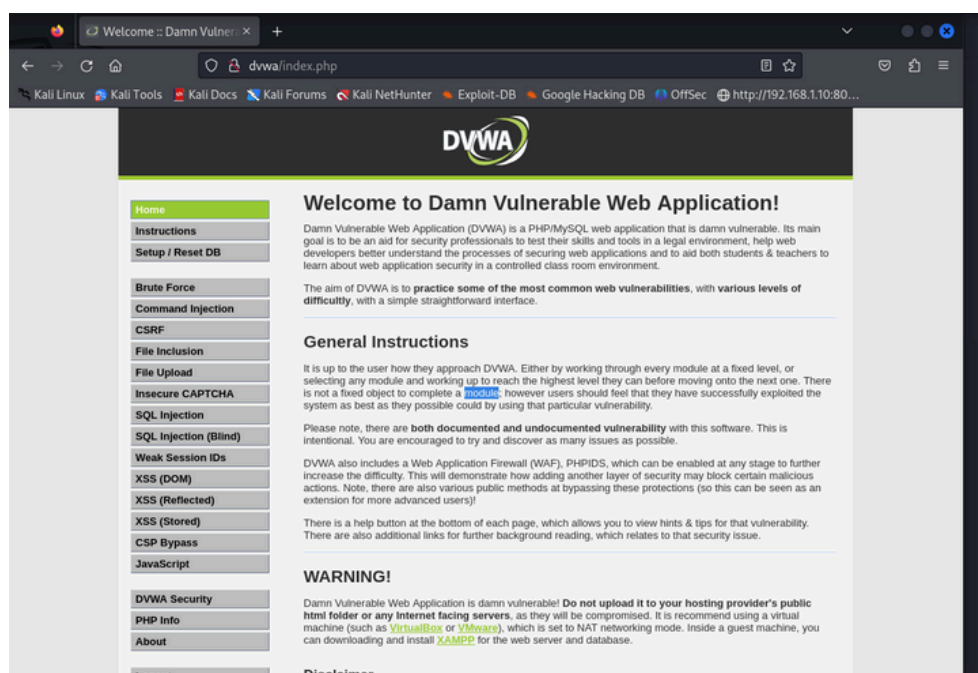
## Accessing DVWA

DVWA Login page can be now accessed at :

**http://DVWA/**

using the credentials :

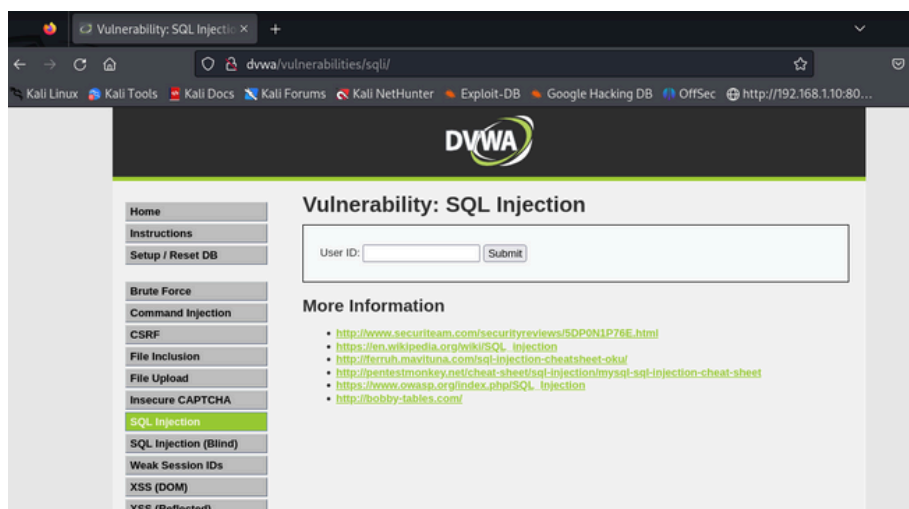
- **Username: admin**
- **Password: password**



## DVWA Home Page

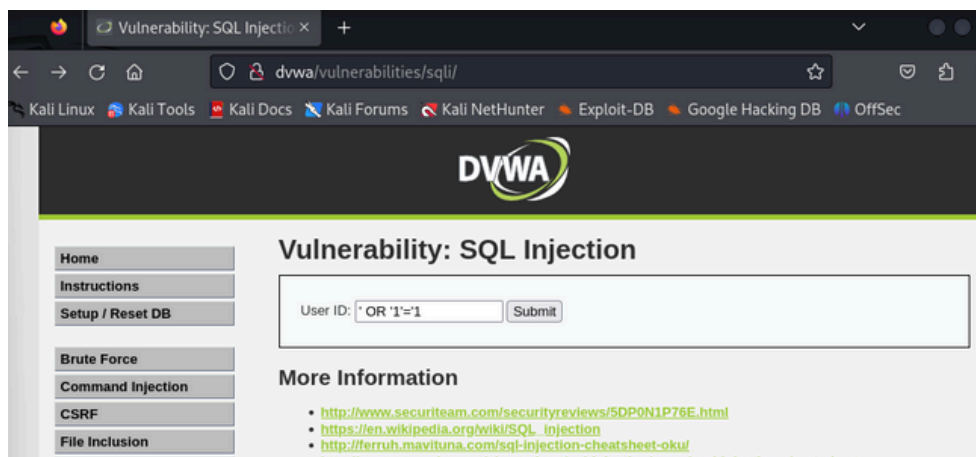
# LOW LEVEL SQL INJECTION

SQL injection was by default at the Low security level. There was a field where we could inject SQL queries.

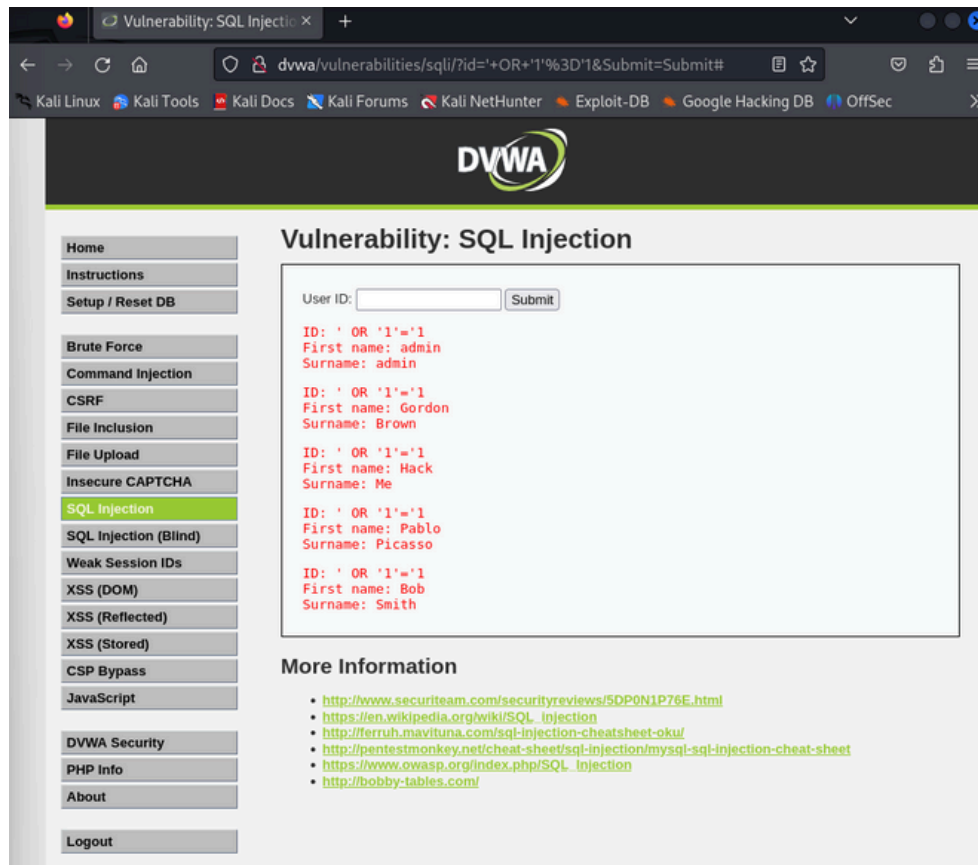


## Low Level SQL Injection

I proceeded with a simple exploitation query: **' OR '1'='1** into the User ID field to bypass login and retrieve database information.



## Query Injection

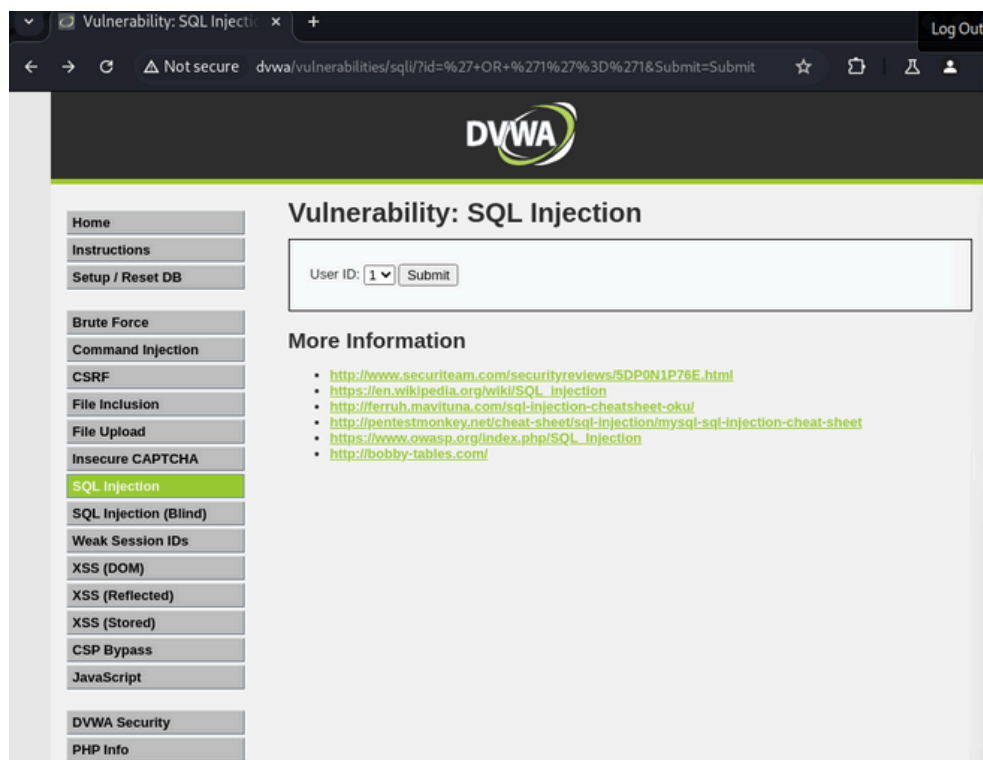


Result



# MEDIUM LEVEL SQL INJECTION

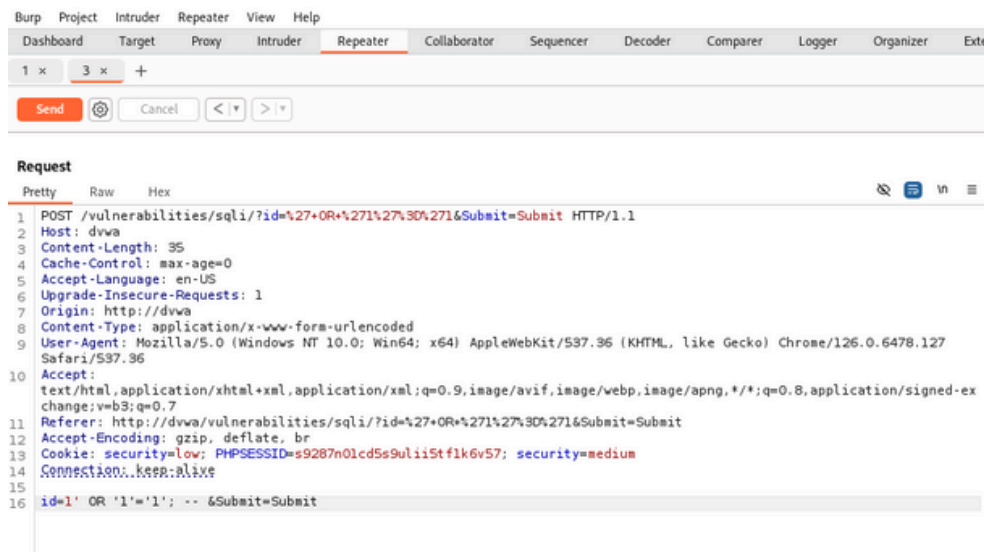
In medium level , I used burp suite to do SQL injection. So I opened browser in burp suite and accessed DVWA there. After logging in , I entered a random code and submitted it. After this , I went to http history in burp suite and found a get request which I got after submitting the data. In the request the security was low. So I changed it into medium and send it. Then I opened the response of the request in browser to receive the medium level task page .



# CONT.....

So in the page , I selected 1 and submit it. Then went back to burp suite which I found a post request . And I send it to repeater and edited the security to medium from low. Then I inserted this particular code in the place of id :

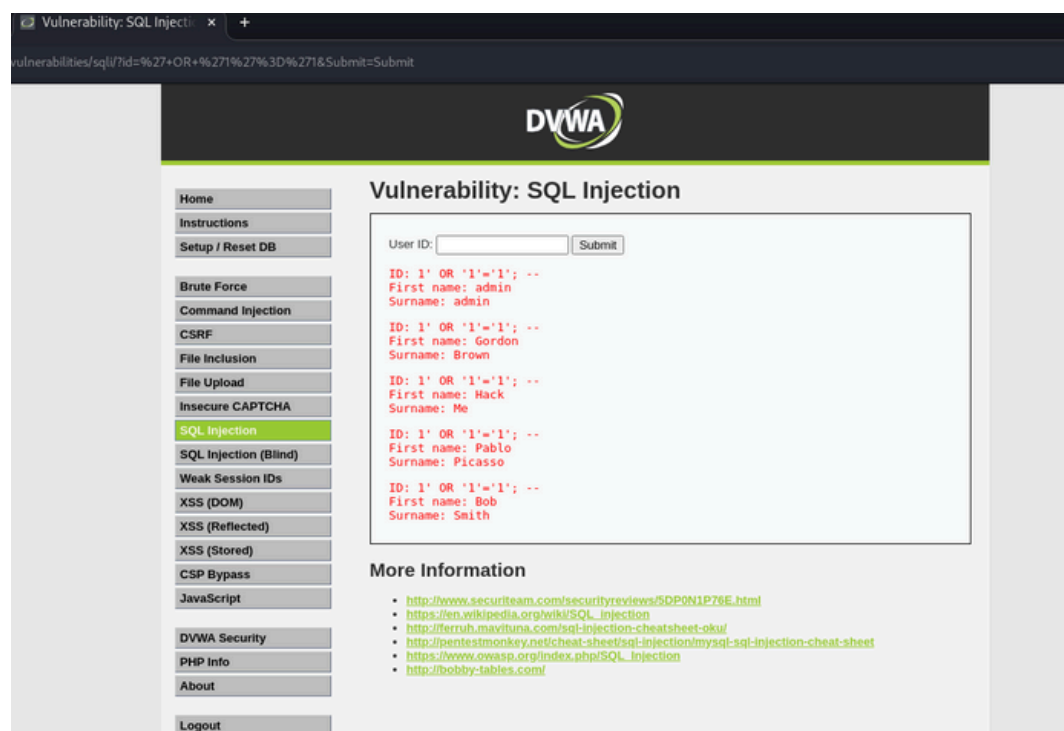
**1' OR '1'='1' ; --**



Post Request Changes

# CONT.....

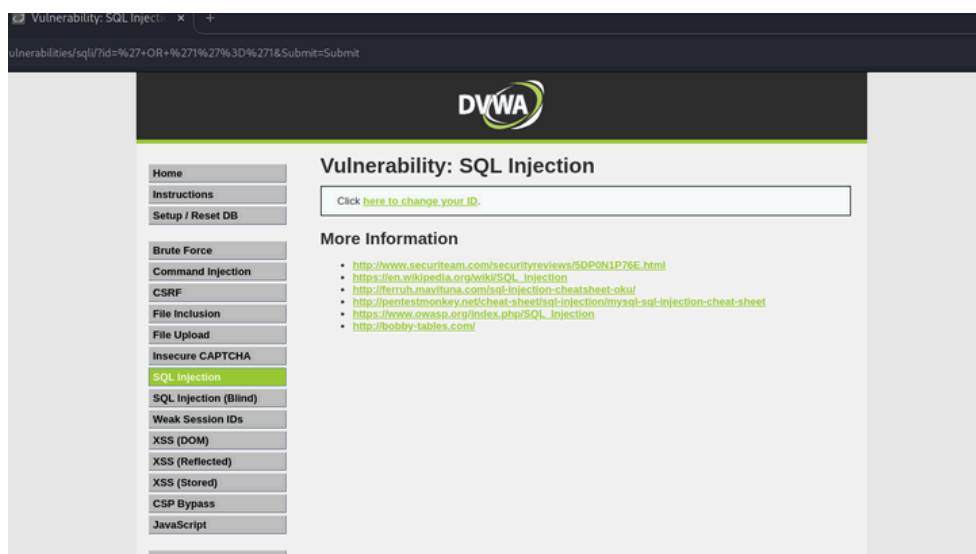
In the response I was able to get the details of the users.



## Result

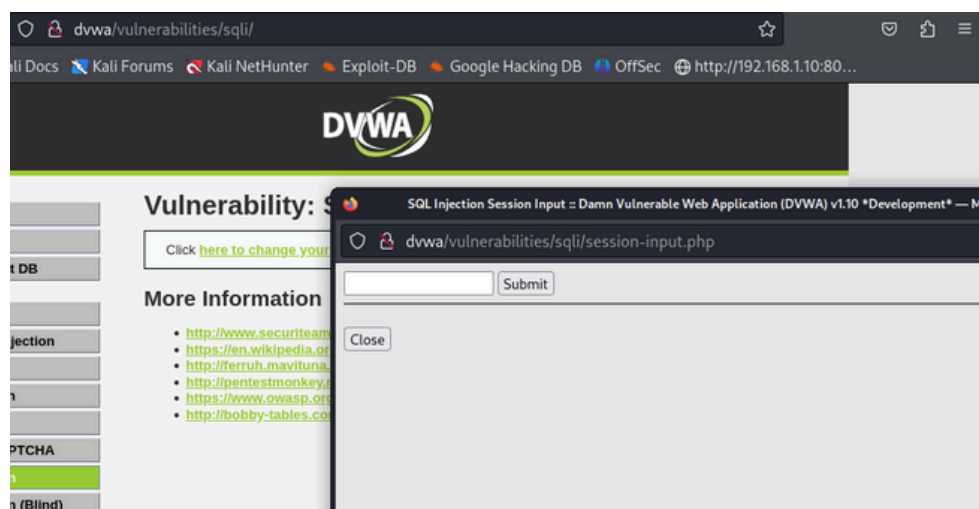
# HIGH LEVEL SQL INJECTION

For testing SQL injection on the High security level, I changed the security level to high in DVWA Security and received the following page.



## High Level SQL Injection

By clicking on the provided link a new page will come where we can inject code into it.



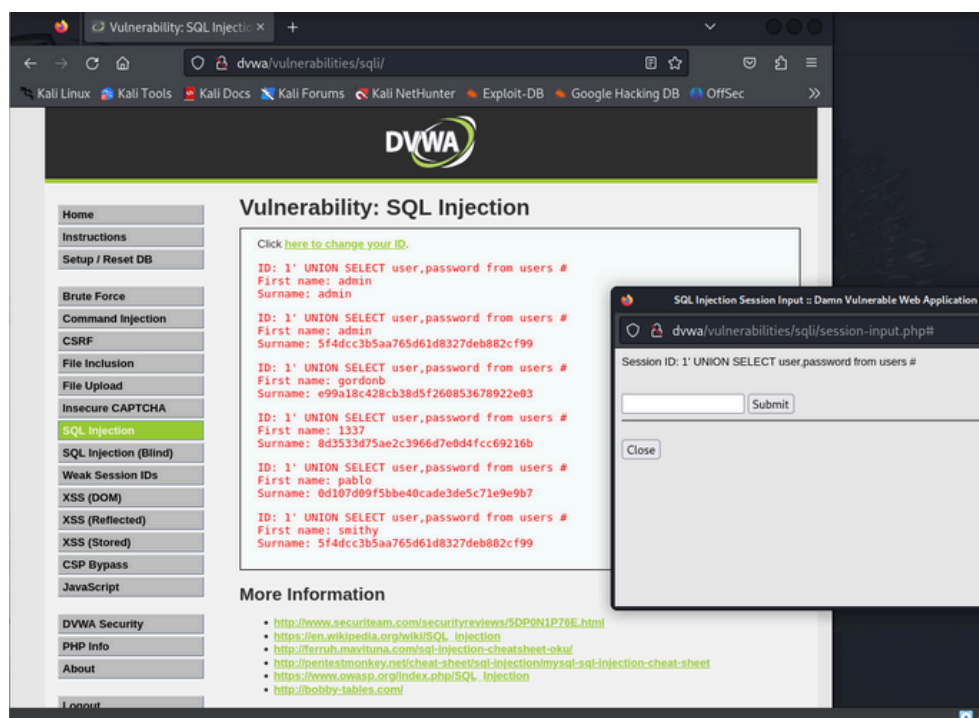
## Injection Point

# CONT.....

To get more information , I injected a code :

**1' UNION SELECT user,password from users #**

After this , I was able to get more details on users.



Result

# CONCLUSION

In this task, the SQL Injection vulnerabilities within DVWA were successfully exploited across three different security levels: Low, Medium, and High. By systematically analyzing the security defenses at each level, we demonstrated how attackers could manipulate poorly sanitized input fields to extract sensitive information from a database. The report highlighted the importance of input validation and proper sanitization techniques to mitigate SQL injection attacks. Furthermore, it underscored the need for adopting secure coding practices and implementing layered security measures in real-world applications to prevent such vulnerabilities from being exploited.