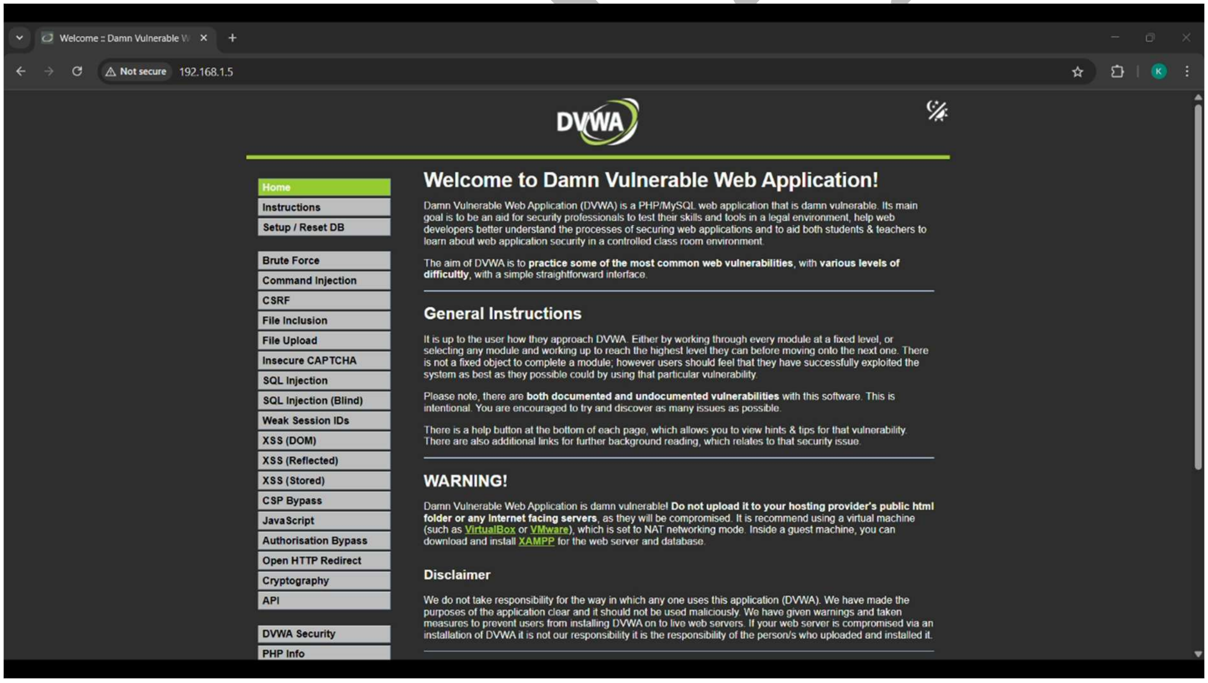
TASK-01(By ROHAN SEN)

Web Application Security Testing

Application Tested: DVWA (Damn Vulnerable Web Application)
Tools Used: Burp Suite, SQLMap.
Tested By: Rohan Sen
Email: rohasnsen1104@gmail.com

DVWA (Damn Vulnerable Web Application) is a PHP/MySQL web application that is intentionally designed to be insecure. Its primary purpose is to provide a legal environment for security professionals, students, and researchers to practice and improve their penetration testing skills. It includes multiple vulnerability levels for different attack types such as SQL Injection, Cross-Site Scripting (XSS), Brute Force, and more.

In this project, I performed manual and automated testing on DVWA using various tools to identify and exploit common web application vulnerabilities. This report details the vulnerabilities found, how they were exploited, and how they can be mitigated.



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

1. Introduction:

This report summarizes the findings of a web application security testing task performed on **DVWA (Damn Vulnerable Web Application)**. The objective of this task was to identify and understand common web application vulnerabilities using real-world techniques and automated tools. The vulnerabilities tested in this assessment include SQL Injection, Cross-Site Scripting (XSS), and Authentication Flaws.

2. Tools Used:

The following tools were used in conducting the security assessment:

- **Burp Suite**: A web vulnerability scanner and proxy tool for identifying security flaws in web applications.
- **SQLMap**: An automated tool for detecting and exploiting SQL injection vulnerabilities.
- **DVWA (Damn Vulnerable Web Application)**: The target web application used for testing.

3. Methodology:

The security assessment followed a structured methodology to test for common web vulnerabilities:

1. **Reconnaissance**:
   Initial scanning of the application to identify attack surfaces and potential weaknesses.
2. **Vulnerability Scanning**:
   Automated tools like **Burp Suite** was used to scan the application for known vulnerabilities.
3. **Exploitation**:
   Exploiting identified vulnerabilities using **SQLMap** for SQL injection and manual testing for XSS vulnerabilities.
4. **Reporting**:
   Documenting the vulnerabilities found, the exploitation methods used, and recommended mitigation strategies.
5.

4. Vulnerabilities Identified

4.1 SQL Injection (SQLi)

Description:
SQL Injection is a common web application vulnerability where malicious SQL code is injected into a query, potentially allowing unauthorized access to the database.

Test Method:

- Manual exploitation and use of **SQLMap** to test for SQL injection points.
- Success was achieved by injecting SQL queries in the input fields that were not sanitized properly.

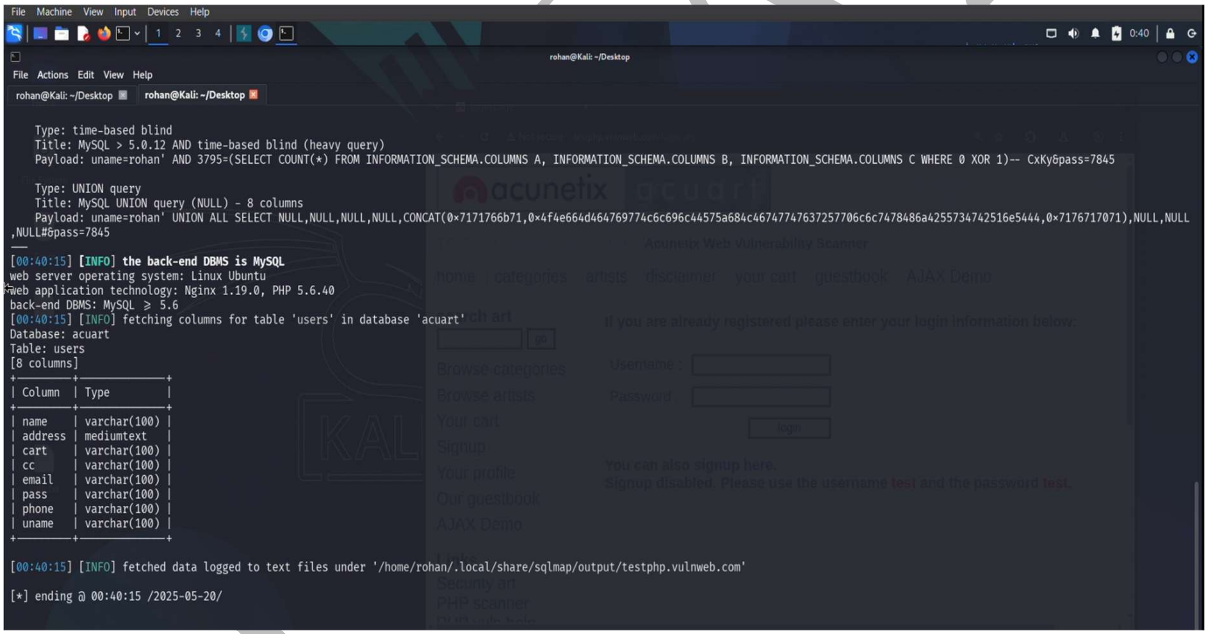
Example of Attack:

- Inputting ' OR '1'='1 in the login form resulted in unauthorized access to the admin panel.

Mitigation Strategy:

- Use parameterized queries and prepared statements to prevent SQL injection.
- Validate and sanitize user input rigorously.

Here's an image demonstrating the successful SQL injection:



```
    Type: time-based blind
    Title: MySQL > 5.0.12 AND time-based blind (heavy query)
    Payload: uname=rohan' AND 3795=(SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS A, INFORMATION_SCHEMA.COLUMNS B, INFORMATION_SCHEMA.COLUMNS C WHERE 0 XOR 1)-- CxKy&pass=7845

    Type: UNION query
    Title: MySQL UNION query (NULL) - 8 columns
    Payload: uname=rohan' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x7171766b71,0x4f4e664d464769774c6c696c44575a684c46747747637257706c6c7478486a4255734742516e5444,0x7176717071),NULL,NULL
,NULL#&pass=7845
---
[00:40:15] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.6
[00:40:15] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+

[00:40:15] [INFO] fetched data logged to text files under '/home/rohan/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 00:40:15 /2025-05-20/
```

4.2 Cross-Site Scripting (XSS)

Description:
Cross-Site Scripting (XSS) vulnerabilities allow attackers to inject malicious scripts into web pages, which are executed in the context of the user's browser.

Test Method:

- Identifying potential XSS vulnerabilities by injecting malicious scripts into form fields.
- **Reflected XSS** was tested by injecting a script into the input field to steal the session ID.

Example of Attack:

- Injecting <script>alert('XSS')</script> into the search input field displayed a pop-up in the browser.

Mitigation Strategy:

- Sanitize all user input to remove harmful scripts.
- Use content security policies (CSP) and HttpOnly cookies to protect against script execution.

4.3 Authentication Flaws (Brute-Force Attack Using Burp Suite):

Description:

Authentication mechanisms are the front line of defense for any web application. Weaknesses in authentication systems can allow attackers to gain unauthorized access through brute-force attacks, default credentials, or insecure login flows.

Test Case:

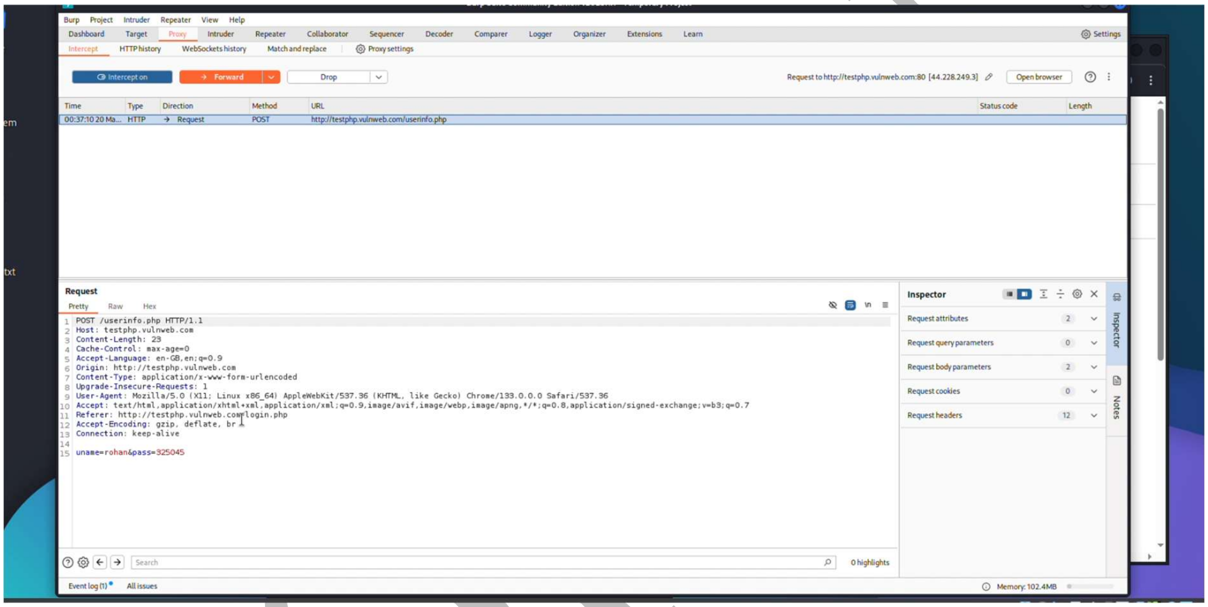
To assess authentication flaws in DVWA, I targeted the login functionality using a brute-force attack. The steps were as follows:

1. Intercepted the Login Request
   - Used Burp Suite to intercept a valid login attempt.
   - Captured the POST request structure with parameters username and password.
2. Configured Burp Intruder
   - Set the attack type to Sniper or Cluster Bomb depending on whether one or both fields were targeted.
   - Loaded a list of common usernames and passwords (e.g., admin, password, 123456, etc.).
3. Launched the Brute-Force Attack
   - Observed changes in server response (status codes or content length).

- Identified a successful login when the server responded with a different pattern or redirected to the dashboard.

Result:

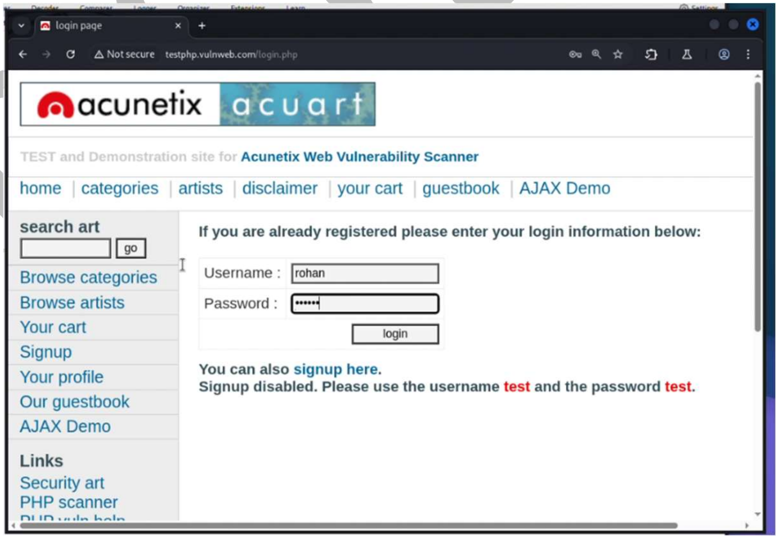
- DVWA allowed unlimited login attempts without CAPTCHA or account lockout.
- Successfully brute-forced valid credentials.
- Session token was issued without any security flags (Secure, HttpOnly, SameSite).



```
Time               Type   Direction   Method   URL
00:37:10 20 Ma...  HTTP   → Request   POST     http://testphp.vulnweb.com/userinfo.php
```

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 23
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

uname=rohan&pass=325045
```



testphp.vulnweb.com/login.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

If you are already registered please enter your login information below:

Username :	rohan
Password :	••••••

You can also signup here.
Signup disabled. Please use the username test and the password test.

Risk Level: High

Authentication flaws such as unrestricted brute-force can allow attackers to compromise user accounts, especially when users use weak or reused passwords.

5. Summary of Findings:

The security assessment of the **DVWA** revealed the following vulnerabilities:

- **SQL Injection** in unfiltered input fields.
- **Cross-Site Scripting (XSS)** vulnerability in user input handling.
- **Authentication flaws** related to session management.

6. Mitigation Recommendations:

- SQL Injection:
  - Always use parameterized queries and prepared statements.
  - Apply proper input validation to prevent malicious input.
- XSS:
  - Sanitize user inputs by escaping special characters.
  - Use security mechanisms like Content Security Policy (CSP) and HttpOnly cookies.
- Authentication Flaws:
  - Implement strong password policies and session management.
  - Use multi-factor authentication (MFA) wherever possible.

7. Conclusion:

This task helped me to understand common web application vulnerabilities and how they can be exploited in real-world scenarios. Using automated tools and manual techniques, I was able to successfully identify vulnerabilities in the **DVWA** application. The task also provided valuable experience in performing web application security assessments and recommending security improvement.