

DVWA Security Level Comparison Project Report

Topic: DVWA Security Level Comparison Project

Name: Yatish S

Date: 30-09-2025

Contents:

1. Introduction
2. Environmental Setup and Tools
3. Vulnerabilities Selected
4. Procedure and Observation
5. Conclusion

1.Introduction

The Damn Vulnerable Web Application (DVWA) is an open-source web application intentionally designed with security vulnerabilities.

Programmed in PHP and utilizing a MySQL database, DVWA serves as a legal and ethical platform for security professionals, developers, and students to learn about and practice web application security concepts in a controlled environment. Its primary objective is to provide a safe space to test penetration testing skills, understand how vulnerabilities are exploited, and learn how to implement defensive measures without compromising a live system.

2.Setup and Tools

- Hosting DVWA On Local VM



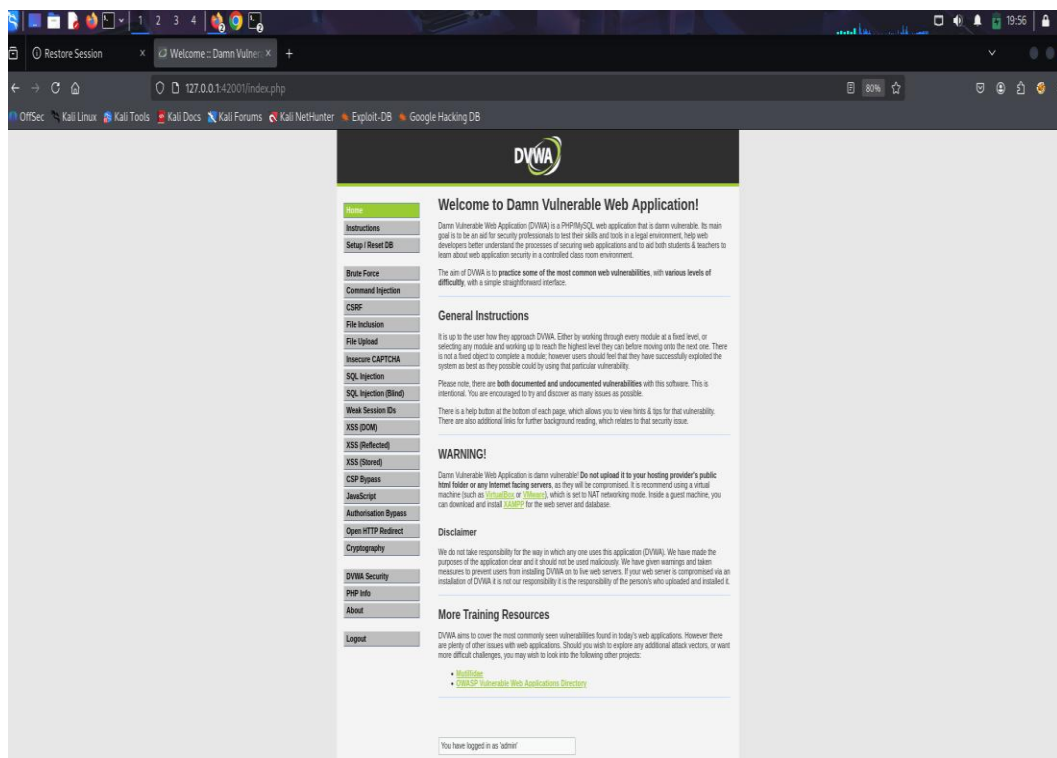
Screenshot 1:

- It is a traditional login page of DVWA and Configuration dashboard.

3. Selecting Vulnerabilities

We have to choose two vulnerabilities to demonstrate attack mechanism and defense strategies:

- **SQL Injection:** Injecting malicious SQL queries to gain unauthorized database information to the website like username or passwords that are stored in the database.
- **Reflected Cross-Site Scripting[XSS]:** Inject malicious JS payloads through URL's or from inputs which victims browser reflects.



Screenshot 2:

- The above interface displays [SQL Injection] and [XSS Reflected] Vulnerabilities.

4. Testing Procedure and Observations

- **SQL Injection**

SQL Injection was tested across various security level by giving increasingly filtered SQL payloads and observing outputs.

When DVWA Security is set to Low

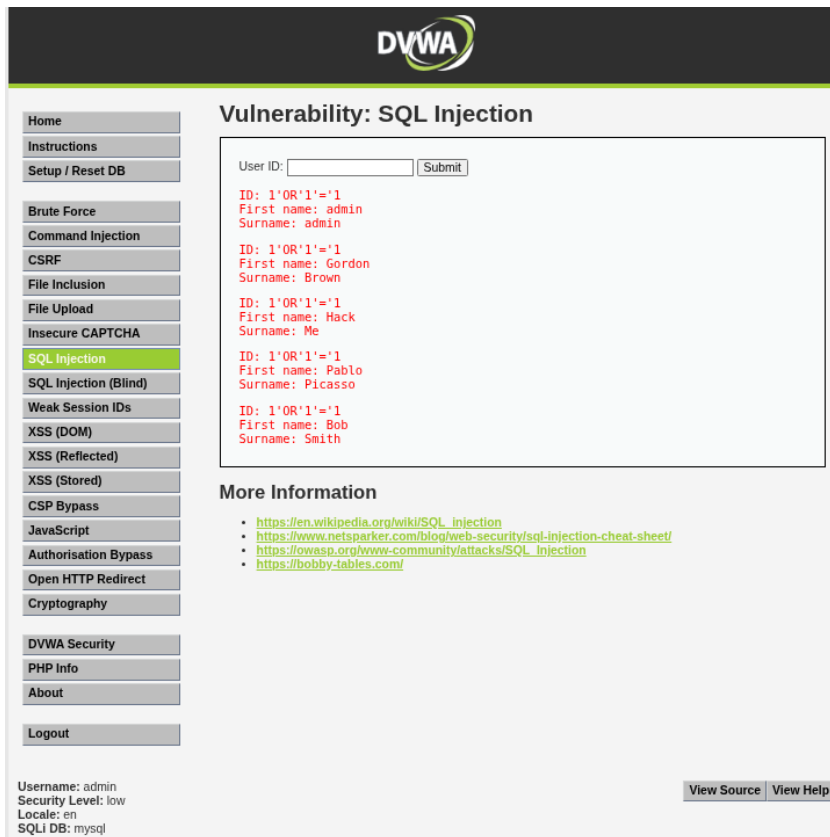
- **Payload:**

1' OR '1'='1

- **Outcome:** No sanitization or filters are applied so the data is leaked from the Database.

The screenshot shows the DVWA interface. The main heading is 'Vulnerability: SQL Injection'. Below it is a form with 'User ID:' and a 'Submit' button. The output of the query is displayed in red text: 'ID: 1', 'First name: admin', and 'Surname: admin'. Under 'More Information', there are four links to external resources about SQL injection. The left sidebar has a menu with items like 'Home', 'Instructions', 'Setup / Reset DB', 'Brute Force', 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection' (highlighted), 'SQL Injection (Blind)', 'Weak Session IDs', 'XSS (DOM)', 'XSS (Reflected)', 'XSS (Stored)', 'CSP Bypass', 'JavaScript', 'Authorisation Bypass', 'Open HTTP Redirect', 'Cryptography', 'DVWA Security', 'PHP Info', 'About', and 'Logout'. At the bottom, the status bar shows 'Username: admin', 'Security Level: impossible', 'Locale: en', and 'SQLi DB: mysql'. There are also 'View Source' and 'View Help' buttons.

Easily Bypassed the database



Screen Shot 4: Using `1'OR'1'='1` logical expression to manipulate the logic of SQL query.

Medium Security

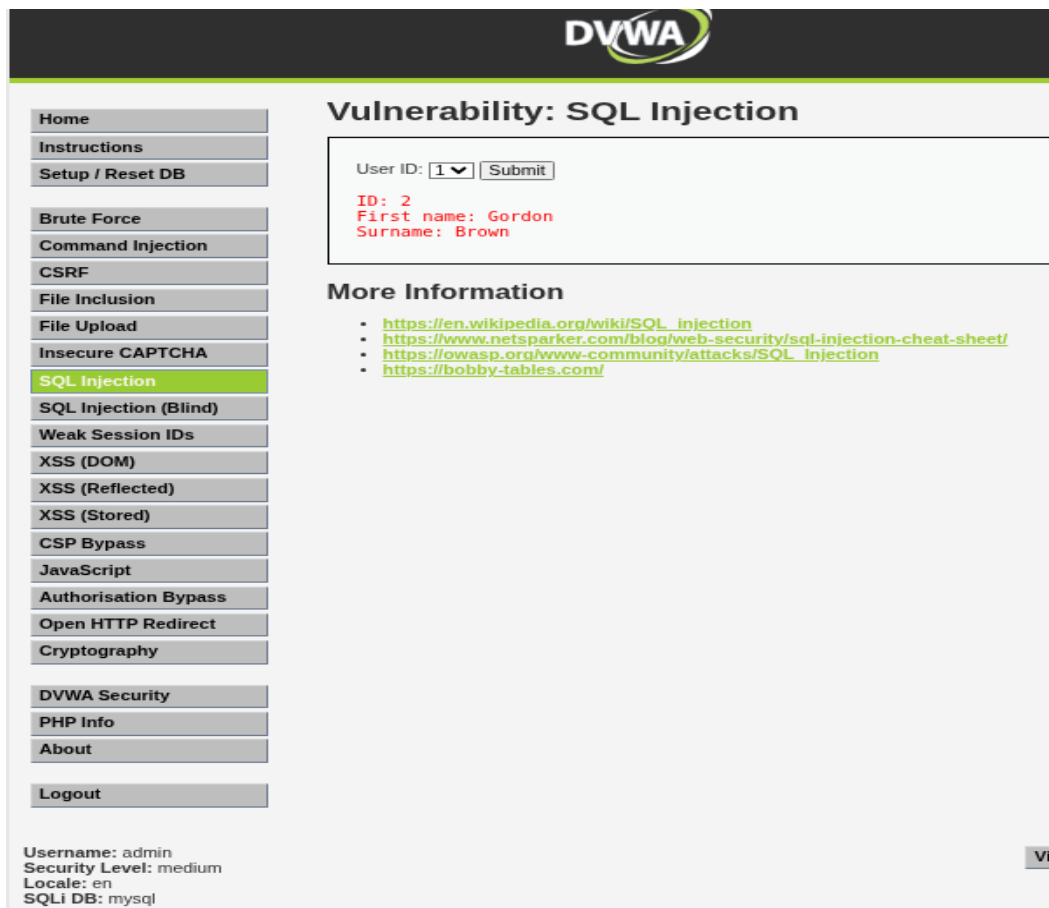
- **Payload**

1 UNION SELECT user, password FROM users#

- **Outcome:** In medium security level there might be partially vulnerable, leaks data on specific conditions.

Moderate input validation implemented, allows some bypasses for UNION-Based SQLi.

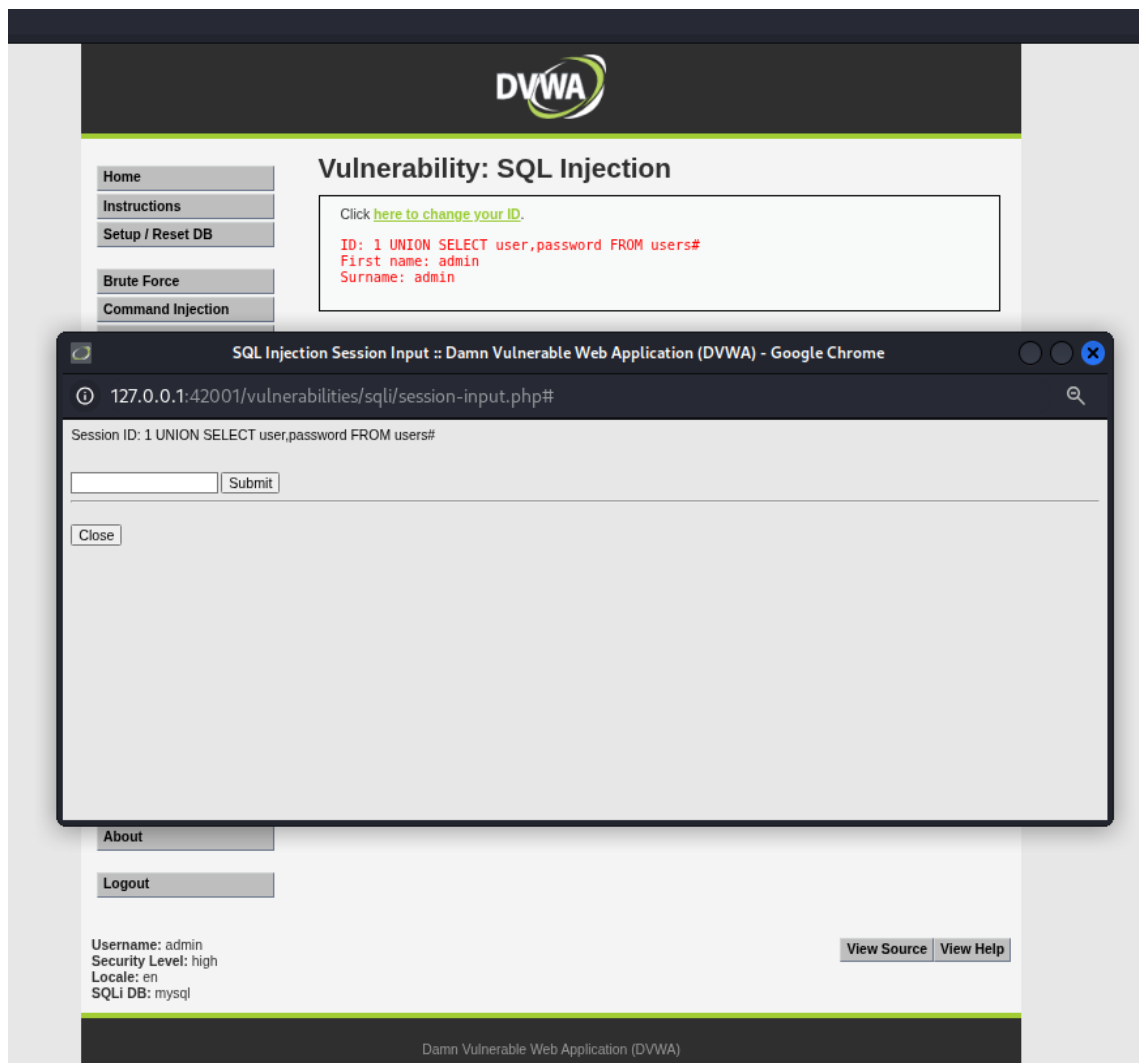
Improved stealth than the low security level but not completely secure from UNIO based payloads.



Screenshot 5: In this data is leaked from the database, modified request through burp suit.

High Security


- High level filtering and SQL attempts are blocked by parameterized queries
- This is the best Industrial standard for preventing SQL injection on enterprise environment
- Session ID Stops unauthorized users from even accessing the page where the vulnerability exists.



- **Screen shoot 6:** After submitting the Session ID and Server sanitized response.

Impossible Security

- All SQL Injection attempts produce no error, no authorization access, or data exposure.
- App uses fully parameterized queries and input valid preventing SQLi.



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

DVWA Security

PHP Info

About

Logout

DVWA Security

Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Impossible ▼

Submit

Security level set to impossible

Username: admin

Security Level: impossible

Locale: en

SQLi DB: mysql

Screenshot 7: Security Level Submitted to Impossible.

Reflected Cross-Site Scripting [XSS]

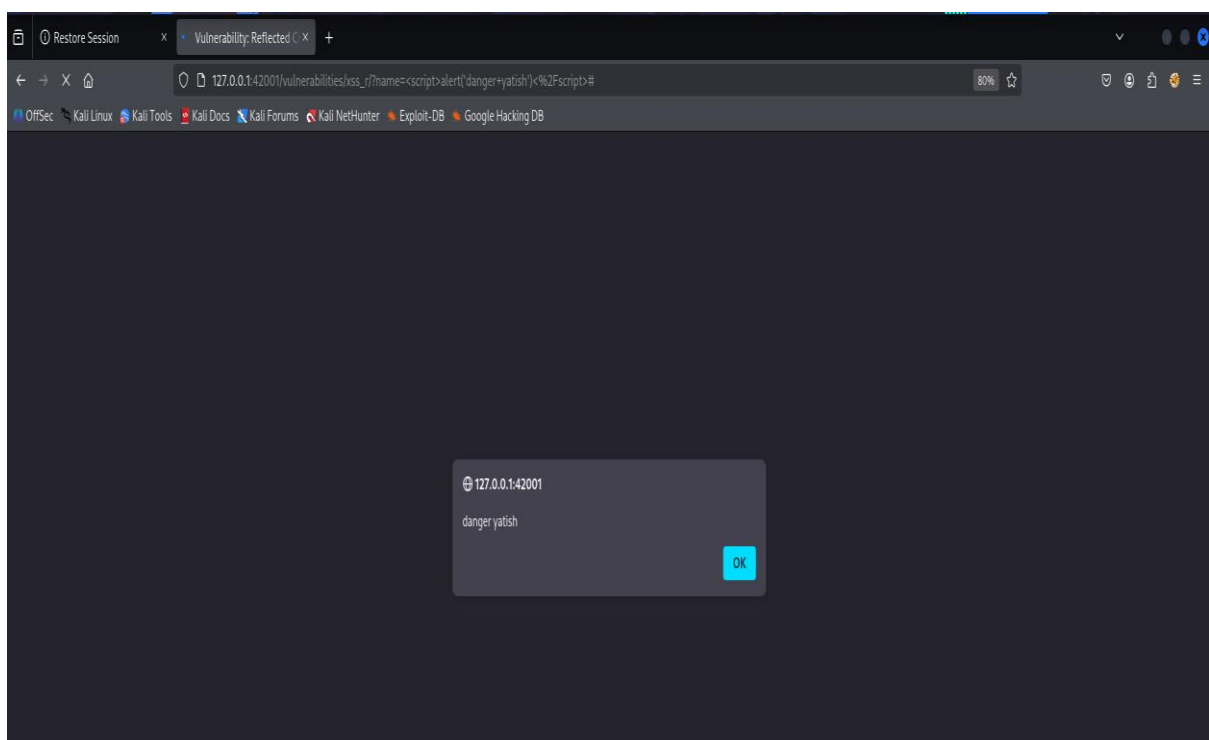
In XSS a malicious JavaScript payload is injected which the payload reflects back through executing scripts in the user's browser.

Low Security

- **Payload:**

<script>alert('XSS') </script>

- **Outcome:** After executing the script and injecting the payload the below result is obtained.

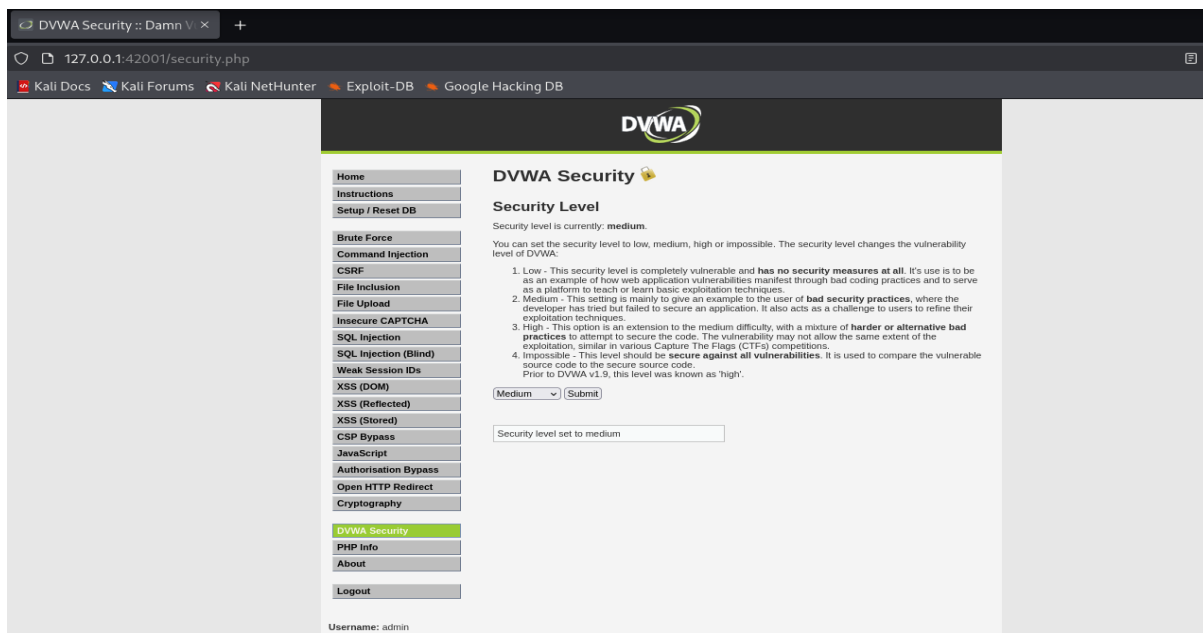


Screenshot 8: This displays the altered message after crossing the actual script.

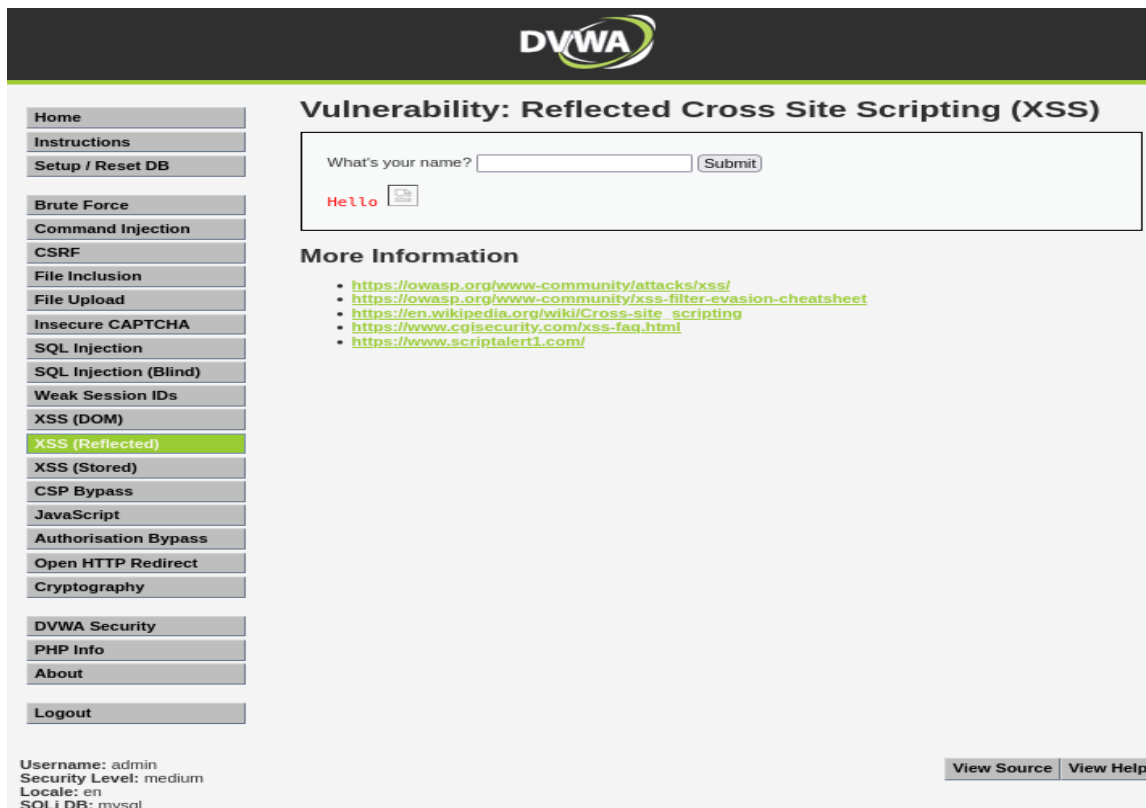
Medium and High Security Level

Payload: < Img src=x onerror=alter('XSS')>

Outcome:



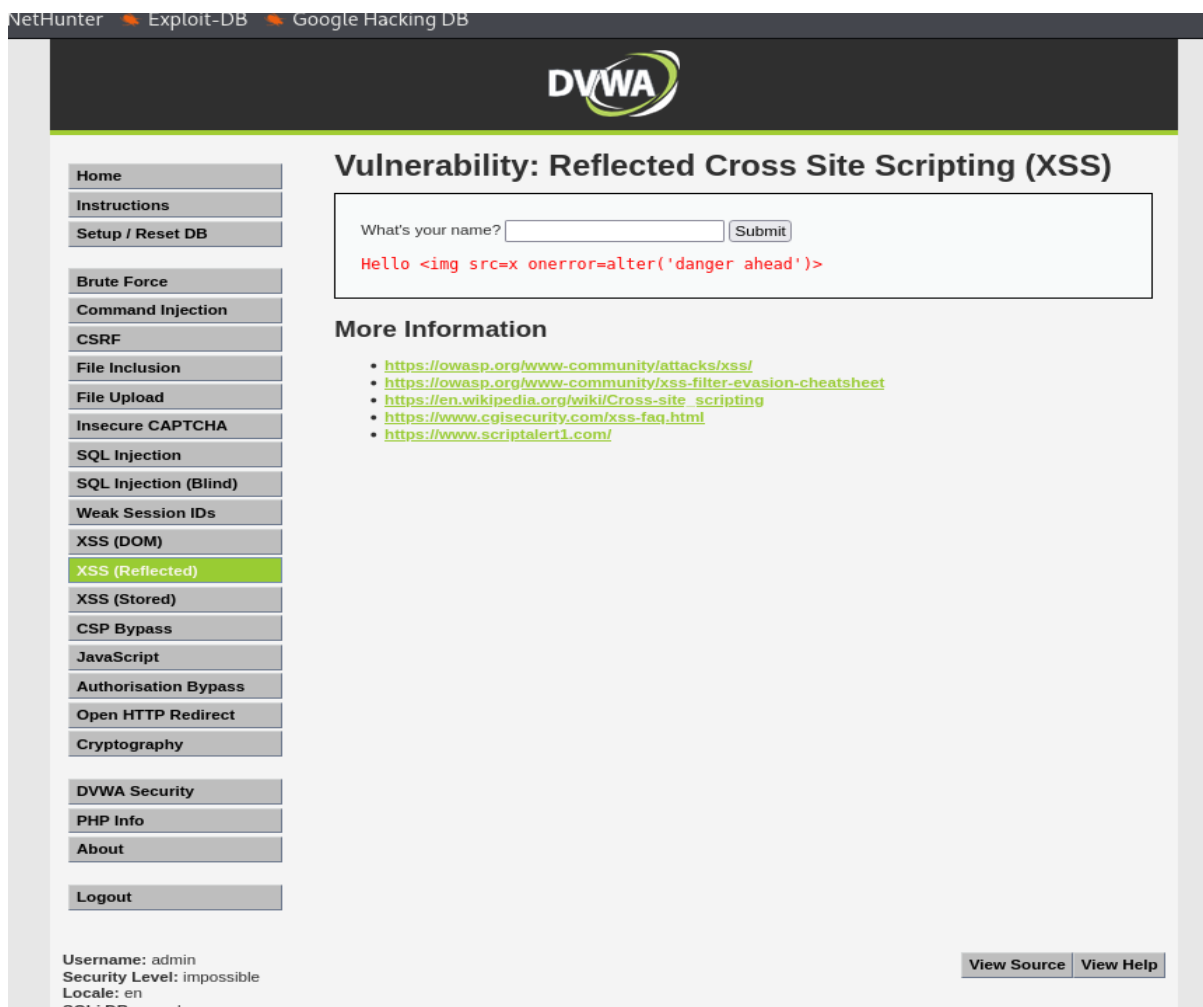
Screenshot 9: The Security level for XSS is changed to medium.



Screenshot 10: After executing the `` code.

Impossible Security Level

In impossible level the message displays the Java Script code itself as the message indicating that no code execution is done.



Screenshot 11: After Injecting the payload output shows no XSS vulnerability.

5.Conclusion

This project underscores the indispensable role of a defense-in-depth security posture in modern web application development. Through the systematic analysis of vulnerabilities across DVWA's escalating security tiers, this study has demonstrated how layered mitigation strategies effectively neutralize evolving attack vectors. The progression from flawed, insecure code to robust, hardened implementations provides a clear, practical illustration of core secure coding principles.

Ultimately, DVWA serves as an invaluable pedagogical tool, offering a controlled, hands-on environment for security practitioners to deconstruct exploits and master defensive techniques. This applied understanding of both attack and defense is fundamental for any professional tasked with safeguarding critical digital assets in today's complex threat landscape.