Task 7 – Web Application Vulnerability Testing using DVWA

Internship: Cyber Security Internship

Name: Adil Firoz

Date: 26-01-2026

Operating System: Kali Linux

Web Server: Apache2

Database: MariaDB (MySQL)

Application: Damn Vulnerable Web Application (DVWA)

Tools Used: DVWA, Web Browser, Linux Terminal

1. Objective

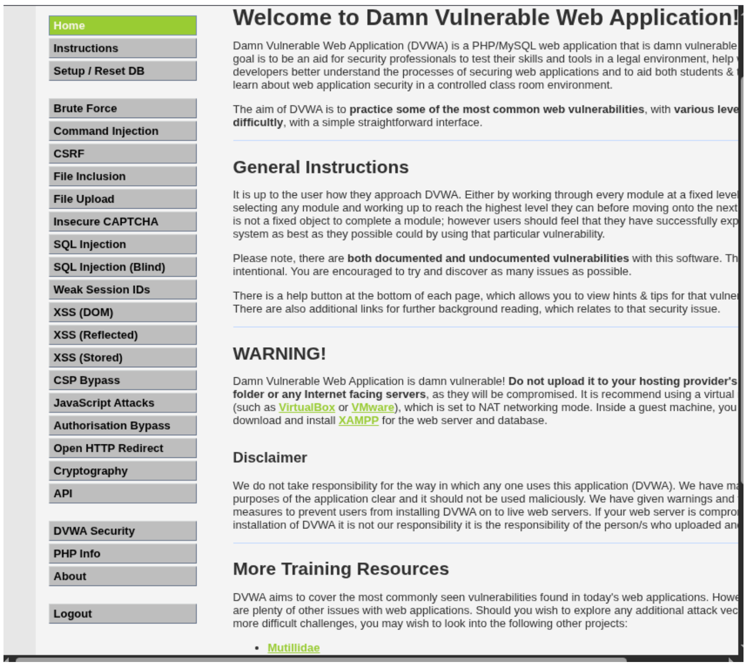
The objective of this task is to understand and analyze common web application vulnerabilities by using Damn Vulnerable Web Application (DVWA). The task focuses on identifying and exploiting vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Brute Force attacks in a controlled lab environment to gain practical knowledge of web application security weaknesses.

2. Environment Setup

The testing environment was configured using the following components:

- Operating System: Kali Linux
- Web Server: Apache2
- Database: MariaDB (MySQL)
- Application: Damn Vulnerable Web Application (DVWA)

DVWA was successfully installed and configured on the local server. The database was created using the setup option, and the security level was set to **Low** to allow demonstration of vulnerabilities.



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable
goal is to be an aid for security professionals to test their skills and tools in a legal environment, help
developers better understand the processes of securing web applications and to aid both students &
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 leve**
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
selecting any module and working up to reach the highest level they can before moving onto the next
is not a fixed object to complete a module; however users should feel that they have successfully exp
system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. Th
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 vulne
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**
folder or any Internet facing servers, as they will be compromised. It is recommend using a virtual
(such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you
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 ma
purposes of the application clear and it should not be used maliciously. We have given warnings and
measures to prevent users from installing DVWA on to live web servers. If your web server is compro
installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded an

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. Howe
are plenty of other issues with web applications. Should you wish to explore any additional attack vec
more difficult challenges, you may wish to look into the following other projects:

- Mutillidae

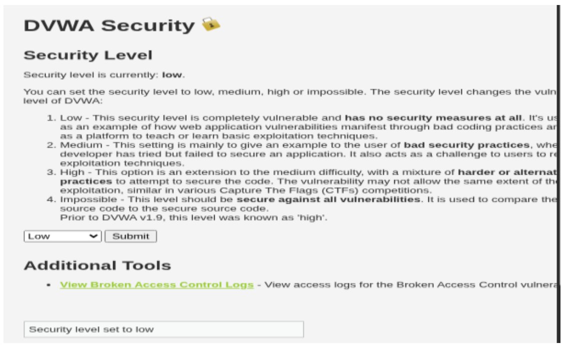
Figure 1: DVWA successfully installed and running on localhost

3. Methodology

The testing methodology involved manually interacting with DVWA modules at the low security level. Crafted inputs were provided in different vulnerability modules, and application responses were analyzed to confirm successful exploitation. Screenshots were captured as evidence of each vulnerability.

4. Security Level Configuration

The DVWA security level was set to **Low** in order to disable protective mechanisms and allow demonstration of vulnerabilities. This configuration makes the application intentionally insecure for learning and testing purposes.



DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vuln
level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's us
   as an example of how web application vulnerabilities manifest through bad coding practices ar
   as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, whe
   developer has tried but failed to secure an application. It also acts as a challenge to users to re
   exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternat**
   practices to attempt to secure the code. The vulnerability may not allow the same extent of th
   exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the
   source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

Low | Submit

Additional Tools

- View Broken Access Control Logs - View access logs for the Broken Access Control vulnera

Security level set to low

Figure 2: DVWA security level configured to Low

5. SQL Injection Testing

Description

SQL Injection occurs when user input is directly inserted into SQL queries without proper validation. This allows attackers to manipulate database queries.

Test Performed

Input used:

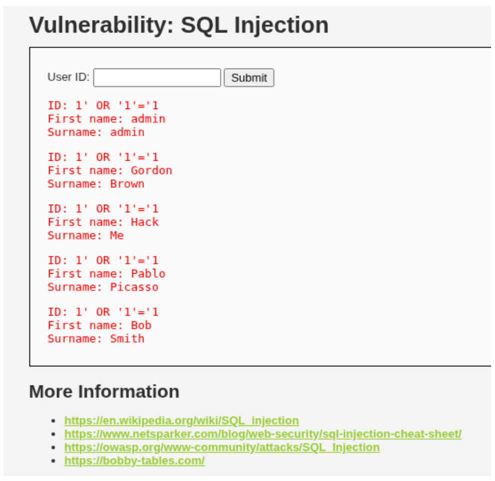
' OR '1'='1

Result

The application returned multiple user records from the database, confirming that SQL Injection was successful.

Observation

The vulnerability allows unauthorized access to sensitive database information.



```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

Figure 3: Successful SQL Injection showing multiple user records

6. Cross-Site Scripting (XSS) Testing

Description

Cross-Site Scripting occurs when malicious scripts are injected into web pages viewed by other users.

Test Performed

Input used:

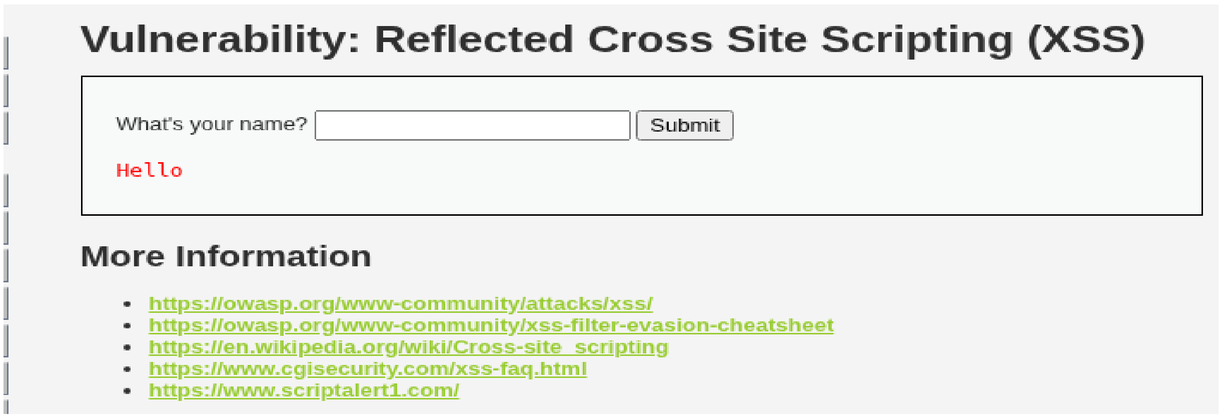
<script>alert('XSS')</script>

Result

The script executed successfully, displaying a browser alert.

Observation

This vulnerability allows attackers to execute arbitrary JavaScript, potentially stealing cookies or session information.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

Figure 4: Reflected XSS vulnerability execution

7. Brute Force Attack Testing

Description

Brute force attacks attempt multiple username and password combinations to gain unauthorized access.

Test Performed

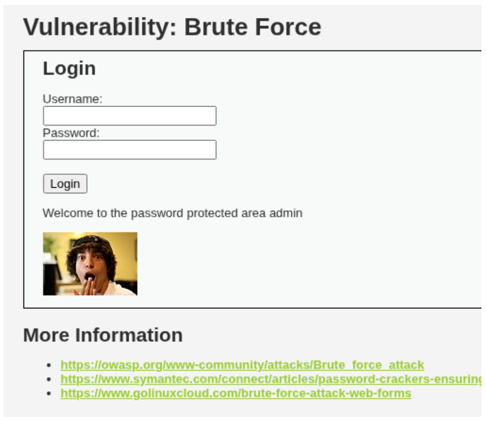
Multiple login attempts were performed on the Brute Force module.

Result

Valid credentials were eventually accepted, granting access to the protected area.

Observation

Weak authentication mechanisms make accounts vulnerable to password guessing attacks.



Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area admin

More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- https://www.symantec.com/connect/articles/password-crackers-ensuring
- https://www.golinuxcloud.com/brute-force-attack-web-forms

Figure 5: Successful brute force login as administrator

8. Results

- SQL Injection allowed unauthorized retrieval of database records.
- XSS enabled execution of malicious scripts in the browser.
- Brute force attack successfully bypassed authentication controls.

These vulnerabilities demonstrate how insecure coding practices can compromise web applications.

9. Impact

In real-world applications, these vulnerabilities can lead to:

- Data leakage
- Account compromise
- Unauthorized system access
- Session hijacking and user impersonation

10. Recommendations

To mitigate these vulnerabilities, the following measures are recommended:

- Use prepared statements and parameterized queries to prevent SQL Injection
- Implement input validation and output encoding to prevent XSS
- Enforce strong password policies and account lockout mechanisms
- Apply secure coding practices and regular vulnerability testing

11. Conclusion

This task provided hands-on experience in identifying and exploiting common web application vulnerabilities using DVWA. The experiments demonstrated the critical importance of secure coding and proper input validation. Understanding these vulnerabilities helps in designing more secure web applications and protecting systems from real-world attacks.