Project:1

DVWA Security Level Comparison Project

This project focuses on **comparing how the same vulnerabilities behave across different security levels** and understanding **what security controls are added** to prevent attacks.

Objective of the Project

The main objectives of this project are:

- To study how vulnerabilities behave at different DVWA security levels
- To test attack payloads at **Low security level**
- To try the same payloads at **Medium and High levels**

Environment & Tools Used

- **Operating System:** Kali Linux
- **Web Application:** DVWA (Damn Vulnerable Web Application)
- **Browser:** Firefox
- **Security Levels Tested:** Low, Medium, High.

Vulnerabilities Selected for Comparison

For this project, the following **two vulnerabilities** were selected:

1. **SQL Injection**
2. **Cross-Site Scripting (XSS – Reflected)**

Vulnerability 1: SQL Injection

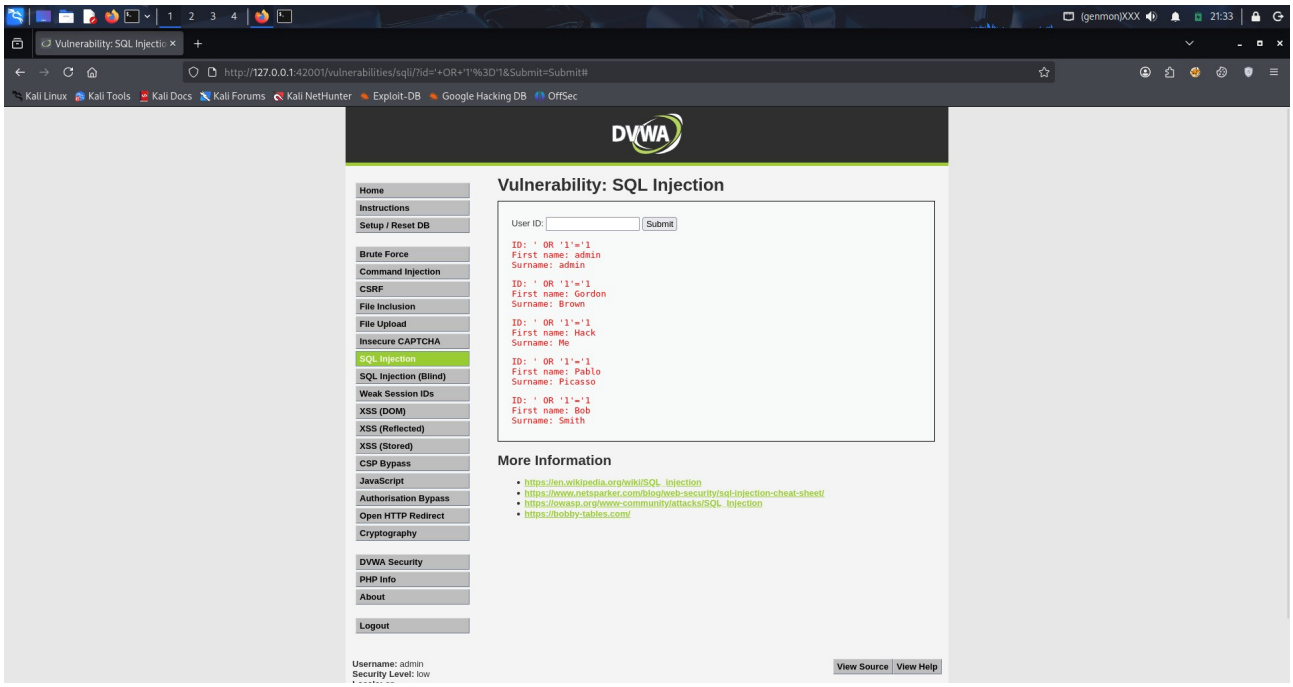
SQL Injection

SQL Injection occurs when user input is directly included in a SQL query without proper validation. This allows attackers to manipulate database queries and gain unauthorized access to data.

SQL Injection at LOW Security Level

Payload Used:

```
' OR '1'='1
```



Observation:

- The payload successfully bypassed authentication.
- The application returned all database records.
- User input was directly concatenated into the SQL query.

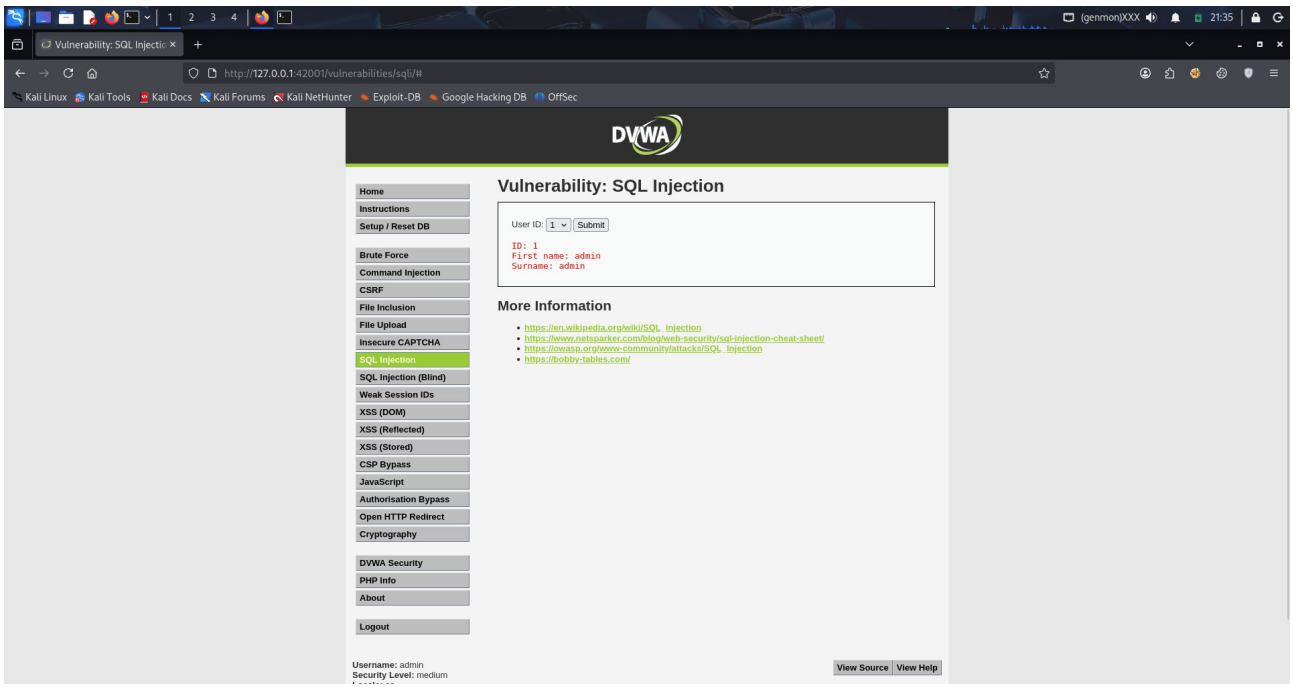
Reason:

- No input validation
- No sanitization
- No prepared statements

SQL Injection at MEDIUM Security Level

Same Payload Used:

```
' OR '1'='1
```



Observation:

- The attack partially failed.
- Some payloads still worked with small modifications.
- Error messages were reduced.

Defense Added:

- Basic input sanitization
- Use of functions like `mysqli_real_escape_string()`
- Removal of some SQL meta-characters

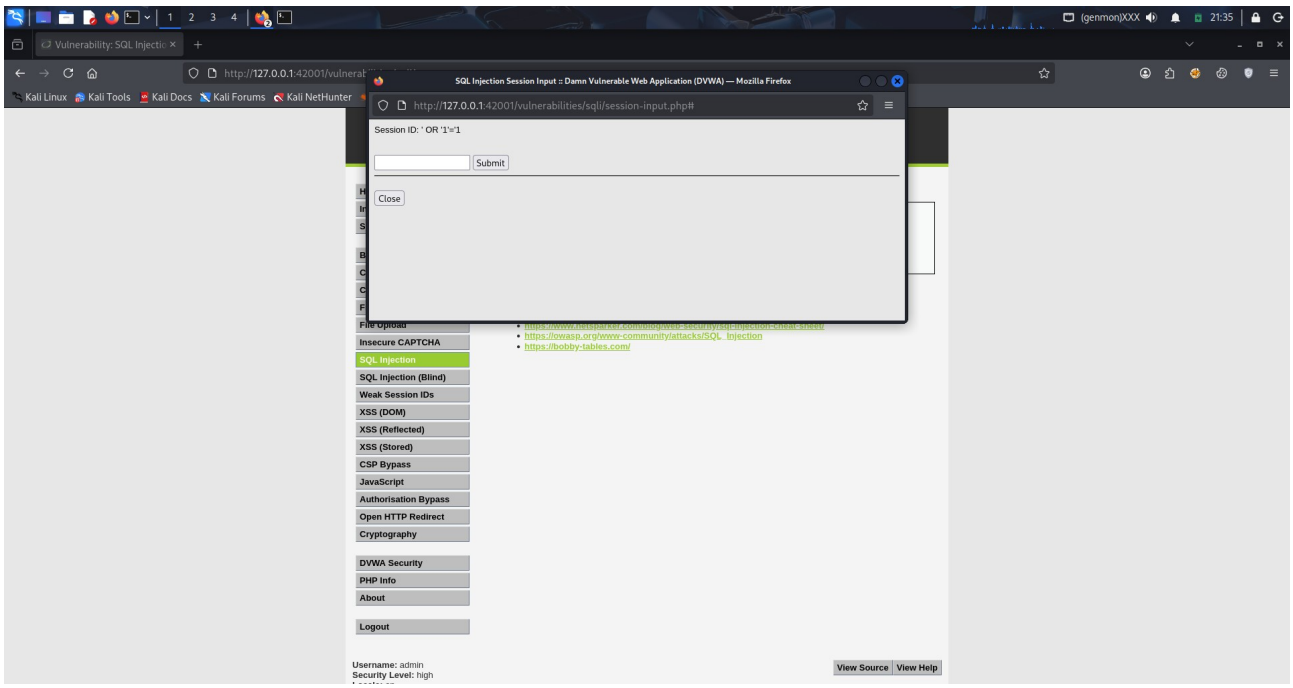
Limitation:

- Still vulnerable to advanced payloads

SQL Injection at HIGH Security Level

Payload Attempted:

```
' OR '1'='1
```



Observation:

- Attack failed completely.
- No database information leaked.
- No error messages shown.

Defense Added:

- Strong input validation
- Query structure controlled
- Use of safer SQL handling methods

Vulnerability 2: Cross-Site Scripting (XSS – Reflected)

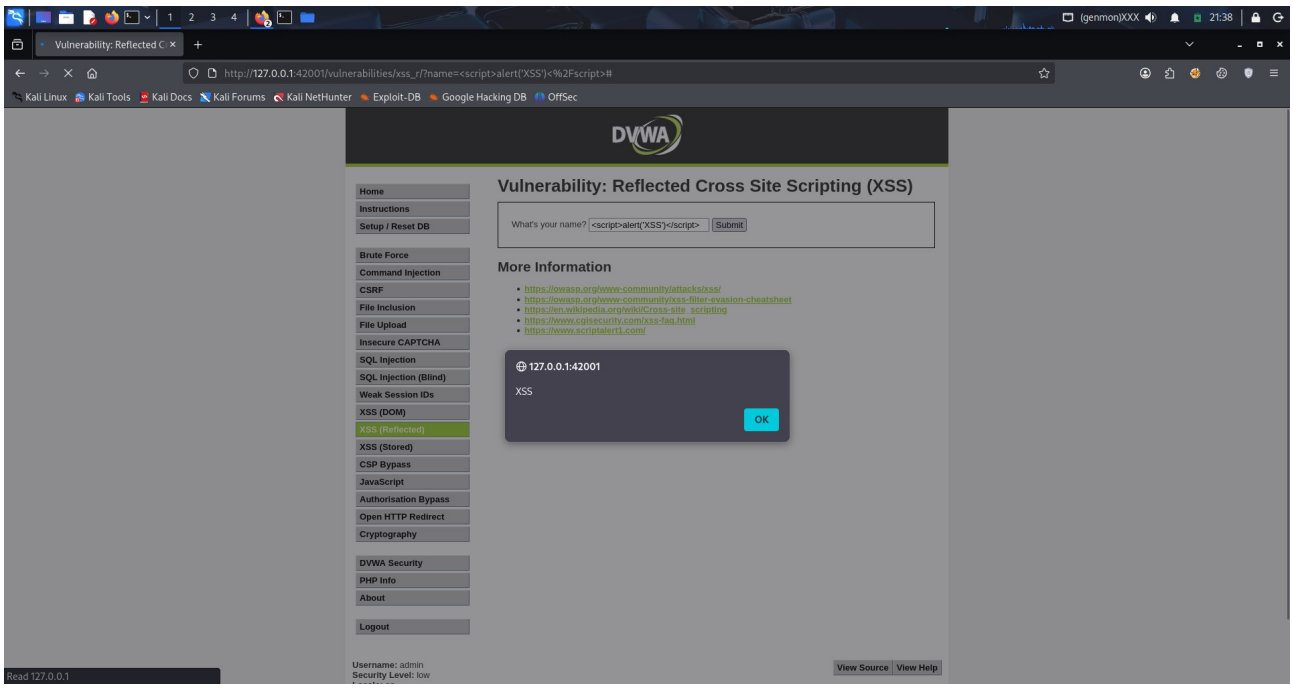
XSS

Cross-Site Scripting (XSS) allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal cookies, session tokens, or perform actions on behalf of the victim.

XSS at LOW Security Level

Payload Used:

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



Observation:

- Script executed successfully.
- Alert popup appeared in browser.

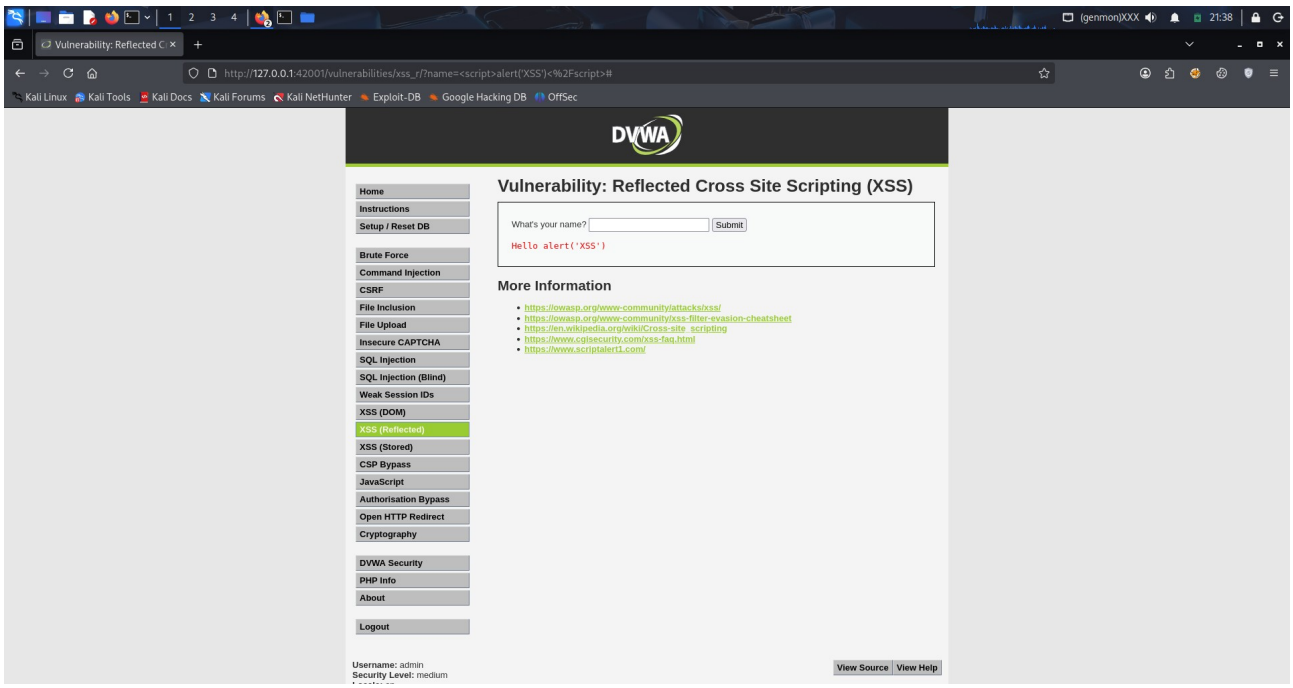
Reason:

- No input filtering
- User input reflected directly to the page

XSS at MEDIUM Security Level

Same Payload Used:

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



Observation:

- Script blocked or partially filtered.
- Some encoded versions still worked.

Defense Added:

- Blacklisting of common tags like `<script>`
- Basic input filtering

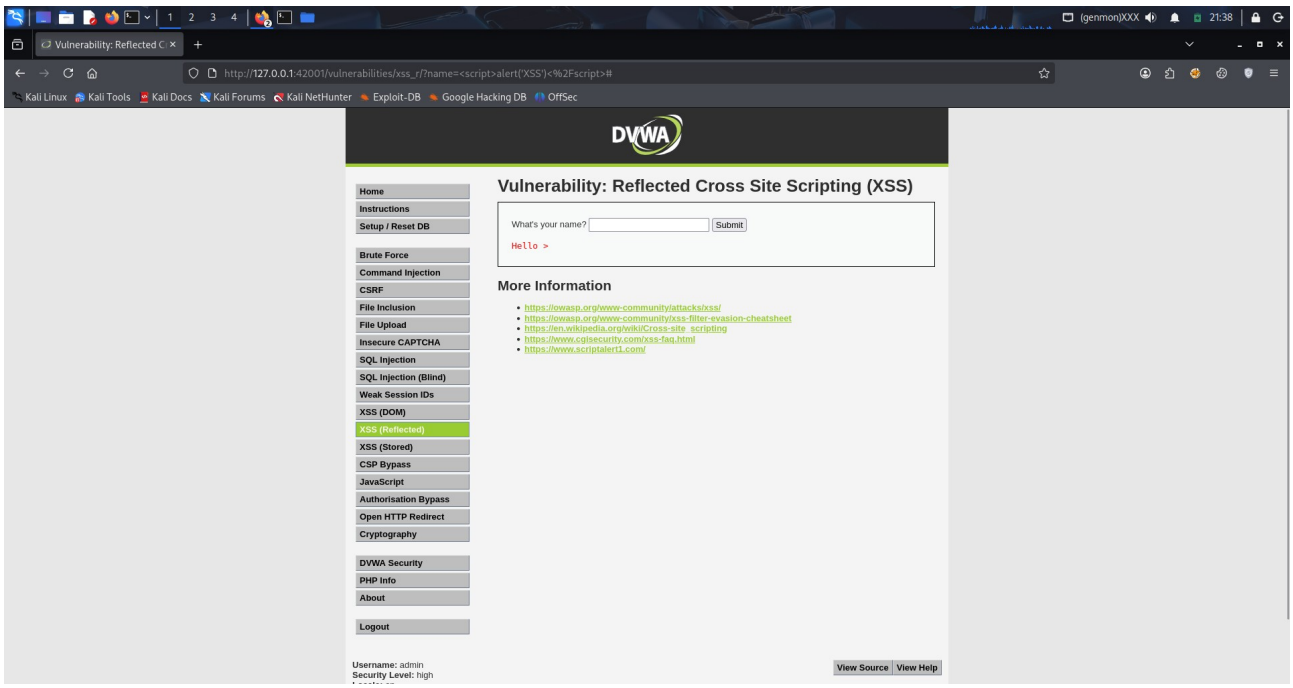
Limitation:

- Encoding bypasses are still possible

XSS at HIGH Security Level

Payload Attempted:

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



Observation:

- Payload did not execute.
- Input displayed as plain text.

Defense Added:

- Output encoding
- Use of `htmlspecialchars()`
- Better input validation

Comparison Summary

Security Level	SQL Injection	XSS	Defense Strength
Low	Fully vulnerable	Fully vulnerable	None
Medium	Partially blocked	Partially blocked	Basic filtering
High	Mostly secure	Mostly secure	Strong validation
Impossible	Fully secure	Fully secure	Best practices

How DVWA Implements Defenses

DVWA gradually improves security by:

- Adding **input sanitization**
- Escaping special characters
- Reducing error disclosure

- Implementing **prepared statements**
- Applying **output encoding**
- Following secure coding standards

This layered approach clearly shows how small security improvements significantly reduce risk.

Conclusion

This project helped in understanding how real-world web vulnerabilities occur and how they can be prevented using secure coding practices. By testing the same payloads across different security levels, it became clear that **security is not about one fix**, but a combination of multiple controls.

DVWA is an excellent platform to learn offensive and defensive security techniques. This comparison strengthened practical knowledge of SQL Injection, XSS, and secure development concepts.