

BUG BOUNTY REPORT

Reflected Cross-Site Scripting (XSS)

Submitted By: Tooba Zainab

1. Basic Information

Project Name: Bounty Hunt 101 – Recon to Report

Target Application: DVWA (Damn Vulnerable Web Application)

Application URL: <http://localhost/DVWA>

Vulnerability Type: Reflected Cross-Site Scripting (XSS)

Severity Level: Medium

Test Environment:

- Kali Linux
 - DVWA (Local Lab)
 - Apache, PHP, MariaDB
-

2. Vulnerability Summary

A **Reflected Cross-Site Scripting (XSS)** vulnerability was discovered in the DVWA application under the **XSS (Reflected)** module.

This vulnerability allows an attacker to inject and execute malicious JavaScript code in the victim's browser.

The issue occurs due to **lack of proper input validation and output encoding**.

3. Environment Setup Confirmation

The DVWA application was successfully installed and configured on Kali Linux.

The application was accessible via browser and the database was initialized correctly.



Username	<input type="text"/>
Password	<input type="password"/>
	<input type="button" value="Login"/>

4. Reconnaissance Performed

An Nmap scan was performed on the target system to identify open ports and running services.

Tool Used: Nmap

Command Used:

```
nmap -sV -p- localhost
```

The scan revealed active services such as Apache web server and MySQL database, confirming a valid web attack surface.

5. Security Configuration

Before testing, the DVWA security level was set to **Low** to simulate a vulnerable application environment.

DVWA Security

Security Level

Security level is currently: **impossible**.

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

1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve 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**, where the developer has failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the 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 vulnerable source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

Low

Additional Tools

- [View Broken Access Control Logs](#) - View access logs for the Broken Access Control vulnerability

6. Steps to Reproduce the Vulnerability

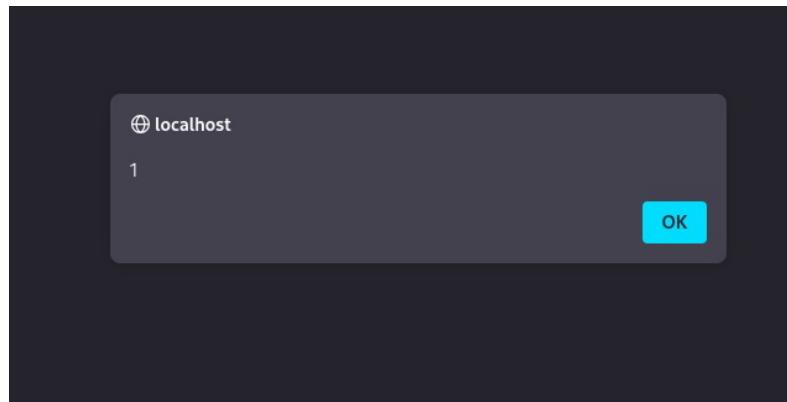
1. Login to DVWA using default credentials:
 - o Username: admin
 - o Password: password
2. Navigate to **DVWA Security** and set the security level to **Low**.
3. Click on **XSS (Reflected)** from the left menu.
4. Enter the following payload into the input field:
5. <script>alert(1)</script>
6. Click the **Submit** button.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

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/>



7. Proof of Concept (PoC)

Payload Used

```
<script>alert(document.cookie)</script>
```

Result

A JavaScript alert popup appeared displaying session cookies, confirming successful execution of injected JavaScript code.

The image shows a web application interface for testing XSS vulnerabilities. At the top, a green bar contains the text "Vulnerability: Reflected Cross Site Scripting (XSS)". Below this is a form with a question "What's your name?". Inside the input field, the user has typed "`t(document.cookie)</script>`". Next to the input field is a "Submit" button. Below the form, the word "Hello" is displayed in red text. Further down, there is a section titled "More Information" with two links:

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>

At the bottom of the page is another dark-themed browser window. It shows the URL "localhost" with a globe icon. The main content area displays the session ID: "security=low; PHPSESSID=fefc9f86a54c23afedab35fbb0ba3f30". In the bottom right corner of this window is a blue "OK" button.

8. Impact Analysis

If exploited in a real-world application, this vulnerability could allow an attacker to:

- Steal session cookies
- Hijack user sessions
- Perform phishing attacks

- Execute malicious scripts in the user's browser
 - Potentially gain unauthorized access to user accounts
-

9. Evidence Summary

The following evidence was collected during testing:

- DVWA running successfully
 - Security level set to Low
 - XSS alert popup
 - Cookie disclosure popup
-

10. Recommendations

To mitigate this vulnerability, the following actions are recommended:

- Validate and sanitize all user inputs
 - Encode output before rendering it in the browser
 - Implement Content Security Policy (CSP)
 - Avoid directly reflecting user input in responses
-

11. Conclusion

This report demonstrates a successfully identified and exploited **Reflected XSS vulnerability** in a controlled lab environment.

The vulnerability highlights the importance of secure input handling and output encoding in web applications.

All testing was performed **ethically and within scope** for educational purposes only.

12. Tester Declaration

This vulnerability assessment was conducted solely for academic learning in a safe lab environment and did not target any real-world systems.