Task 3 : SQL Injection on DVWA (Low Security)

Introduction

DVWA, or Damn Vulnerable Web Application, is a deliberately vulnerable PHP/MySQL web application designed as a safe and legal environment for learning about web application security.
SQL Injection (SQLi) is a code injection technique in cybersecurity that exploits vulnerabilities in an application's software to manipulate or compromise its backend database.
This task demonstrates an SQL Injection attack on DVWA running at low security. The purpose is to understand input validation flaws and authentication bypass techniques.

Objectives

-Set up DVWA on local server
-Set security level to low
-Perform SQL Injection
-Explain vulnerability

Tools used

-DVWA
-Apache2
-Kali Linux
-Browser

Methodology

1. Setting Up DVWA :

DVWA was installed by configuring Apache, PHP, and MySQL, then placing the DVWA folder in /var/www/html/.
Database credentials were configured in config.inc.php.

2. Setting Security Level :

From the DVWA interface:
DVWA Security -> Low
Low level disables input validation and filtering, making SQLi possible.

3. Performing SQL Injection on Login Page :

On the DVWA Login page, the following payload was used:
admin' OR '1'='1
Here,
- **admin'** closes the username string.
-**OR '1'=1** forces SQL query to always return TRUE.
-Database logs in the user without password verification.
This successfully bypassed authentication and logged into the application as admin.

Finding

1. Authentication Bypass Successful

The SQL Injection allowed logging in **without the correct password**, proving the login form is vulnerable.

2. No Input Validation

DVWA (Low Security) blindly inserts user input into the SQL query:
SELECT * FROM users WHERE username='$user' AND password='$pass';
This form is vulnerable because inputs are not sanitized and special characters like ' are not escaped.

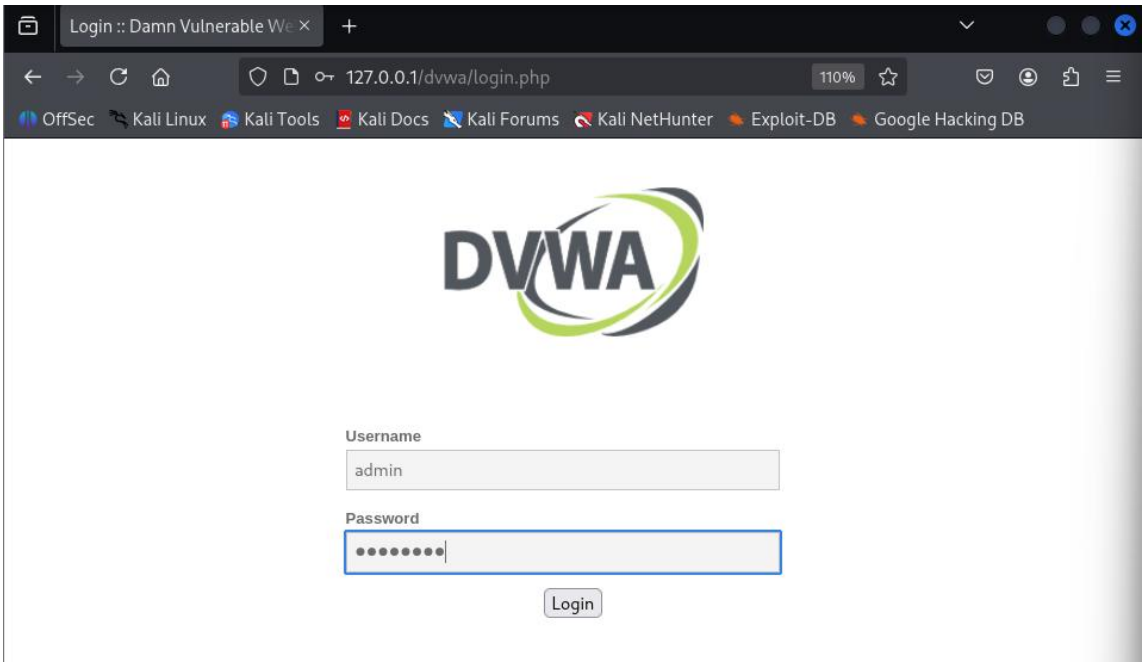
3. Database Error Messages Indicate Vulnerability

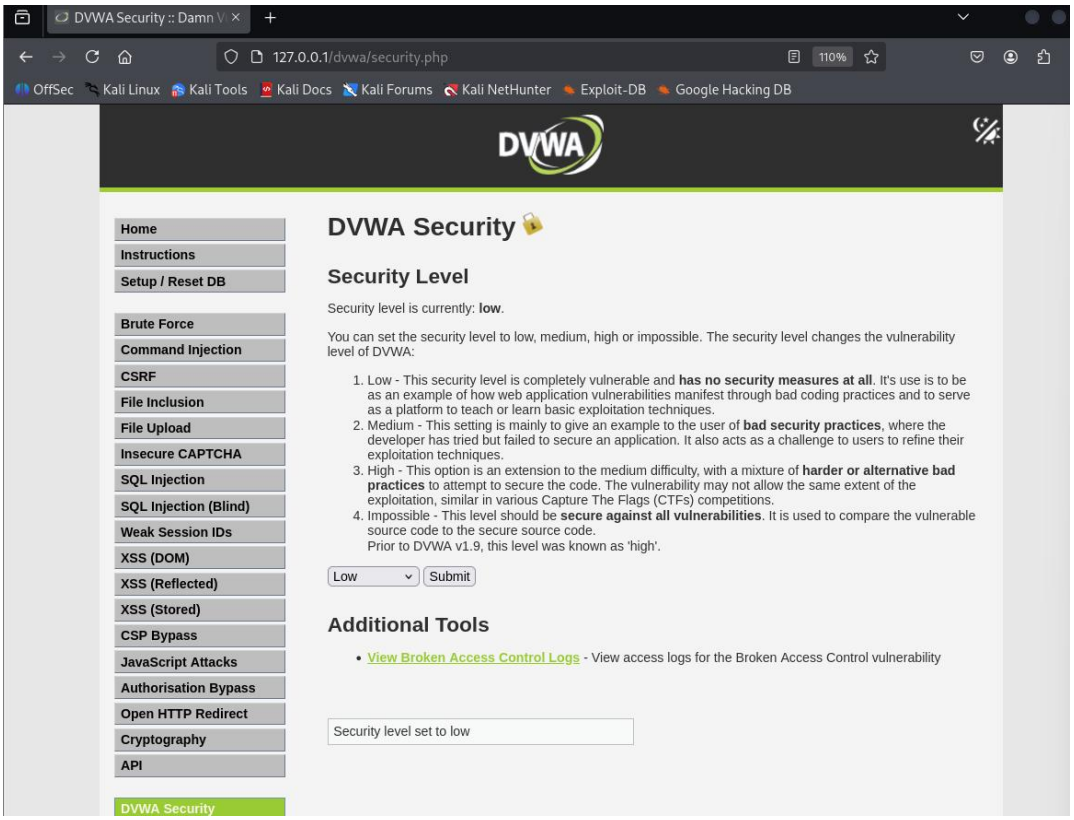
DVWA displays clear database responses, which help attackers craft payloads.

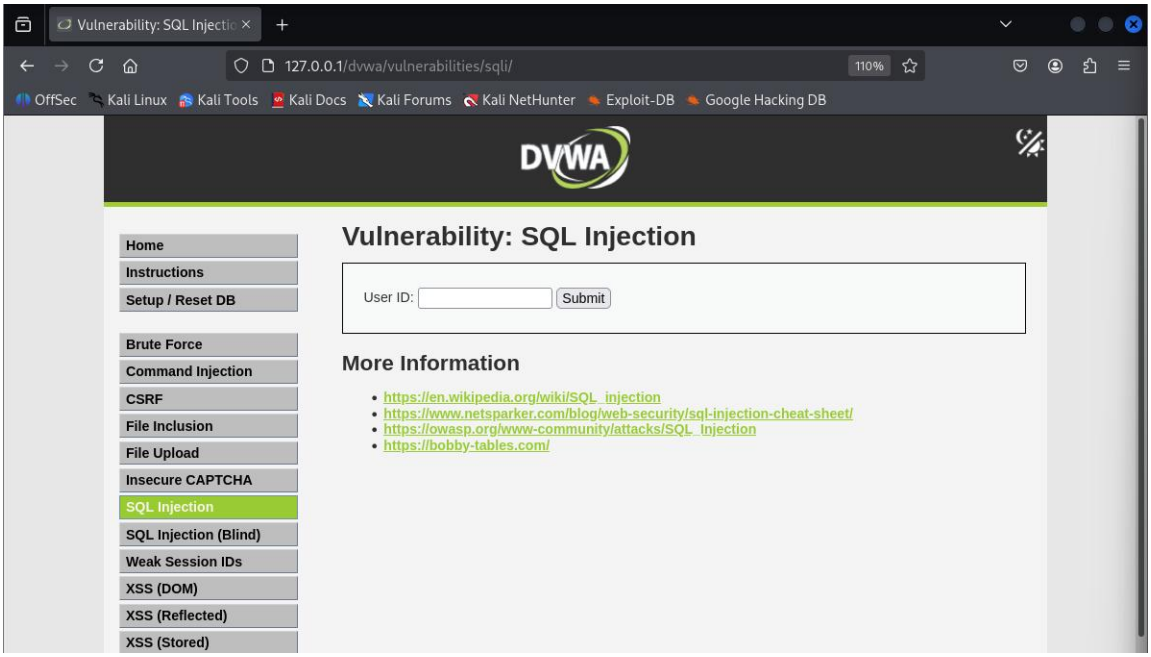
4. Application Logic Can Be Manipulated

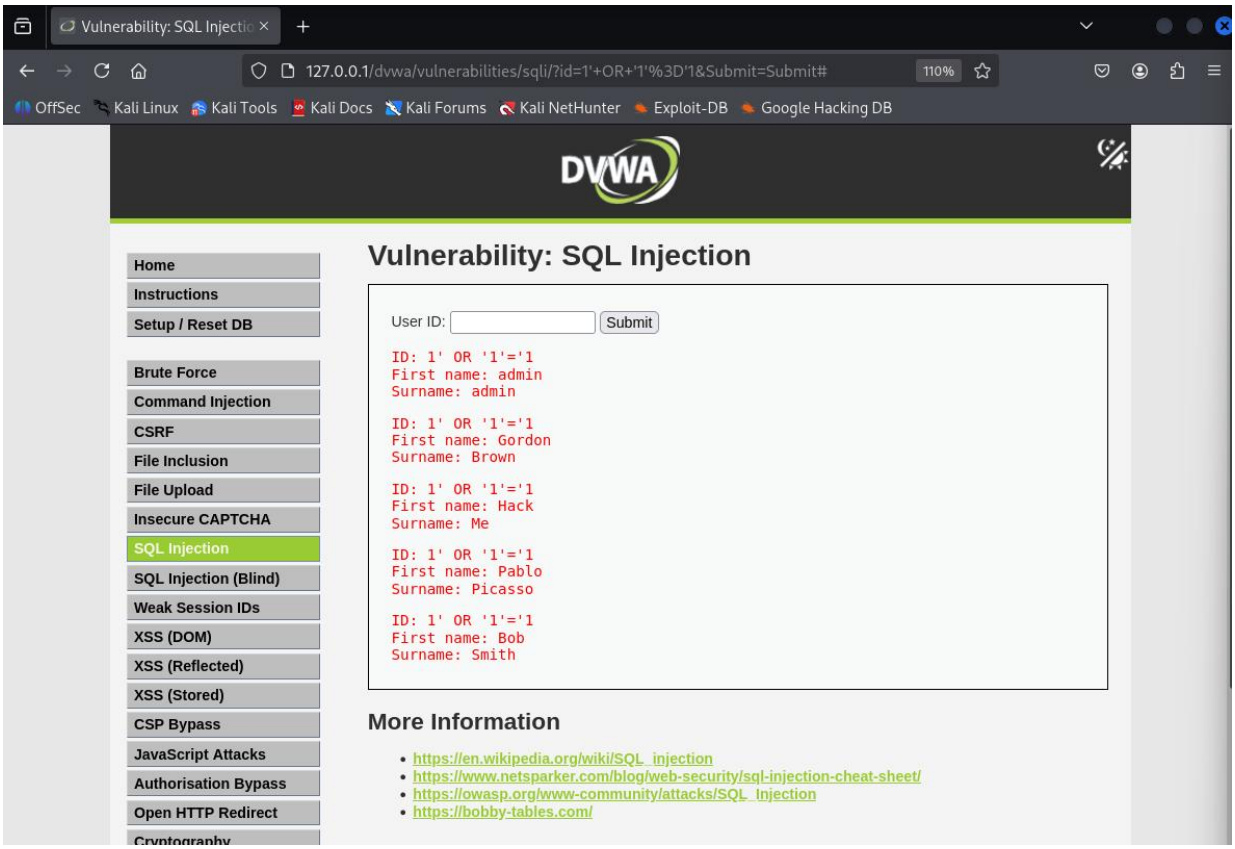
An attacker can:
-Log in as admin
-Extract database data
-Perform UNION-based attacks
-Potentially escalate privileges

Screenshots :






Vulnerability: SQL Injectio
127.0.0.1/dvwa/vulnerabilities/sqli/
110%
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
DVWA
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)
Vulnerability: SQL Injection
User ID:
Submit
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/


Vulnerability: SQL Injectio
127.0.0.1/dvwa/vulnerabilities/sqli/?id=1'+OR+'1'%3D'1&Submit=Submit#
110%
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
DVWA
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 Attacks
Authorisation Bypass
Open HTTP Redirect
Cryptography
Vulnerability: SQL Injection
User ID:
Submit
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
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/

Prevention Measures:

To prevent SQL Injection:
-Use **Prepared Statements** or **Parameterized Queries**.
-Escape user inputs properly.
-Implement server-side validation.
-Restrict error messages sent to clients.
-Use Web Application Firewalls (WAF).

Conclusion:

This task demonstrates how unsafe coding practices can lead to severe vulnerabilities like SQL Injection. Proper validation, secure coding standards, and sanitization are essential to prevent database compromise. DVWA served as a safe environment to practice and understand the impact of SQL Injection.