Vulnerability Assessment and Penetration Testing Report

Project: Kali Linux & DVWA VAPT Demonstration

Prepared By: Vedant Adhikari

Date: May 28, 2025

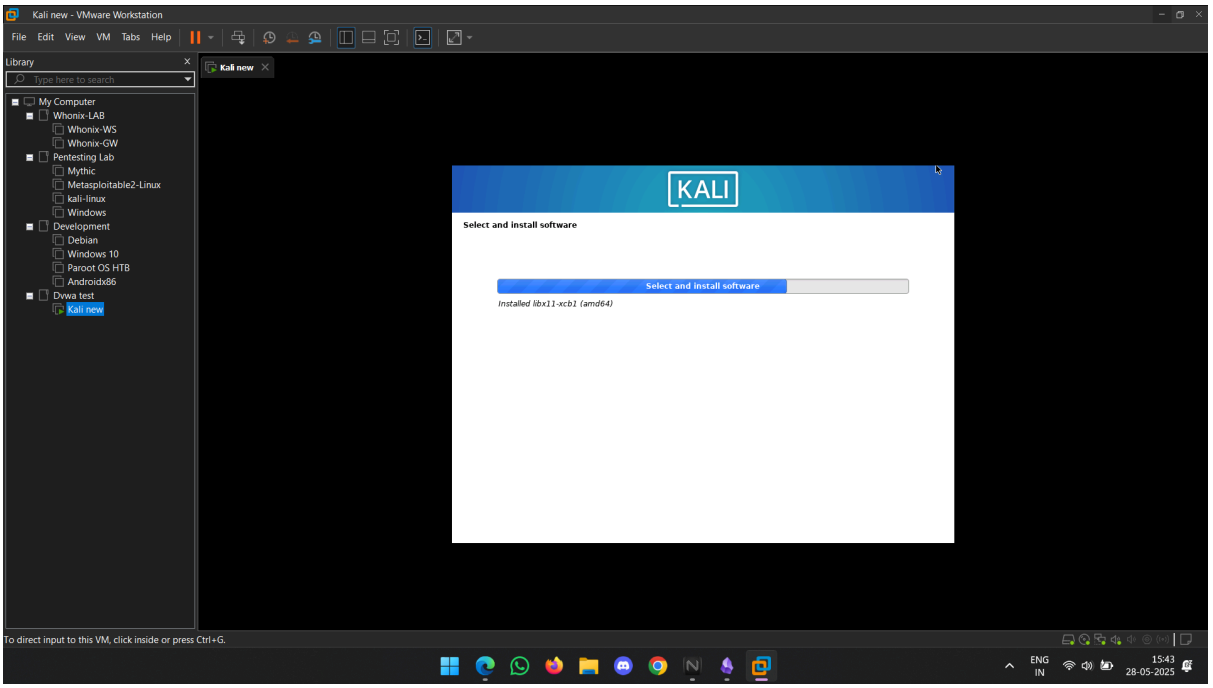
1. Introduction

This document details establishing a penetration testing environment using Kali Linux and configuring the Damn Vulnerable Web Application (DVWA). The report focuses on identifying and exploiting five common web vulnerabilities within this controlled DVWA setting. Each step and finding is documented with clear, time-stamped screenshots, providing a transparent record of the work performed.

2. Kali Linux Environment

This section offers tangible evidence of the Kali Linux environment prepared for this security assessment.

2.1. Kali Linux Installation

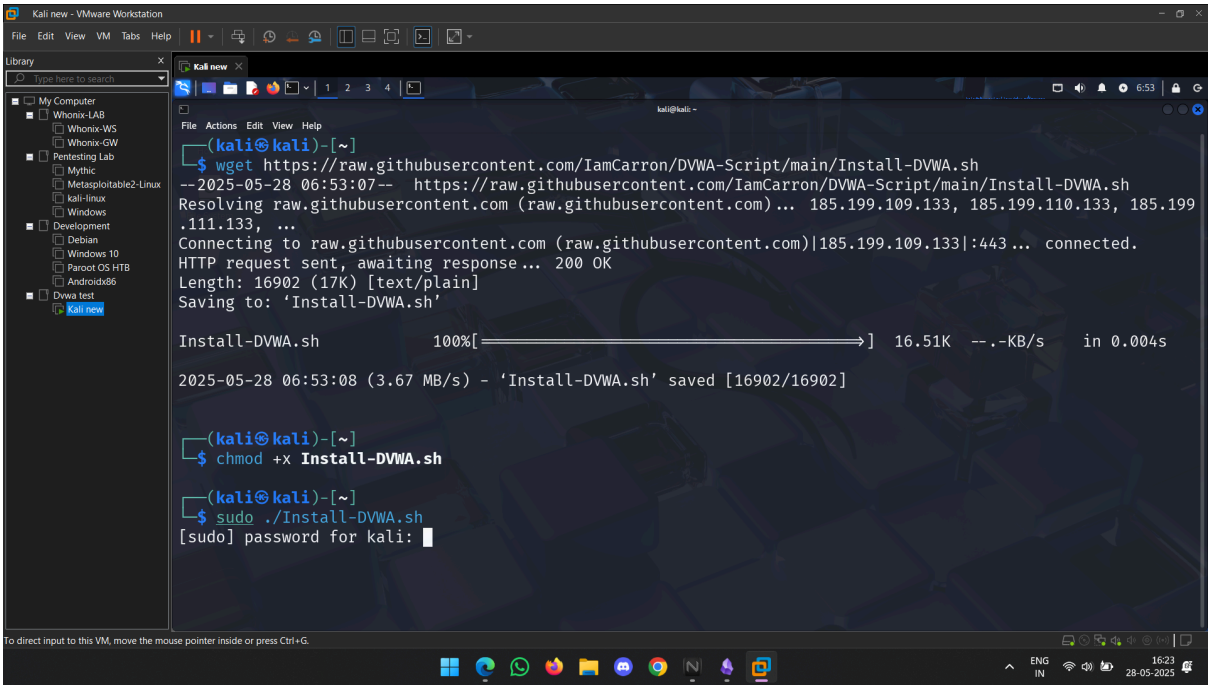


Captured on 28/05/2025 at 15:43.

3. DVWA Setup

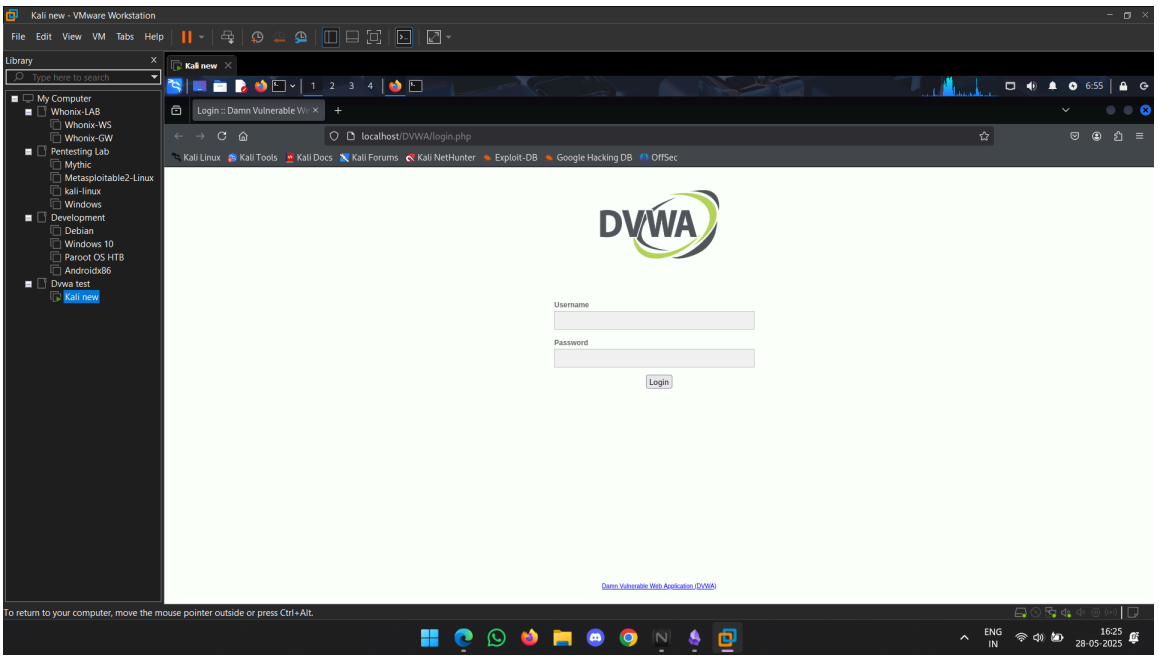
This section outlines the successful deployment and initial configuration of DVWA within the Kali Linux environment.

3.1. DVWA installation



Captured on 28/05/2025 at 16:23.

3.2. DVWA Login Page



Captured on 28/05/2025 at 16:25.

3.3. Setup Process Overview

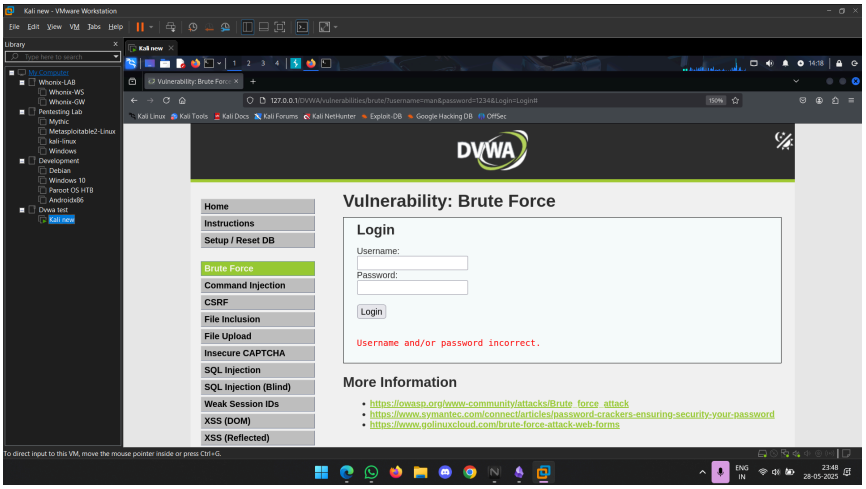
DVWA was installed by cloning its GitHub repository into the Apache web server's root directory. The `config.inc.php` file was adjusted for MySQL database connectivity, and necessary file permissions were set. The database was then initialized via the DVWA web interface.

4. Vulnerability Testing

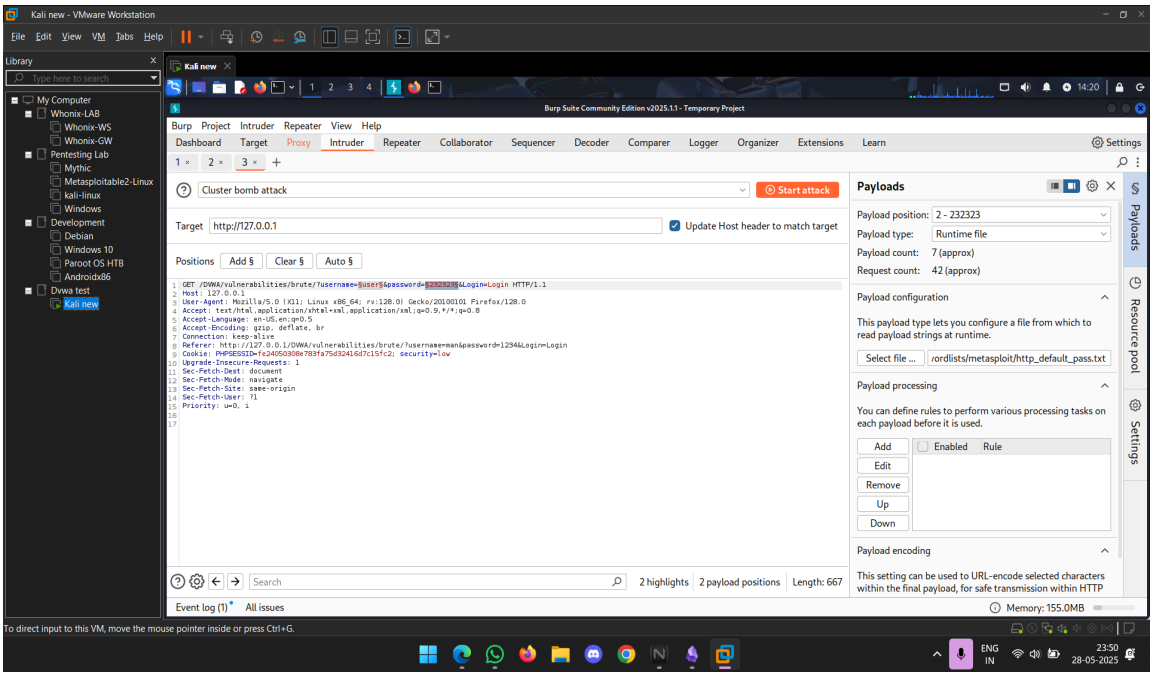
This section presents a detailed account of exploiting five distinct web vulnerabilities within the DVWA environment, set to 'Low' security. For each, a description, exploitation steps, and supporting screenshots are provided.

4.1. Brute Force Vulnerability

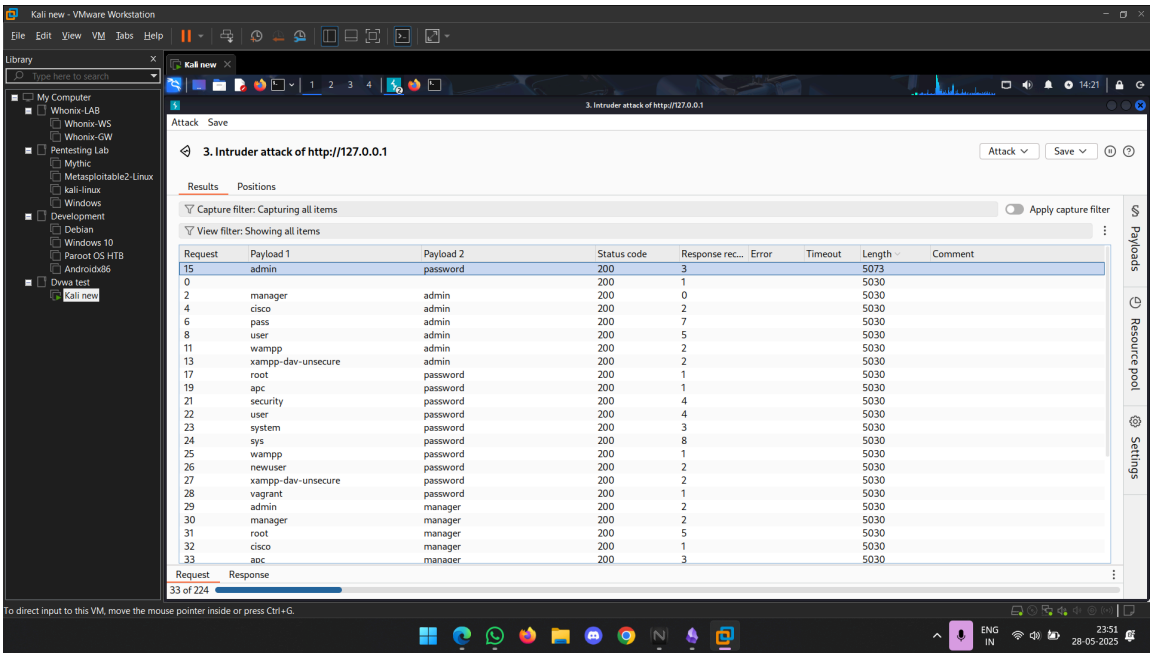
- **Description:** Brute Force in DVWA demonstrates how an attacker systematically guesses login credentials. This often succeeds with weak passwords or common usernames.
- **Steps to Exploit:**
    - Navigate to the "Brute Force" section in DVWA.
    - Open Burp Suite and configure your browser to use its proxy.
    - Enter `user` for username and `232323` for password on the DVWA login page and click "Login".
    - In Burp Suite's "Proxy" tab, find the captured login request and send it to "Intruder".
    - In "Intruder," go to the "Positions" tab. Clear existing payload markers. Select and add markers (§) around the `user` and `232323` values.
    - Go to the "Payloads" tab. For Payload Set 1 (username), load a simple username list (e.g., `http_default_users.txt`). For Payload Set 2 (password), load a simple password list (e.g., `http_default_pass.txt`).
    - Start the attack and observe the responses. Look for a response length or status code that indicates a successful login (e.g., a redirect to the main page).
- **Screenshots:**



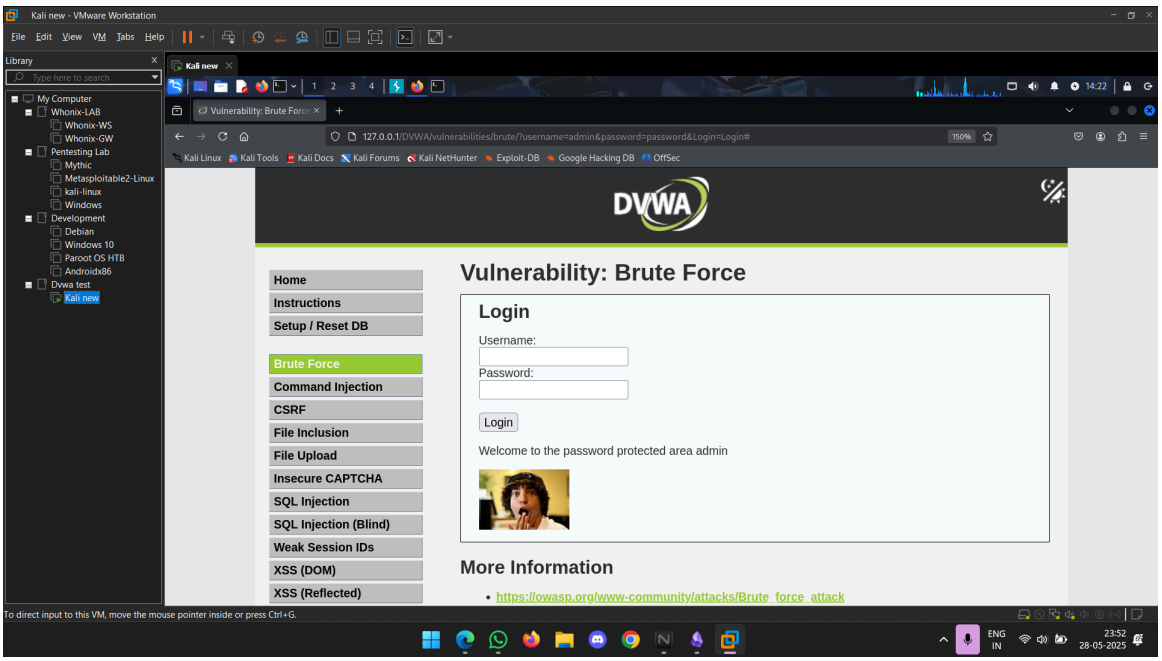
Captured on 28/05/2025 at 23:48.



Captured on 28/05/2025 at 23:50.



Captured on 28/05/2025 at 23:51.



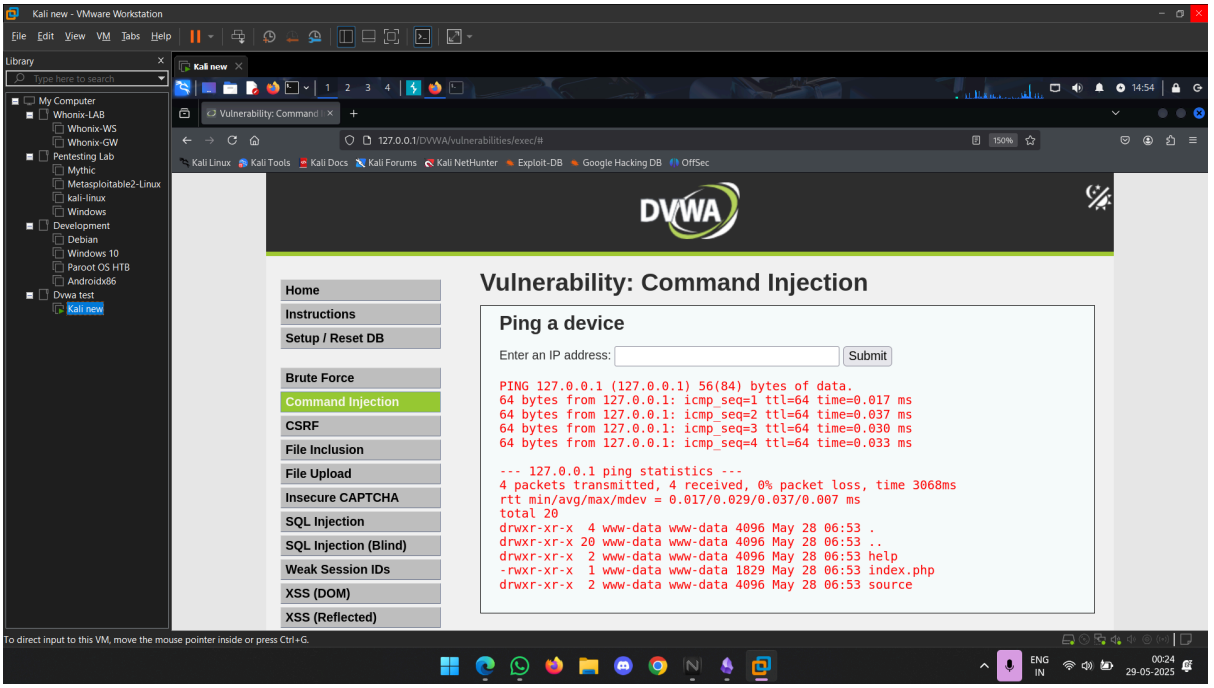
Captured on 28/05/2025 at 23:52.

- **Impact:** Unauthorized account access, leading to data compromise, privilege escalation, or further system compromise.
- **Mitigation:** Implement strong password policies, account lockout mechanisms, CAPTCHA, and multi-factor authentication (MFA).

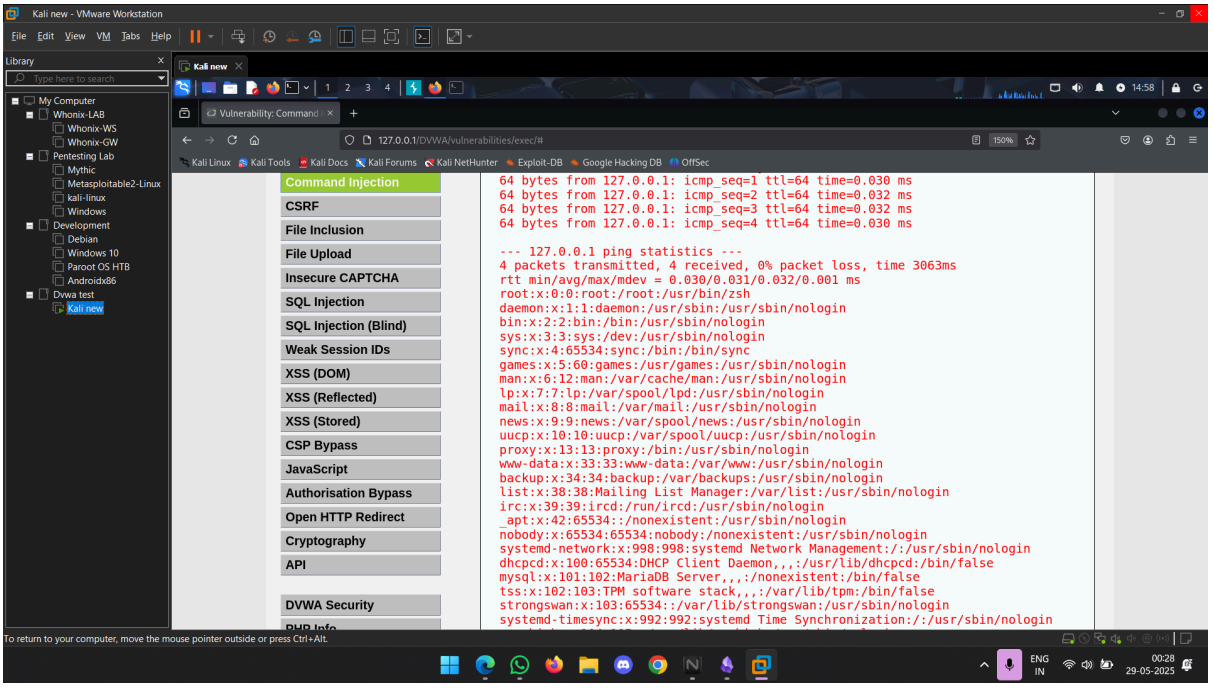
4.2. Command Injection Vulnerability

- **Description:** Command Injection allows an attacker to execute arbitrary system commands on the host OS. This occurs when unsanitized user input is directly incorporated into a system command.
- **Steps to Exploit:**
    1. Access the "Command Injection" section in DVWA.
    2. In the input field, enter `127.0.0.1`. Click "Submit" to see normal behavior.
    3. Now, enter `127.0.0.1 && ls -la /` and click "Submit". Observe the output.
    4. Try another command: `127.0.0.1 ; cat /etc/passwd` and click "Submit". Observe the output.

- **Screenshots:**



Captured on 29/05/2025 at 00:24.

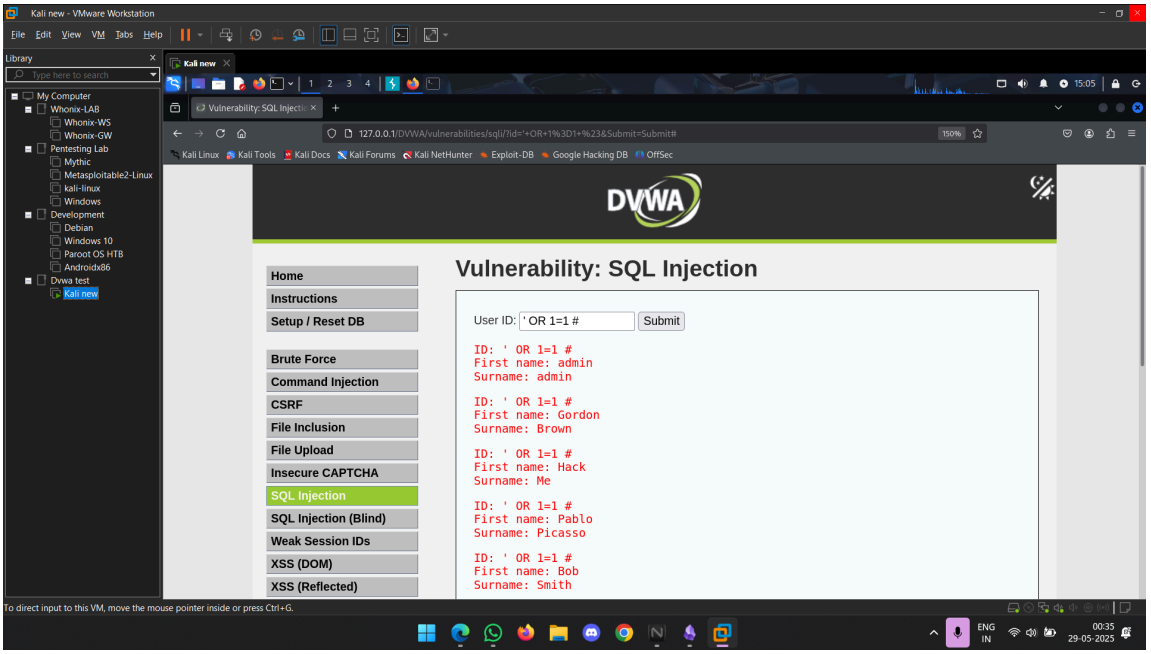


Captured on 29/05/2025 at 00:28.

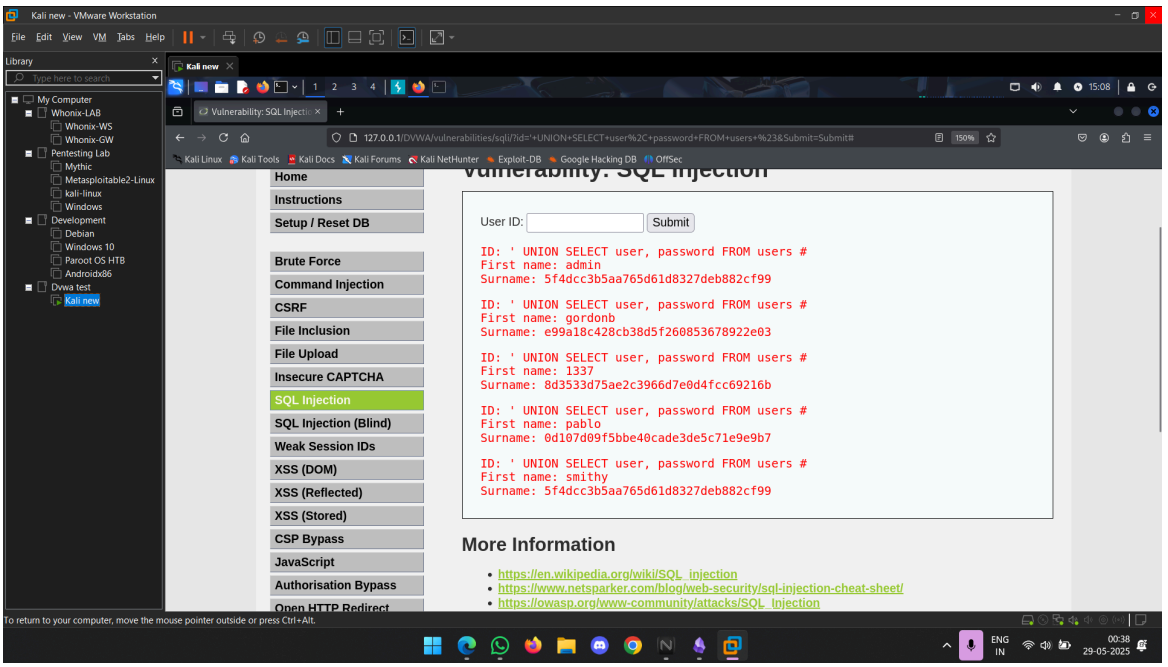
- **Impact:** Complete compromise of the underlying server, including data exfiltration, arbitrary code execution, and full system control.
- **Mitigation:** Avoid direct execution of system commands with user input. Use safer APIs, whitelist allowed commands, and rigorously validate/sanitize all user input.

4.3. SQL Injection Vulnerability

- **Description:** SQL Injection is a code injection technique targeting data-driven applications. Malicious SQL statements are inserted into input fields and executed by the database, leading to unauthorized data access or manipulation.
- **Steps to Exploit:**
    - Navigate to the "SQL Injection" section in DVWA.
    - In the User ID field, enter `' OR 1=1 #` and click "Submit". Observe the result (it should show all users).
    - Now, try to extract specific data. Enter `' UNION SELECT user, password FROM users #` and click "Submit". Observe the result (it should show usernames and passwords).
- **Screenshots:**



Captured on 29/05/2025 at 00:35.



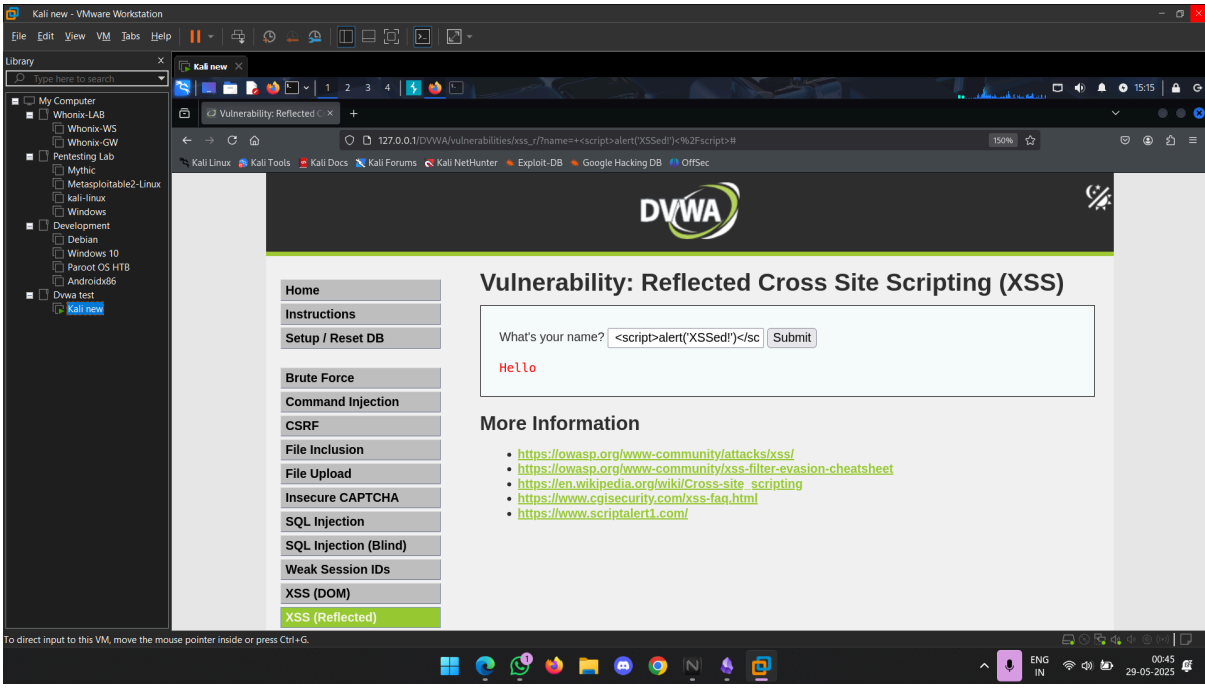
Captured on 29/05/2025 at 00:38.

- **Impact:** Unauthorized access to sensitive data, data modification/deletion, administrative database access, or even remote code execution.
- **Mitigation:** Use prepared statements with parameterized queries, employ ORMs, validate/sanitize all user input, and implement least privilege for database accounts.

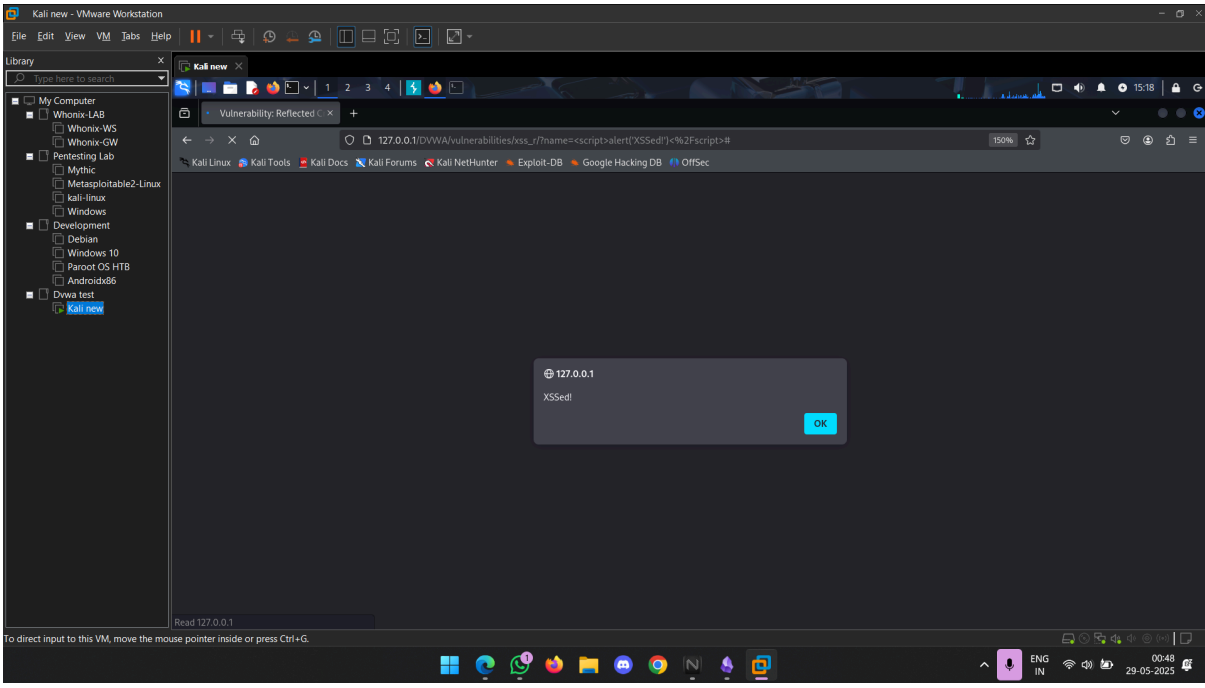
4.4. XSS (Reflected) Vulnerability

- **Description:** Reflected Cross-Site Scripting (XSS) is a client-side code injection vulnerability where a malicious script, injected by an attacker, is immediately "reflected" off a web server and executed by the user's browser.
- **Steps to Exploit:**
    - Navigate to the "XSS (Reflected)" section in DVWA.
    - In the input field, enter `<script>alert('XSSed!')</script>` and click "Submit".
    - Observe the pop-up alert box.

- **Screenshots:**



Captured on 29/05/2025 at 00:45.

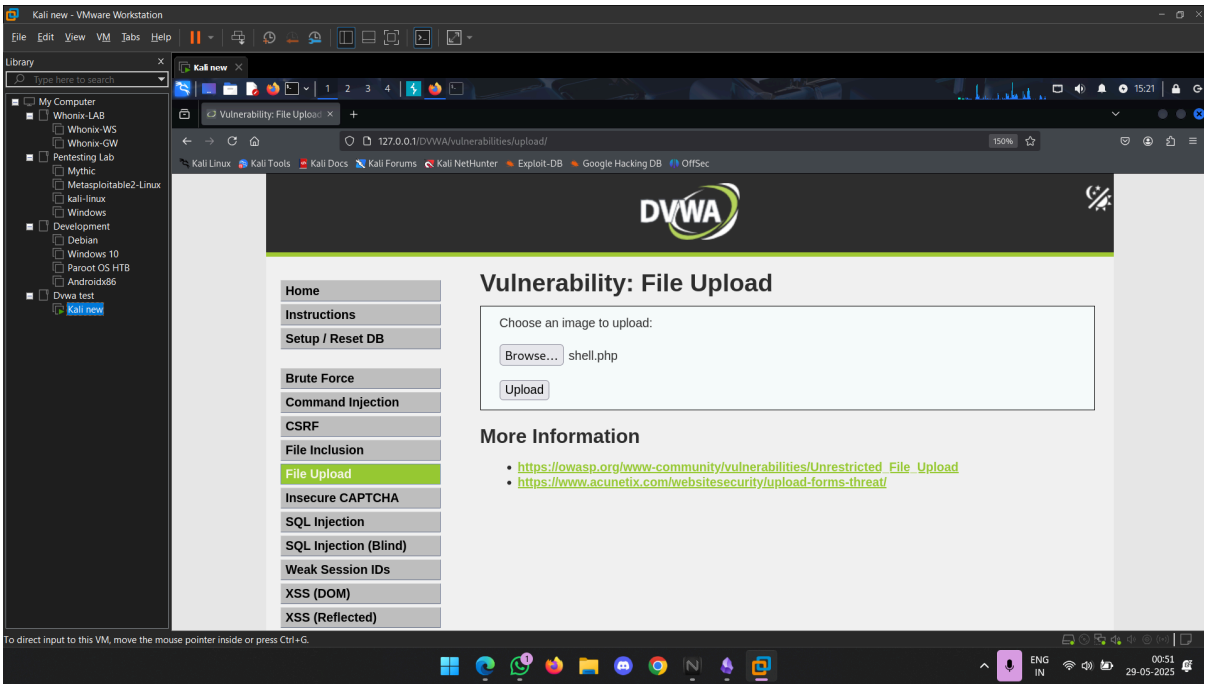


Captured on 29/05/2025 at 00:48.

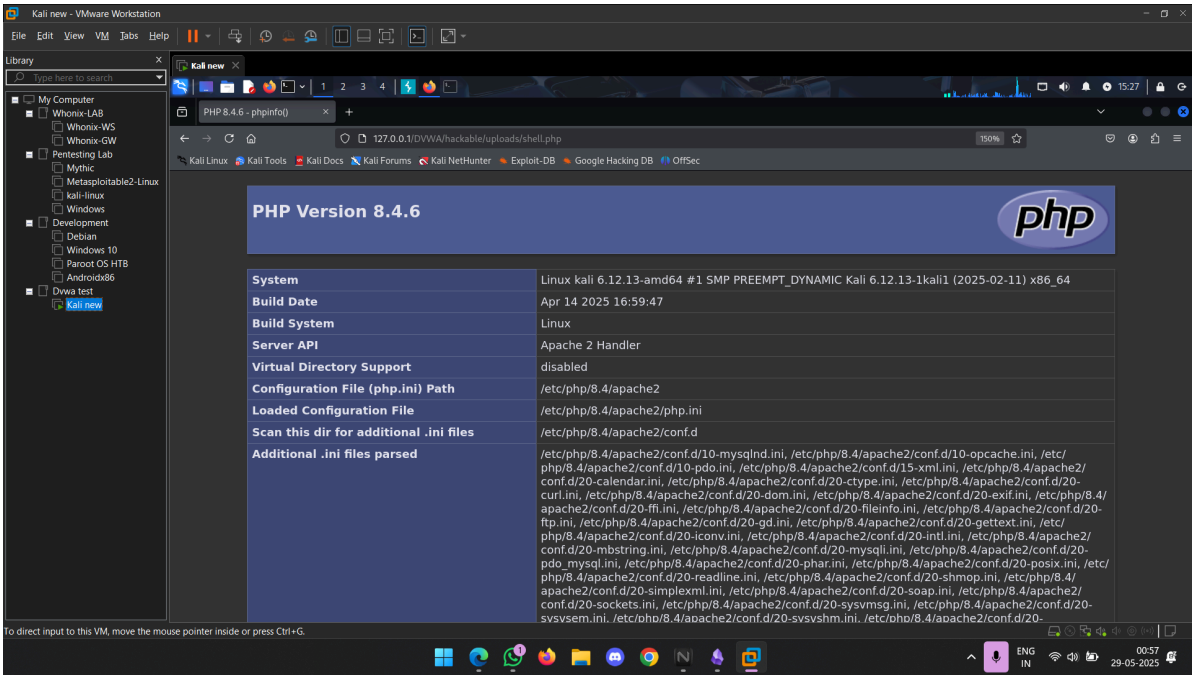
- **Impact:** Session hijacking, page defacement, redirection to malicious sites, or execution of arbitrary client-side code, leading to data theft or malware delivery.
- **Mitigation:** Implement strict input validation and robust output encoding for all user data before rendering it in HTML. Adopt a strong Content Security Policy (CSP).

4.5. File Upload Vulnerability

- **Description:** File Upload vulnerabilities allow an attacker to upload arbitrary, potentially malicious, files (e.g., web shells) to a server. If executable, this can lead to remote code execution.
- **Steps to Exploit:**
    - Navigate to the "File Upload" section in DVWA.
    - Create a simple PHP file on your Kali machine, e.g., `shell.php`, with content `<?php phpinfo(); ?>`.
    - Click "Browse..." on the DVWA page, select your `shell.php` file, and click "Upload".
    - If the upload is successful, you will see a path like `/hackable/uploads/shell.php`. Navigate to this path in your browser (e.g., `http://localhost/DVWA/hackable/uploads/shell.php`).
    - Observe the `phpinfo()` page, confirming the shell execution.
- **Screenshots:**



Captured on 29/05/2025 at 00:51.



Captured on 29/05/2025 at 00:57.

- **Impact:** Full remote code execution on the server, leading to system compromise, data manipulation, or denial of service.
- **Mitigation:** Implement strict file type validation (whitelist), rename uploaded files, store files outside the web root, scan for malware, and enforce least privilege on upload directories.

5. Conclusion

This report documents the setup of a VAPT environment using Kali Linux and DVWA, and the successful exploitation of five common web vulnerabilities. This exercise reinforced my understanding of these security flaws and highlighted the importance of implementing robust security measures to protect web applications. Continued practice and adherence to secure coding principles are essential for building resilient systems.