

---

# INTERNSHIP REPORT

---

## PROJECT 2 – WEB APPLICATION PENETRATION TESTING



JUNE 2025 - JULY 2025

**SUBMITTED BY**  
Ranjith M C

## Project 2 : Web Application Penetration Testing

### Introduction

Web applications are essential components of modern businesses, yet they are often prime targets for cyber attackers due to misconfigurations, insecure code, or poor authentication mechanisms.

This project involves performing penetration testing on **Damn Vulnerable Web Application (DVWA)**—an intentionally insecure web application designed for testing and learning purposes. The goal is to simulate real-world attacks based on the **OWASP Top 10 vulnerabilities**, the industry-standard list of the most critical security risks to web applications.

### Problem Statement

The objective of this project is to perform a comprehensive penetration test on a deliberately vulnerable web application—**Damn Vulnerable Web Application (DVWA)**. The assessment follows the **OWASP Top 10** methodology and targets several key vulnerabilities such as **Broken Authentication**, **Cross-Site Scripting (XSS)**, **SQL Injection**, and **Sensitive Data Exposure**.

The goal is to:

- Identify and exploit existing security flaws
- Understand how attackers might leverage these issues
- Recommend practical **remediation strategies** to mitigate risks and strengthen the web application's security posture

### Accessing DVWA in Kali Linux

To start testing, DVWA was launched inside **Kali Linux** using a pre-configured script.

#### ◆ Steps:

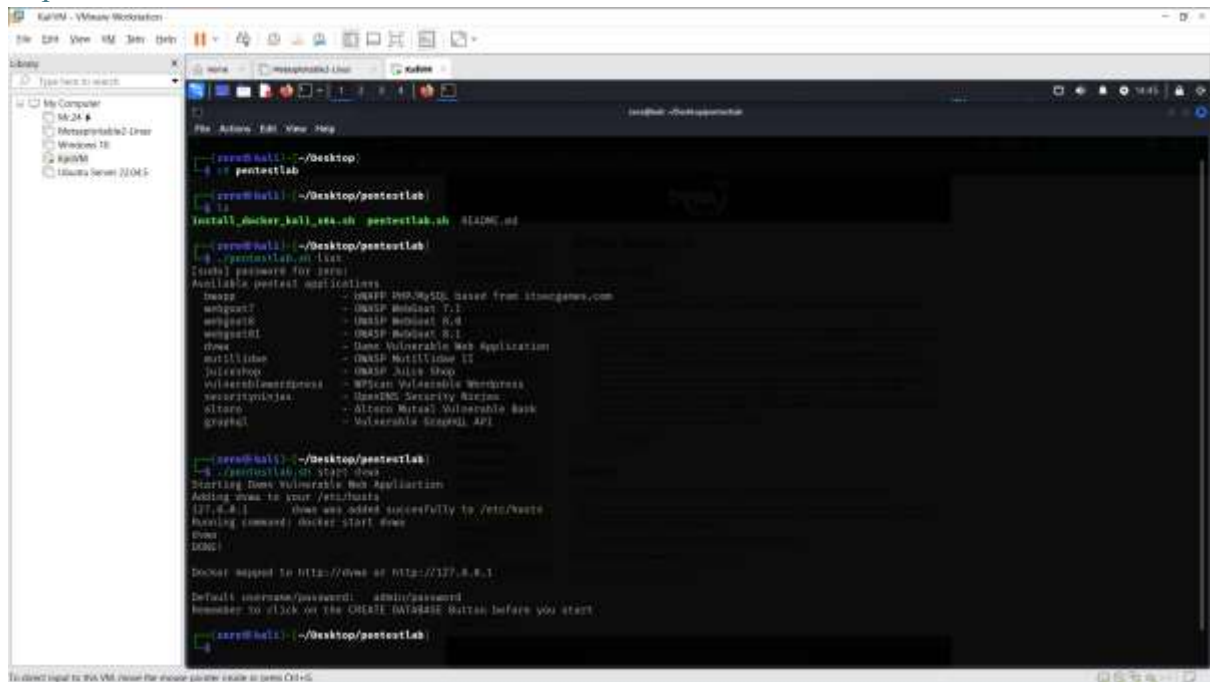
1. **Open Kali Linux** in VMware Workstation.
2. **Open Terminal** and go to the folder:  
`cd Desktop/pentestlab`
3. **Check available vulnerable apps:**  
`./pentestlab.sh list`
4. **Start DVWA:**  
`./pentestlab.sh start dvwa`

This:

- Starts DVWA in Docker
- Adds DVWA to your system hosts file
- Runs DVWA on: <http://127.8.0.1> or <http://dvwa>

5. **Open Browser** and go to:

<http://127.8.0.1>



```
root@kali: ~/Desktop
└─$ pentestlab

root@kali: ~/Desktop/pentestlab
└─$ ./pentestlab.sh

root@kali: ~/Desktop/pentestlab
└─$ ./pentestlab.sh lux

Install Docker, Kali, etc. on Kali Linux
Available pentest applications
troop - OWASP Wappal, based from itnongene.com
webpact - OWASP Wapal, based from itnongene.com
webpact - OWASP Wapal, based from itnongene.com
dms - OWASP Wapal, based from itnongene.com
mutillidae - OWASP Wapal, based from itnongene.com
juiceitop - OWASP Wapal, based from itnongene.com
vulnerablewordpress - OWASP Wapal, based from itnongene.com
vulnerablewordpress - OWASP Wapal, based from itnongene.com
altara - OWASP Wapal, based from itnongene.com
graphql - OWASP Wapal, based from itnongene.com

root@kali: ~/Desktop/pentestlab
└─$ ./pentestlab.sh start dms
Starting Dms Vulnerable Web Application
Adding dms to your /etc/hosts
127.8.8.1 dms was added successfully to /etc/hosts
Running command: docker start dms
Dms
DONE!

Docker added to http://dms or http://127.8.8.1
Default username/password: admin/password
Remember to click on the CREATE DATABASE button before you start

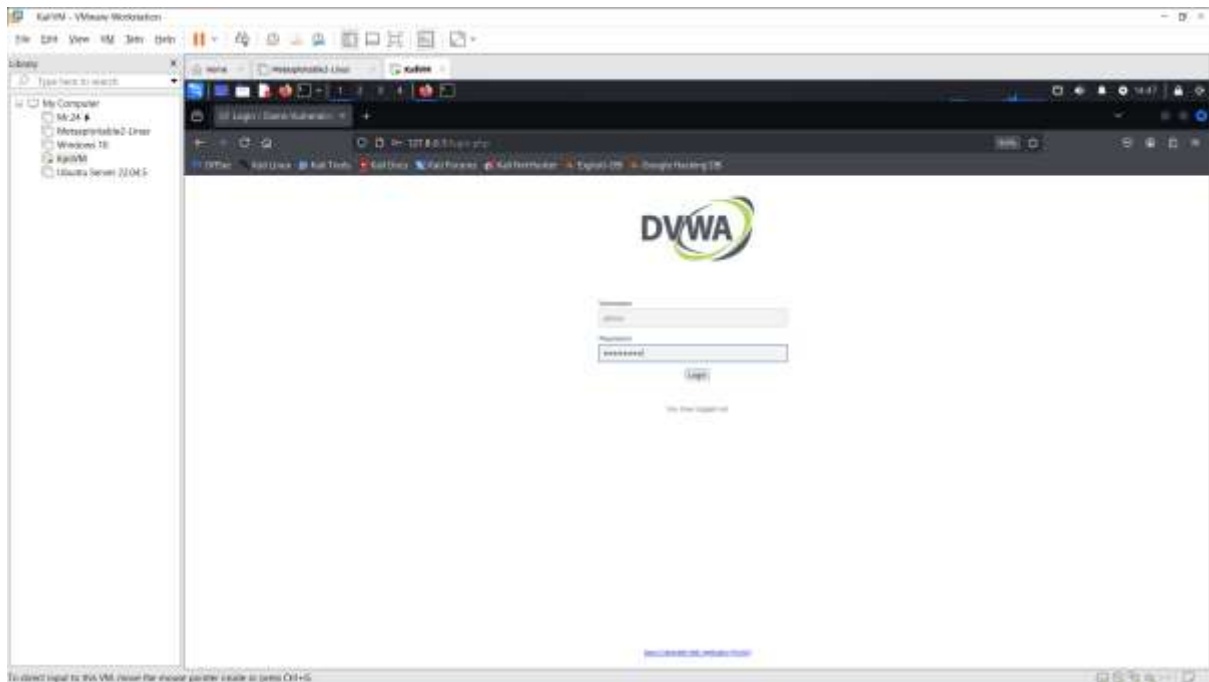
root@kali: ~/Desktop/pentestlab
```

6. **Login using default credentials:**

Username: admin

Password: password

7. **Click on "Create Database"** to finish setup.



## **Broken Authentication – Brute Force**

**Description :** Broken authentication occurs when an attacker can guess or brute-force login credentials due to weak protections. DVWA's login page is vulnerable to such attacks, allowing unauthorized access through repeated login attempts.

### **Tools Used**

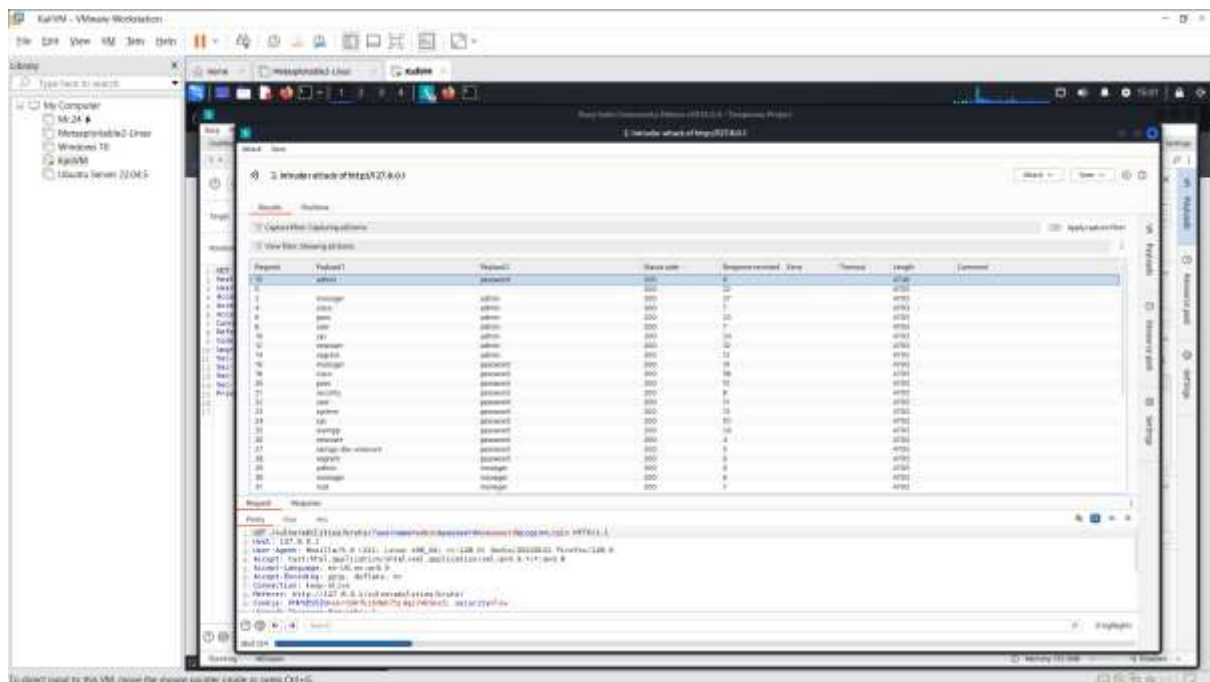
- DVWA Login page
- Burp Suite
- Wordlist file (runtime.txt with usernames and passwords)

### **Steps**





1. **Open DVWA login page** in browser.
2. **Open Burp Suite** → Turn on **Intercept**.
3. Try logging in with any details. Burp will capture the request.
4. **Send to Intruder**.
5. In **Positions**, select both username and password as **attack points**.
6. Choose **Attack type: Cluster Bomb** (to test all combinations).
7. In **Payloads**:
  - Set 1 → usernames (from file or list)
  - Set 2 → passwords (from file or list)

8. Click **Start Attack**.

9. Look for the different response that shows successful login.



## Fix (Remediation)

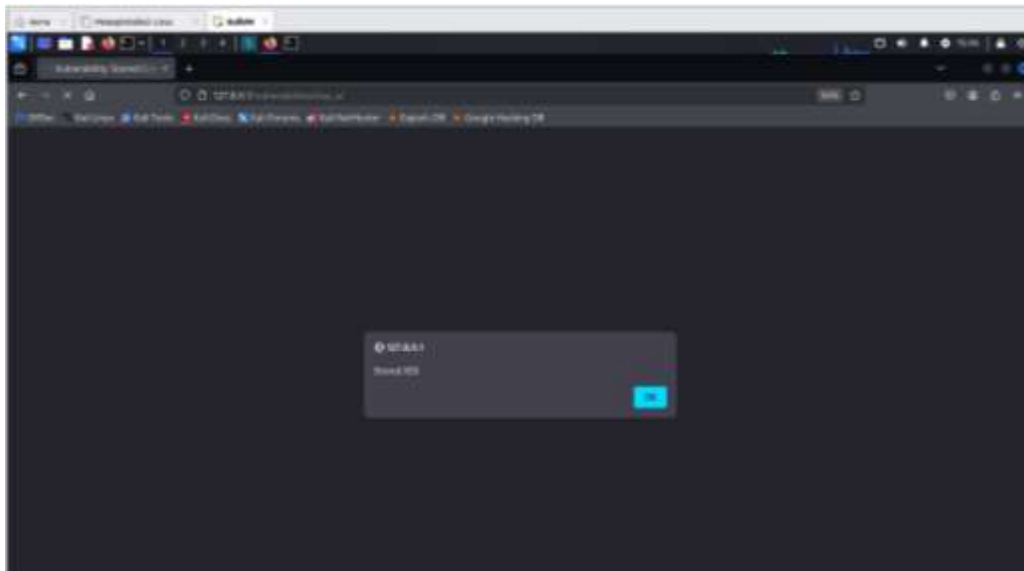
-  Add **account lockout** after 3–5 wrong attempts
-  Use **strong passwords**
-  Add **CAPTCHA**
-  Show **generic error messages** (like "Invalid login")

## ● Cross-Site Scripting (XSS) – Stored

**Description :** Stored XSS means the attacker's script is **saved in the website database** and runs **every time someone visits the page**. It can be used to steal data or show fake content.

### 🔍 Steps

1. Go to **Stored XSS** in DVWA.
2. In the form, enter:
  - **Name:** Hacker
  - **Message:** `<script>alert('Stored XSS')</script>`
3. Click **Sign Guestbook**.
4. A popup appears when the page reloads.



### 🛡️ How to Fix

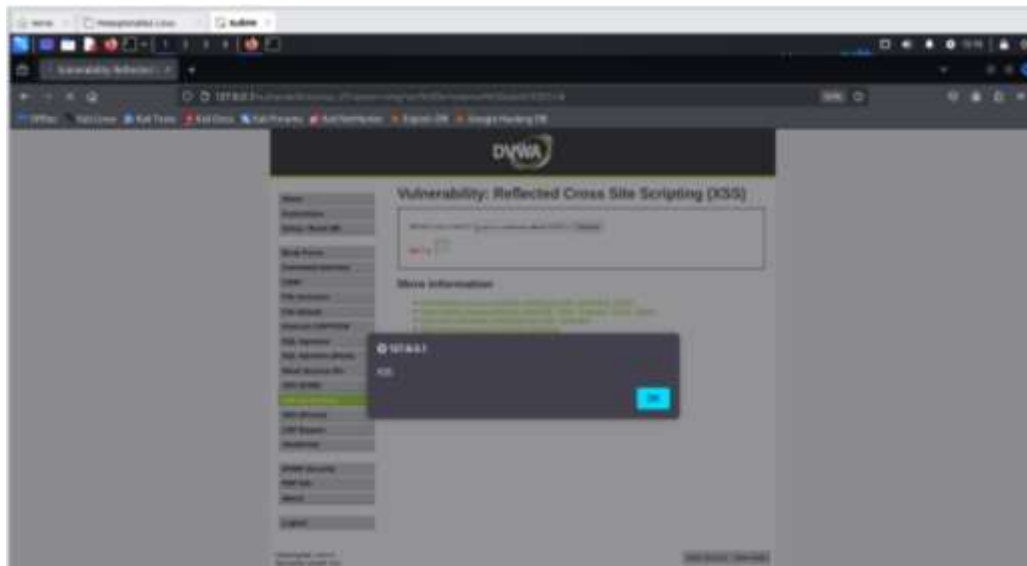
- ✂️ Remove or block `<script>` tags
- 🔒 Use input validation (allow only plain text)
- 💜 Encode special characters like `<`, `>` before saving
- 🧰 Use security tools like **DOMPurify**

## ⚠️ Cross-Site Scripting (XSS) – Reflected





**Description :** **Reflected XSS** happens when user input is **immediately displayed on the page** without checking for scripts. The script runs only once, when the user clicks a malicious link or submits a form.

## Steps

1. Go to **Reflected XSS** in DVWA.
2. In the input field that says “What’s your name?”, enter:  
`<img src=x onerror=alert('XSS')>`
3. Click **Submit**.
4. A popup appears right after submitting the form.



## How to Fix

-  Sanitize user input (remove script-related tags)
-  Encode output (turn `<` into `&lt;`, `>` into `&gt;`)
-  Use secure frameworks that handle escaping
-  Block suspicious inputs using validation

## SQL Injection

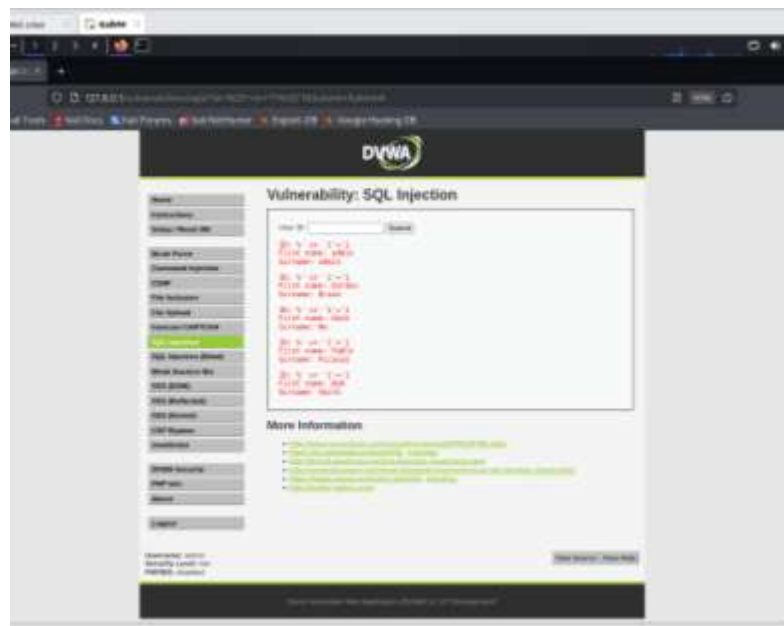
**Description :** **SQL Injection** is a vulnerability that allows attackers to **inject SQL code** into a web form or URL, tricking the database into revealing or modifying data.

## Steps





1. Go to the **SQL Injection** page in DVWA.
2. In the **User ID** field, enter:

`% ' or '1'='1`

3. Click **Submit**.



### How to Fix

-  Use **prepared statements** or **parameterized queries**
-  Don't directly insert user input into SQL queries
-  Validate and sanitize inputs
-  Use ORM (Object Relational Mapping) tools or secure database APIs


Sure! Here's the **simplified version** of the **Sensitive Data Exposure** section:

### Sensitive Data Exposure

**Description :** This happens when important data like **login sessions** or **config files** are not properly protected, allowing others to access them easily.

### What I Found

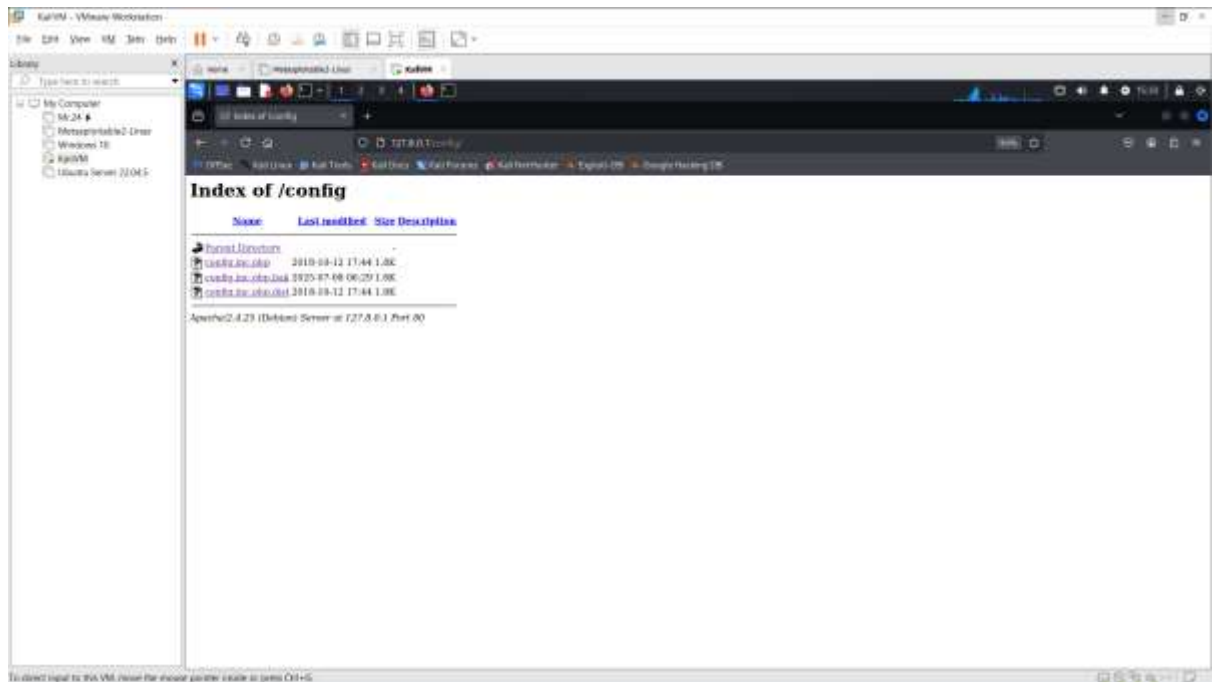
#### 1. Session Still Works in New Tab

- After logging in, I copied the URL and pasted it into a **new tab**.
- The page opened without asking for login again.
-  **Problem:** Anyone with the link can open the page and see private data.



## 2. Config File is Public

- I entered this URL:  
`http://127.8.0.1/config/`
- It showed the **config settings**.
- **Problem:** This can expose database info or system paths.



### ✓ What to Do (Fixes)

- 🗑️ End the session if user opens in a different tab or after logout
- ⌚ Add **auto logout** after inactivity
- 🚫 Block access to folders like `/config/`
- ✓ Use **HTTPS** to protect data in transit