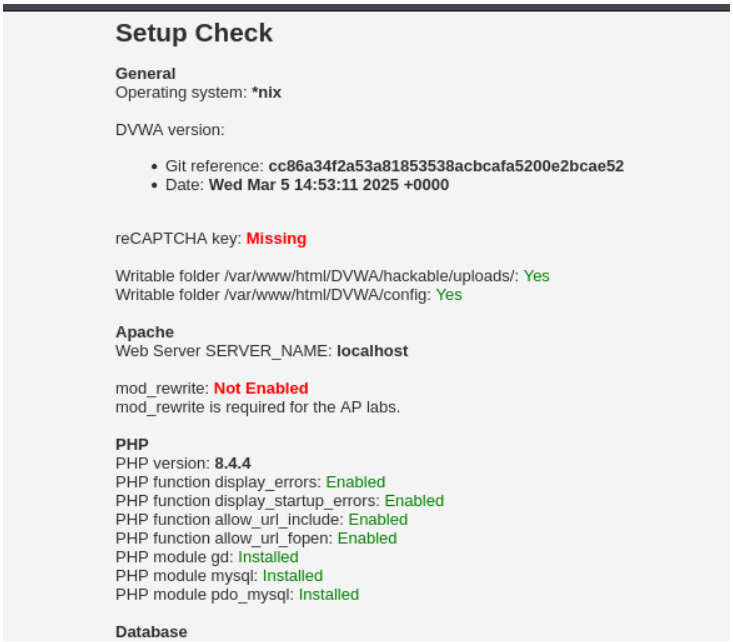
Abdullrahman Alghanim

Lab 8 Database Hacking

CIS 4204

03/15/2025



PHP module mysqli: Installed
PHP module pdo_mysql: Installed

Kali NetHunter
https://www.kali.org/kali-nethunter/

Database
Database username: **dvwa**
Database password: ******
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

API
This section is only important if you want to use the API module.
Vendor files installed: **Not Installed**

For information on how to install these, see the **README**.

***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.

Create / Reset Database

- 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)

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

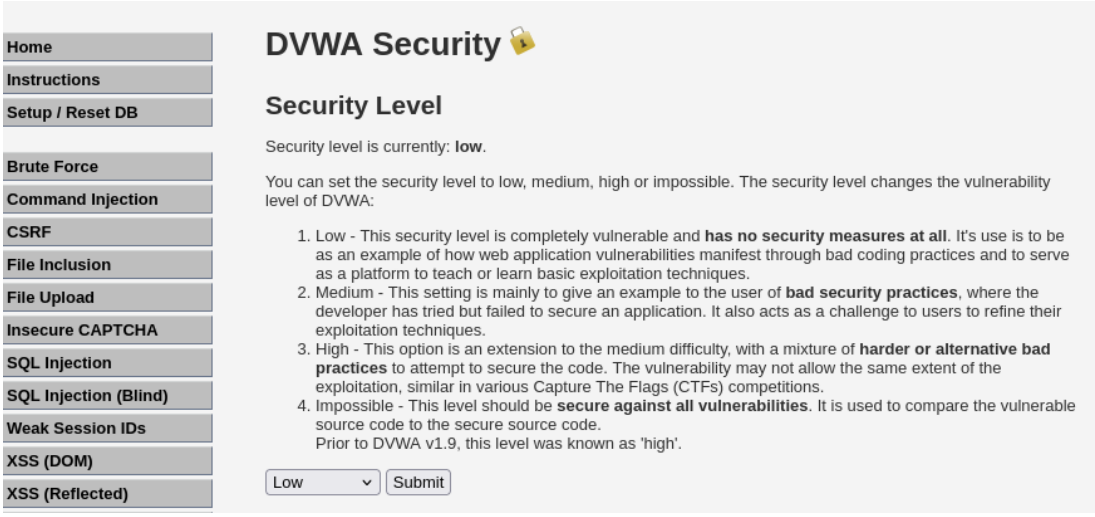
General Instructions

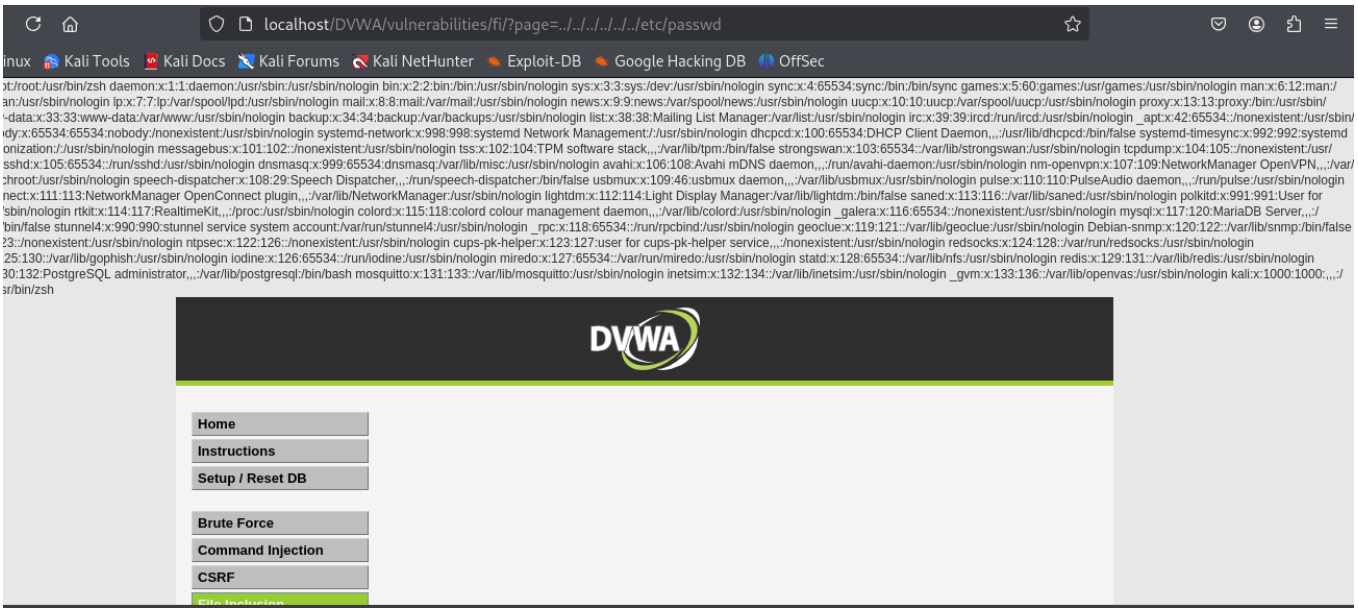
It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

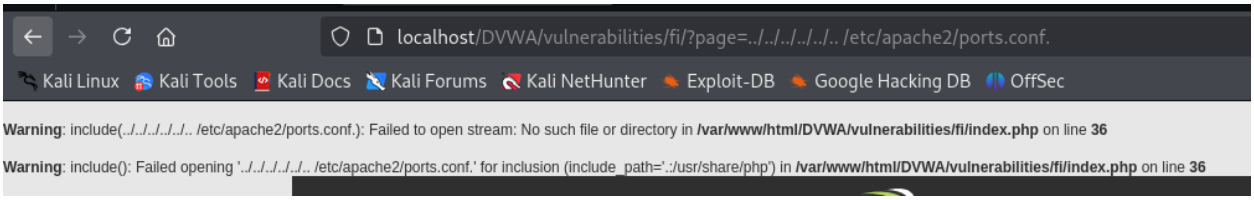
Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

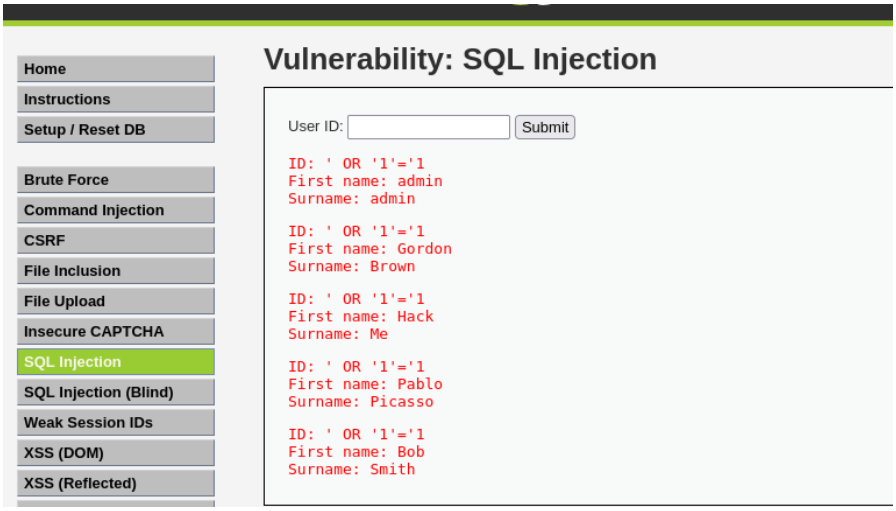
WARNING!







The ports that are configured are a great amount and you can see the difference between the two photos and how much of a difference in the ports there is



The reason 'OR '1'='1 is effective in an SQL injection attack lies in how it manipulates the logic of a database query. When user input is not properly sanitized, attackers can insert specially crafted strings that alter the intended behavior of a SQL statement. The phrase 'OR '1'='1 is a classic example that forces a conditional statement to always evaluate as true. By doing this, an attacker can potentially bypass login credentials or access data they are not authorized to see. The expression '1'='1 is always true, so when it is injected into a query, it tricks the database into thinking the condition has been met, even if the user provides incorrect information.

This form of attack is known as SQL injection and is considered one of the most dangerous vulnerabilities in web applications. According to the Open Web Application Security Project (OWASP), SQL injection is consistently ranked among the top security threats due to its simplicity and severe impact. The MITRE Corporation also classifies this issue under CWE-89, which refers to improper neutralization of special elements used in SQL commands. The attack works because the application fails to distinguish between user data and command syntax, allowing input to interfere with how the query operates. To prevent this, developers are encouraged to use secure coding practices such as parameterized queries and input validation.

Sources:

- OWASP. (2023). *SQL Injection*. Retrieved from https://owasp.org/Top10/A01_2021-Broken_Access_Control/

- MITRE. (2023). *CWE-89: Improper Neutralization of Special Elements used in an SQL Command*. Retrieved from https://cwe.mitre.org/data/definitions/89.html