**Web Application Penetration Testing**

**Project Title:** Vulnerability Assessment of DVWA  
**Prepared by:** Sai Sangeeth v  
**Testing Target:** DVWA (Damn Vulnerable Web Application)  
**Tools Used:** Burp Suite, Browser Dev Tools  
**Methodology:** OWASP Top 10 Framework

**Table of Contents**

1. Executive Summary
2. Scope of Work
3. Methodology
4. Activities Performed
5. Proof of Concept
6. Conclusion

**Executive Summary**

This report details the penetration testing activities conducted on the **DVWA (Damn Vulnerable Web Application)**. The assessment focused on identifying vulnerabilities based on the OWASP Top 10. Several critical issues, including **SQL Injection**, **Broken Authentication**, and **Reflected XSS**, were successfully discovered and exploited. Each finding is documented with evidence and actionable remediation steps.

**Scope of Work**

* **Target Application:** DVWA (Local Environment)
* **Objective:** Identify and exploit OWASP Top 10 vulnerabilities
* **Testing Type:** Manual + Tool-Assisted

**Methodology**

| **Phase** | **Description** |
| --- | --- |
| Reconnaissance | Mapping pages using Burp Suite spider |
| Vulnerability Discovery | Manual and automated scans via Burp Suite |
| Exploitation | Executing controlled attacks to confirm vulnerabilities |
| Documentation | screenshots, and writing detailed reports |

**Activities Performed**

* Studied OWASP Top 10 vulnerabilities.
* Set up DVWA in a controlled environment.
* Configured Burp Suite for scanning and intercepting HTTP requests.
* Conducted manual testing on:
  + SQL Injection (Authentication Bypass)
  + Broken Authentication (Session Reuse)
  + Reflected Cross-Site Scripting (XSS)
  + Sensitive Data Exposure
  + Local File Inclusion
  + Remote Code Execution

**Proof of Concept**

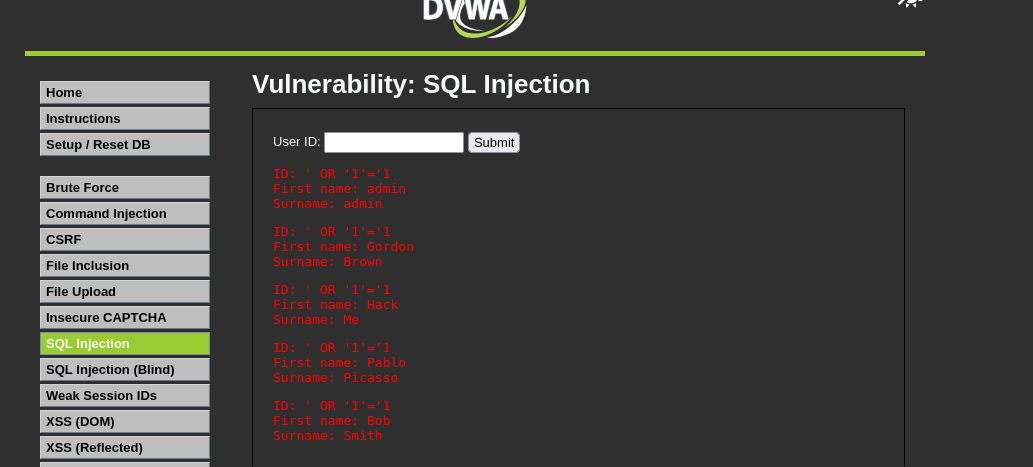
**1. Title: SQL Injection via User ID Field**

Risk Level: High

CVSS Score: 9.0  
Affected Page: vulnerabilities/sqli/  
Payload Used: ' OR '1'='1

**Description:** The input field for User ID is vulnerable to SQL Injection. The payload

' OR '1'='1 tricks the backend SQL query into always evaluating TRUE, returning all user records instead of one specific result.



**Impact:**

* Unauthorized access to sensitive data (usernames, names).
* Possible data exfiltration or database compromise.
* Opens potential for authentication bypass or full DB control.

**Recommendation:**

* Use parameterized queries (e.g., PDO, mysqli\_prepare() in PHP).
* Apply strict input validation and escaping.
* Use ORM frameworks or security libraries that handle SQL safely.
* Avoid constructing SQL queries with raw user input.

**2. Title: Broken Authentication due to Missing Brute-Force Protection**Risk Level: High

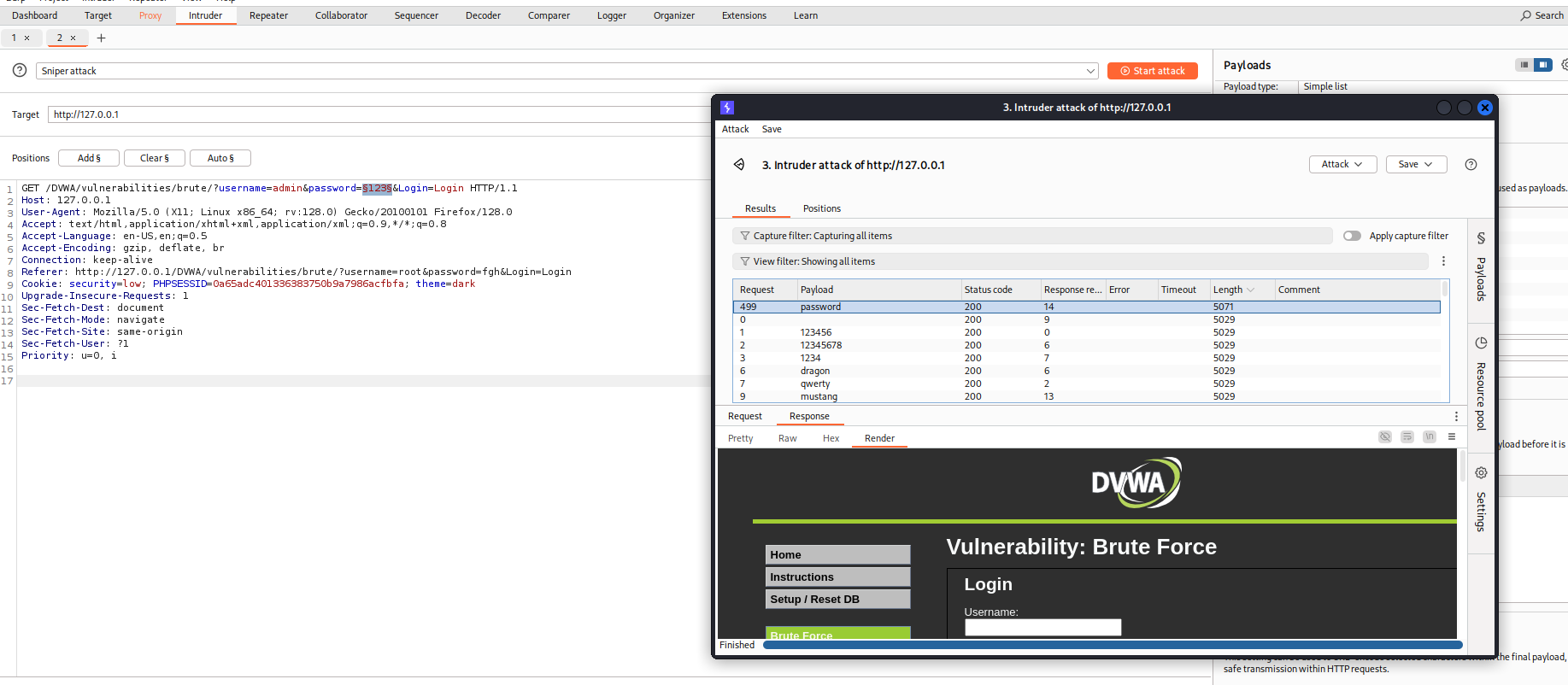
CVSS Score: 8.0

Affected Module: DVWA – Brute Force (/vulnerabilities/brute/)  
Test Tool: Burp Suite (Intruder)

**Description :** The application lacks adequate protections against brute-force attacks. By sending a series of automated password attempts, an attacker was able to guess a valid admin password (password) and successfully authenticate.

This confirms the login mechanism is vulnerable to **broken authentication**, violating best practices such as:

* No rate-limiting
* No account lockout
* No CAPTCHA or secondary challenge



**Impact:**

* Unauthorized access to sensitive user accounts.
* Privilege escalation (e.g., admin access).
* Risk of full application compromise if further chained

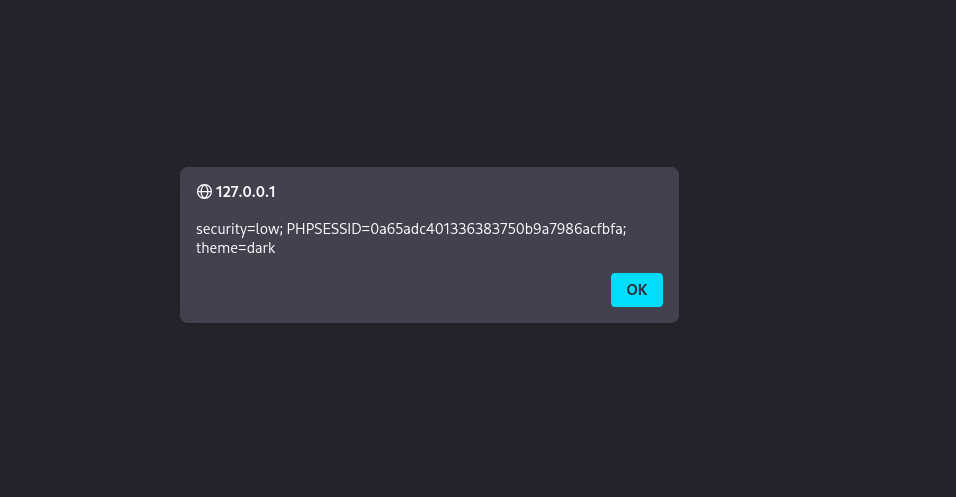
**Recommendations:**

* Implement rate-limiting and delay mechanisms after failed login attempts.
* Lock account after multiple failed attempts.
* Use CAPTCHA to distinguish bots from humans.
* Enforce multi-factor authentication (MFA) for high-privilege accounts.

**3. Title: Reflected XSS Enabling Cookie Access**Risk Level: High

CVSS Score: 8.0  
Payload Used: <script>alert(document.cookie)</script>

**Description:** The application improperly reflects unsanitized input back into the response. This allowed arbitrary JavaScript execution, including accessing cookies via document.cookie. This poses a serious **session hijacking** risk if session IDs or auth tokens are exposed.



**Impact:**

* Exposes session identifiers to attackers.
* Could lead to session hijacking, impersonation, or privilege escalation.
* Enables chaining with other vulnerabilities like CSRF.

**Recommendation:**

* Set the HTTPOnly flag on sensitive cookies (e.g., session ID) to prevent JavaScript access.
* Sanitize and encode user input before reflecting it in HTML.
* Apply Content Security Policy (CSP) headers to restrict script execution.
* Upgrade to modern web frameworks with secure templating.

**4.Title : Sensitive Data Exposure (Weak Hashing - MD5)**

Risk Level: High

CVSS Score: 7.0

**Description :** DVWA stores user passwords using the MD5 hashing algorithm, which is outdated and insecure. MD5 hashes can be reversed using rainbow tables or brute-force attacks due to its speed and lack of salting.



**Impact:**

* Passwords can be cracked instantly using precomputed tables.
* Enables credential reuse attacks.
* Fails compliance with modern data protection standards (e.g., OWASP, GDPR).

**Recommendation:**

* Use strong, salted hashing algorithms like:
  + bcrypt
  + scrypt
  + argon2
* Never use unsalted MD5 or SHA1 for password storage.
* Enforce strong password policies.

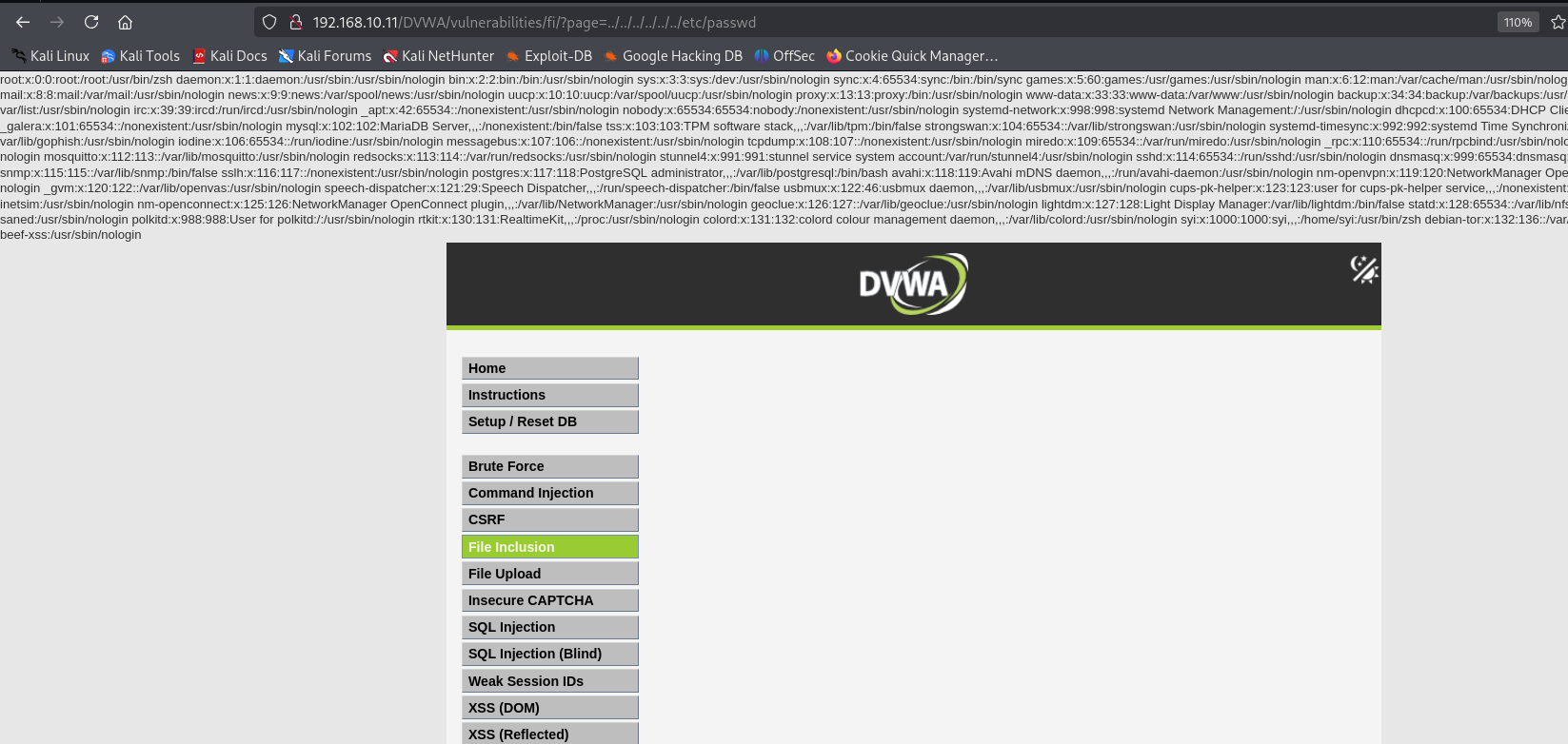
**5. Title : Local File Inclusion (LFI)**

Risk Level: High

CVSS Score: 7.0

Target URL:  
http://192.168.10.11/DVWA/vulnerabilities/fi/?page=../../../../../../etc/passwd

**Description :** The application is vulnerable to **Local File Inclusion (LFI)**, which allows an attacker to include files from the server. By manipulating the page parameter, we were able to read the contents of the sensitive file /etc/passwd.

****

**Impact:**

* Exposure of sensitive system files.
* Potential for further attacks like:
  + Log poisoning leading to RCE.
  + Reading web application source code.
  + Enumerating users and system info.

**Recommendation:**

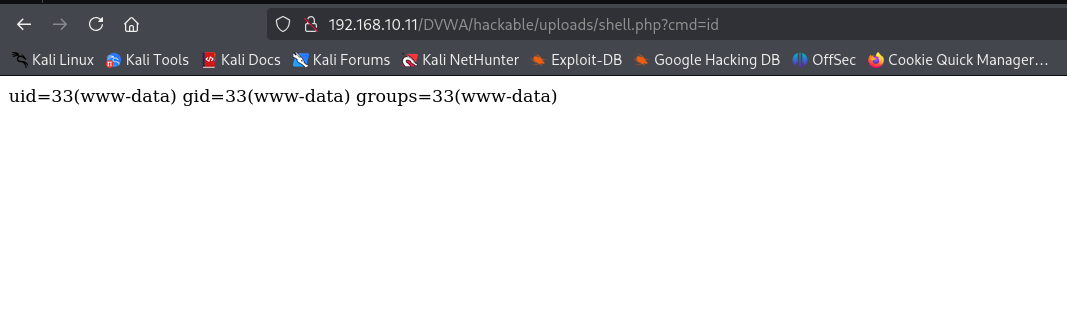
* Validate and sanitize user input strictly.
* Use whitelisting for file access.
* Disable allow\_url\_include and allow\_url\_fopen in php.ini (if applicable).
* Avoid including files based on user input wherever possible.

**6. Title : Remote Code Execution (RCE) – File Upload Abuse**

Risk Level: High

CVSS Score: 9.0

**Description:** The application allows the upload of PHP files without proper validation. An attacker can upload a malicious PHP shell script and execute arbitrary OS commands via a cmd parameter, resulting in full remote code execution under the web server context (www-data).

****

**Impact:**

* Arbitrary command execution on the server.
* Potential for full server compromise.
* Allows for privilege escalation, lateral movement, and data exfiltration.

**Recommendation:**

* Restrict file uploads to safe MIME types (e.g., .jpg, .png, .pdf).
* Implement server-side checks and validation of file extensions and MIME types.
* Rename and store uploaded files outside the web root.
* Block execution of uploaded files using .htaccess:

**Conclusion**

The assessment of DVWA revealed several exploitable vulnerabilities across key components of the application. Each issue has been thoroughly documented with proof-of-concept examples and remediation strategies. Fixing these vulnerabilities will enhance application security and better align with OWASP-recommended practices.