

Cyber Security Project: Bounty Hunt 101

Bug Hunt 101 — Recon to Report

1. Introduction

This project demonstrates a beginner-level bug bounty workflow performed in a controlled lab environment using **Damn Vulnerable Web Application (DVWA)**. The objective was to understand the end-to-end vulnerability assessment process including environment setup, reconnaissance, exploitation of common web vulnerabilities, and professional reporting.

2. Objective

The main objectives of this project are:

- To understand how real-world bug bounty testing is performed.
- To identify common web vulnerabilities such as Cross-Site Scripting (XSS) and SQL Injection.
- To exploit vulnerabilities safely within the defined scope.
- To document findings with proof of concept and impact analysis.

3. Scope

- **Target Application:** Damn Vulnerable Web Application (DVWA)
- **Hosting:** Localhost (127.0.0.1)
- **Testing Scope:** Only the locally hosted DVWA instance
- **Out of Scope:** Any external websites or systems

All testing was performed ethically and strictly within the lab environment.

4. Environment Setup

- **Operating System:** Kali Linux
- **Web Server:** Apache
- **Backend Database:** MySQL
- **Application:** DVWA v1.10
- **Security Level:** Low
- **Testing Tools Used:**
 - Web Browser

- o Burp Suite
- o Nmap

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, it will be cleared and the data will be reset.
You can also use this to reset the administrator credentials ("admin // password") at any stage.

Setup Check

Operating system: *nix
Backend database: MySQL
PHP version: 7.0.30-0+deb9u1

Web Server SERVER_NAME: 127.0.0.1

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

All Products

	Apple Juice (1000ml) 1.99		Apple Pomace 0.89		Banana Juice (1000ml) 1.99
--	------------------------------	--	----------------------	--	-------------------------------

This website uses fruit cookies to ensure the juiciest tracking experience.
But me wait!

Me want it!

5. Reconnaissance

Reconnaissance was performed to identify open ports and running services on the target system.

Tool Used: Nmap

Command:

```
nmap -sC -sV -p- 127.0.0.1
```

Purpose:

- Identify open ports
- Detect running services and versions
- Understand application exposure

Findings:

- Port 80 (HTTP) was open
- Apache web server was running
- PHP-based web application detected.

```
(harshitha㉿kali)-[~]
└─$ nmap -sC -sV -p- 127.0.0.1

Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-15 20:37 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000060s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.2p1 Debian 2 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
| http-title: Login :: Damn Vulnerable Web Application (DVWA) v1.10 *Develop ...
|_Requested resource was login.php
|_http-server-header: Apache/2.4.25 (Debian)
| http-robots.txt: 1 disallowed entry
|_/
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
3000/tcp  open  ppp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Access-Control-Allow-Origin: *
|     X-Content-Type-Options: nosniff
|     X-Frame-Options: SAMEORIGIN
|     Feature-Policy: payment 'self'
|     X-Recruiting: /#/jobs
|     Accept-Ranges: bytes
|     Cache-Control: public, max-age=0
|     Last-Modified: Thu, 15 Jan 2026 15:05:45 GMT
|     ETag: W/"1252f-19bc230ee0e"
|     Content-Type: text/html; charset=UTF-8
|     Content-Length: 75055
```

```

Session Actions Edit View Help
Content-Length: 75055
Vary: Accept-Encoding
Date: Thu, 15 Jan 2026 15:09:45 GMT
Connection: close
<!--
Copyright (c) 2014-2026 Bjoern Kimminich & the OWASP Juice Shop contributors.
SPDX-License-Identifier: MIT
<!doctype html>
<html lang="en" data-beasties-container>
<head>
<meta charset="utf-8">
<title>OWASP Juice Shop</title>
<meta name="description" content="Probably the most modern and sophisticated insecure web application">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link id="favicon" rel="icon"
HTTPOptions, RTSPRequest:
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Vary: Access-Control-Request-Headers
Content-Length: 0
Date: Thu, 15 Jan 2026 15:09:45 GMT
Connection: close
Help, NCP:
HTTP/1.1 400 Bad Request
Connection: close
2855/tcp open http Golang net/http server
_http-title: Site doesn't have a title (text/plain; charset=utf-8).
_fingerprint-strings:
FourOhFourRequest:
HTTP/1.0 404 Not Found
Date: Thu, 15 Jan 2026 15:09:55 GMT
Content-Length: 19
Content-Type: text/plain; charset=utf-8

```



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 difficulty**, 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 vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users!)



Username

Password

[Logout](#)

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.

More Training Resources

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

- [bWAPP](#)
- [NOWASP](#) (formerly known as [Mutilidae](#))
- [OWASP Broken Web Applications Project](#)

You have logged in as 'admin'

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 vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. Its 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 tried but 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'.

6. Vulnerability Assessment & Exploitation

6.1 Reflected Cross-Site Scripting (XSS)

Vulnerability Type: Reflected XSS

Severity: Medium

Affected Module: XSS (Reflected)

Description

Reflected XSS occurs when user-supplied input is immediately reflected in the response without proper validation or encoding.

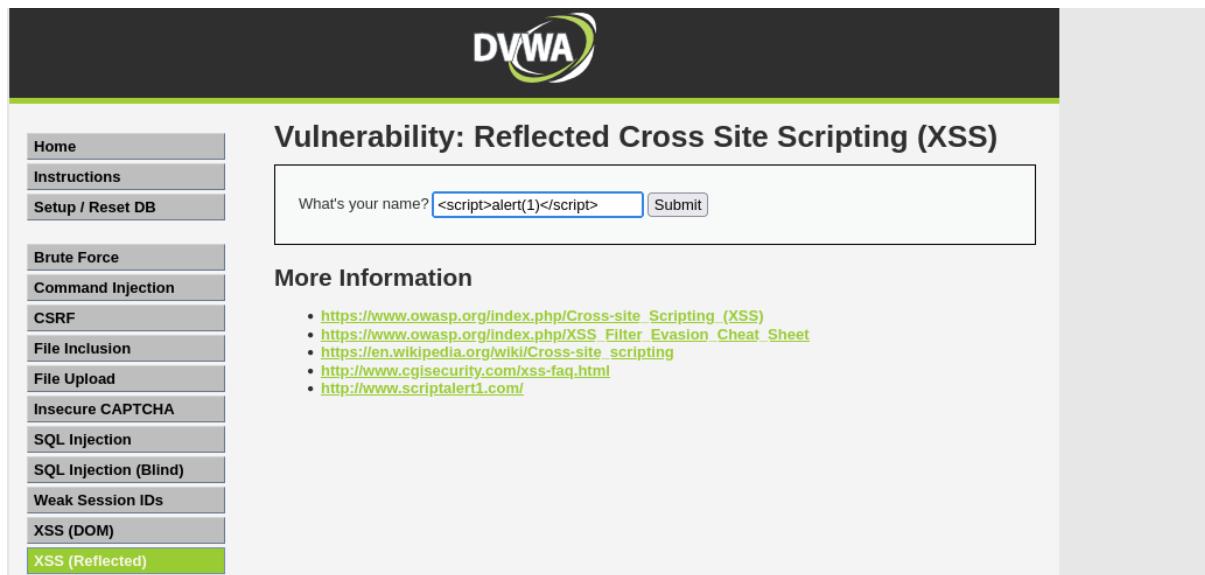
Steps to Reproduce

1. Navigate to **XSS (Reflected)** module in DVWA

```
<script>alert(1)</script>
```

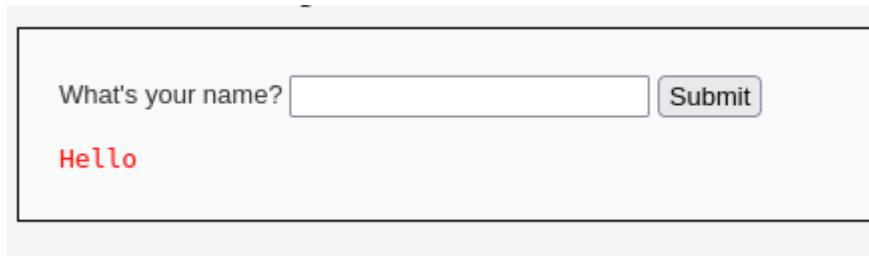
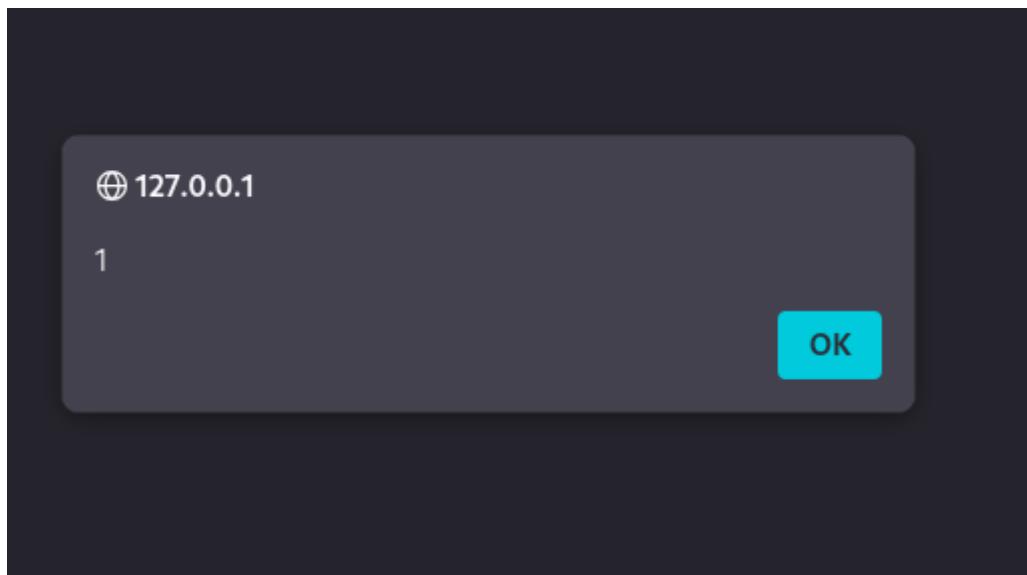
Proof of Concept

- A JavaScript alert popup was triggered, confirming successful execution.



The screenshot shows the DVWA application interface. On the left is a sidebar with various security test categories. The 'XSS (Reflected)' option is highlighted with a green background. The main content area has a title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below the title is a form field with placeholder text 'What's your name? <script>alert(1)</script>' and a 'Submit' button. To the right of the form, under the heading 'More Information', is a bulleted list of links related to XSS.

- [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS Filter Evasion Cheat Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>



The screenshot shows the DVWA application after a successful attack. The main content area displays the text 'Hello' in red, which was reflected from the user input 'What's your name? Hello'. Below this, there is a 'Submit' button.

Impact

An attacker can execute arbitrary JavaScript in the victim's browser, which can lead to:

- Session hijacking
- Phishing attacks

- Credential theft

Recommendation

- Implement proper input validation
- Encode output before rendering
- Use Content Security Policy (CSP)

6.2 Stored Cross-Site Scripting (XSS)

Vulnerability Type: Stored XSS

Severity: High

Affected Module: XSS (Stored)

Description

Stored XSS occurs when malicious input is stored on the server and executed every time a user accesses the affected page.

Steps to Reproduce

1. Navigate to **XSS (Stored)** module

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

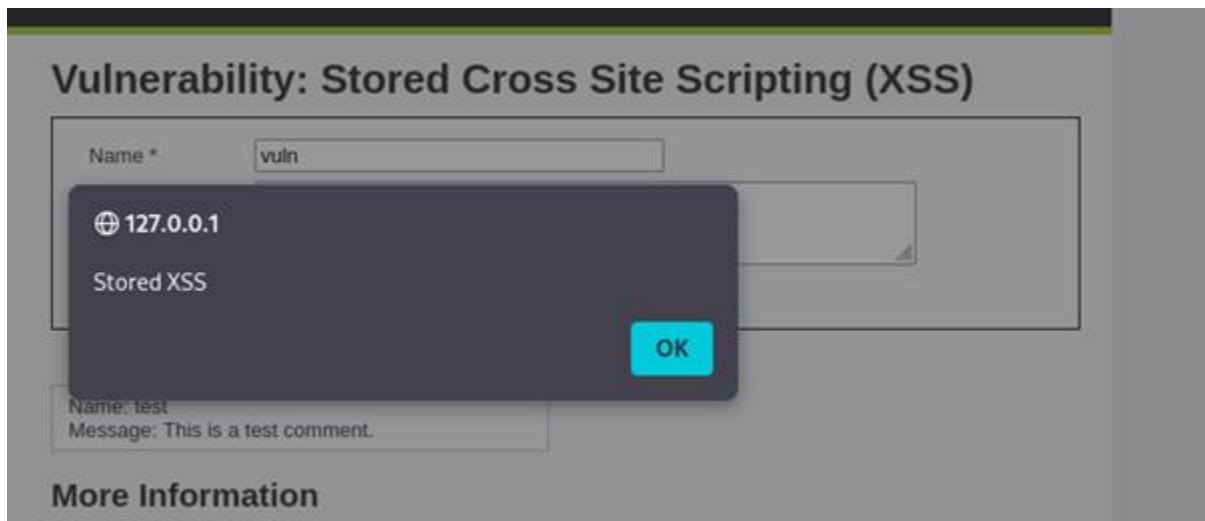
Proof of Concept

- JavaScript alert executed automatically on page load.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="vuln"/>
Message *	<input type="text" value="<script>alert('Stored XSS')</script>"/>
<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>	

Name: test
Message: This is a test comment.



Impact

Stored XSS is more dangerous as it affects all users accessing the page and can result in:

- Persistent malicious script execution
- Large-scale user compromise

Recommendation

- Sanitize and validate user input
- Encode stored content
- Restrict script execution

6.3 SQL Injection

Vulnerability Type: SQL Injection

Severity: High

Affected Module: SQL Injection

Description

SQL Injection occurs when user input is directly used in database queries without proper sanitization.

Steps to Reproduce

1. Navigate to **SQL Injection** module

' OR '1'='1

Proof of Concept

- Application returned multiple user records, indicating query manipulation.

Vulnerability: SQL Injection

User ID:

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

User ID:

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

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

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

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

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

Impact

SQL Injection can lead to:

- Unauthorized access to sensitive data
- Data modification or deletion
- Complete database compromise

Recommendation

- Use prepared statements and parameterized queries
- Validate user input

- Apply least privilege access to database users

6.4 Command Injection

Vulnerability Type: Command Injection

Severity: High

Affected Module: Command Injection

Description

Command Injection is a vulnerability where user-supplied input is passed directly to system-level commands without proper validation. This allows an attacker to execute arbitrary operating system commands on the server.

Steps to Reproduce

1. Navigate to the **Command Injection** module in DVWA.
2. In the input field (IP address field), enter the following payload:

127.0.0.1; whoami

Proof of Concept

- The application executed the `whoami` command after the IP address.
- The output revealed the server user as `www-data`.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.owasp.org/index.php/Command_Injection

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.129 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.110 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.174 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.110 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.110/0.131/0.174/0.026 ms
www-data
```

[More Information](#)

Impact

Successful command injection allows an attacker to:

- Execute arbitrary system commands
- Access sensitive files
- Escalate privileges
- Potentially gain full control over the server

This vulnerability can lead to complete server compromise.

Recommendation

- Validate and sanitize user input
- Avoid using system commands with user-controlled input
- Use safer APIs instead of direct command execution
- Apply least privilege to web server users

7. Conclusion

This project successfully demonstrated the complete bug bounty workflow, from reconnaissance to exploitation and reporting. The identified vulnerabilities highlight the importance of secure coding practices such as input validation, output encoding, and secure database interaction. Performing such assessments in a controlled environment helps in understanding real-world security risks without causing harm.

8. Ethical Disclaimer

All testing activities were performed on a deliberately vulnerable application hosted locally for educational purposes only. No real-world systems or unauthorized targets were tested.