|  |
| --- |
|  |
| Web Application Security Testing Report |
| Task 1 |

|  |
| --- |
| Jeffery Tetteyfio  4-17-2025 |

Contents

[**1. Executive Summary** 1](#_Toc195750415)

[**2. Tools Used** 1](#_Toc195750416)

[**3. Testing Procedure** 2](#_Toc195750417)

[**3.1 Test Cases:** 2](#_Toc195750418)

[**4.** **Findings & Risk Assessment** 6](#_Toc195750419)

[**5.** **Recommendations** 6](#_Toc195750420)

[**6.** **Conclusion** 7](#_Toc195750421)

## **Introduction**

This report documents the discovery of **SQL injection (SQLi)** vulnerabilities in DVWA (Damn Vulnerable Web App). Manual testing revealed critical flaws allowing unauthorized database access, including extraction of user credentials.

## **2. Tools Used**

| **Tool** | **Purpose** |
| --- | --- |
| DVWA | Vulnerable test environment |
| Web Browser (Chrome/Firefox) | Manual testing interface |
| OWASP ZAP | Automated vulnerability scanning (optional) |

## **3. Testing Procedure**

Step 1: Setup

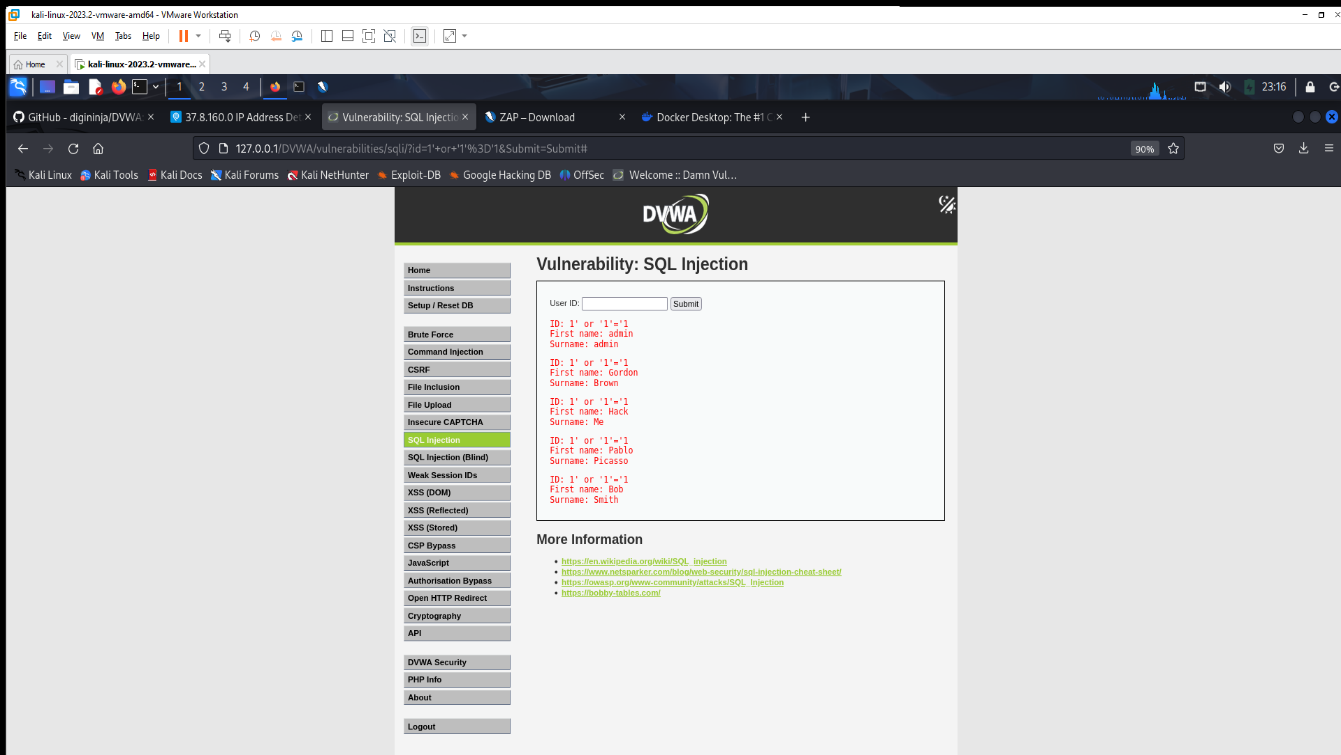
1. Launched DVWA on http://127.0.0.1
2. Set security level to **"Low"** (DVWA - Security)

### **3.1 Test Cases:**

*■* ***SQL Injection:***

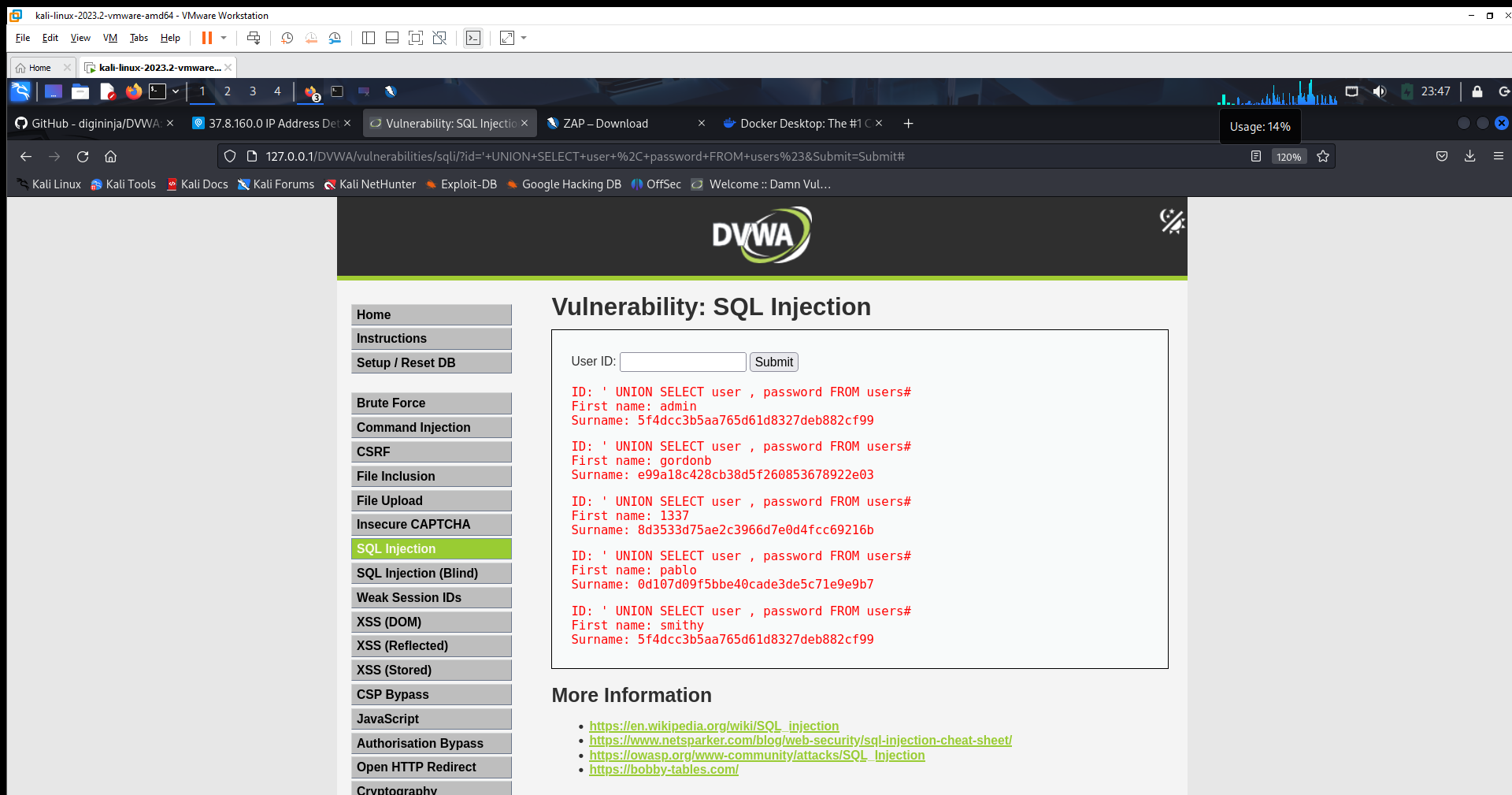
**Test 1: Basic Authentication Bypass**

* **Target**: 127.0.0.1/DVWA/vulnerabilities/sqli/
* **Payload**: ' OR '1'='1
* **Result**:
  + Returned **all 5 users** (first/last names)
  + Proof of Broken Authentication



**Test 2: *Database Schema Extraction***

* **Payload**:  ' UNION SELECT user, password FROM users #
* **Result**:
  + Exposed **usernames and MD5-hashed passwords as Surnames**



**Impact Analysis:**

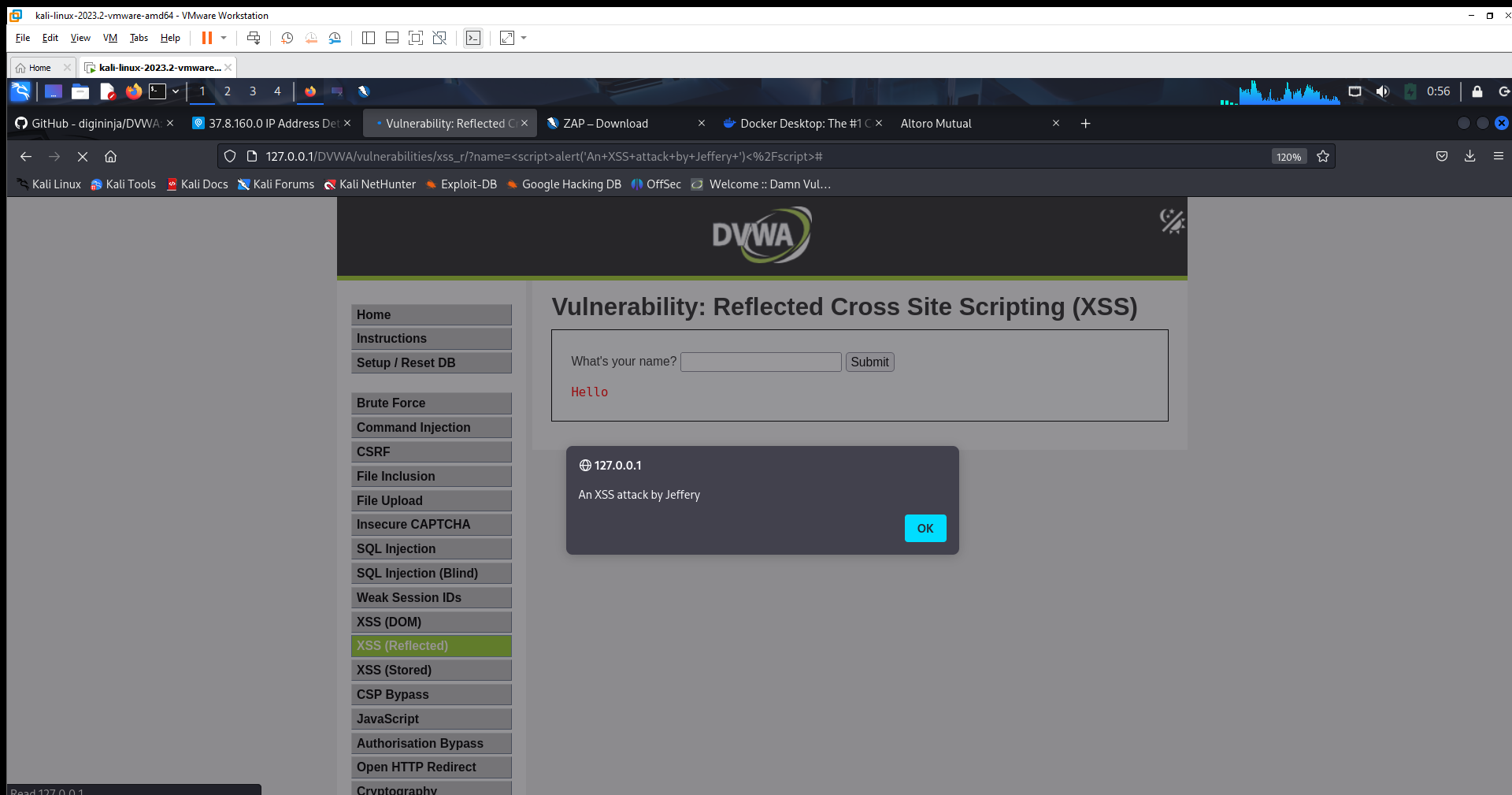
* Extracted all user credentials (5 records)
* Critical data leakage risk
* Bypassed authentication mechanisms
* Potential for complete database compromise

***■ Cross-Site Scripting (REFLECTED XSS):***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Reflected XSS** | **XSS DOM (DOM Based Cross Site Scripting )** | **XSS STORED** |
| Link | http://127.0.0.1/vulnerabilities/xss\_d/?name=# | http://127.0.0.1/vulnerabilities/xss\_d/?default=<script>alert('English by Jeff DOM XSS')</script> | http://127.0.0.1/vulnerabilities/xss\_d/ |
| Payload Used | <script>alert('An XSS attack by Jeffery')</script> | <script>alert('English by Jeff DOM XSS')</script> | <script>alert('Hack by Jeff Stored XSS’)</script> |
| Impact | Attackers can steal cookies or redirect users. |  | All users viewing the page get attacked. |

**PROOF:**

**Reflected XSS**



**DOM XSS**

**STORE XSS**

## **Findings & Risk Assessment**

| **Vulnerability** | **Risk** | **Proof** |
| --- | --- | --- |
| SQL Injection (Union-based) | Critical | Extracted user credentials |
| Reflected XSS | High | Executed alert('XSS') |
| Stored XSS | High | Persistent script in guestbook |

## **Recommendations**

**Technical Fixes**

SQLi Prevention:

Implement parameterized queries:

$stmt = $conn->prepare("SELECT \* FROM users WHERE id = ?");

$stmt->bind\_param("i", $input);

XSS Mitigation:

Implement htmlspecialchars():

htmlspecialchars($input, ENT\_QUOTES, 'UTF-8')

echo htmlspecialchars($user\_input, ENT\_QUOTES, 'UTF-8')

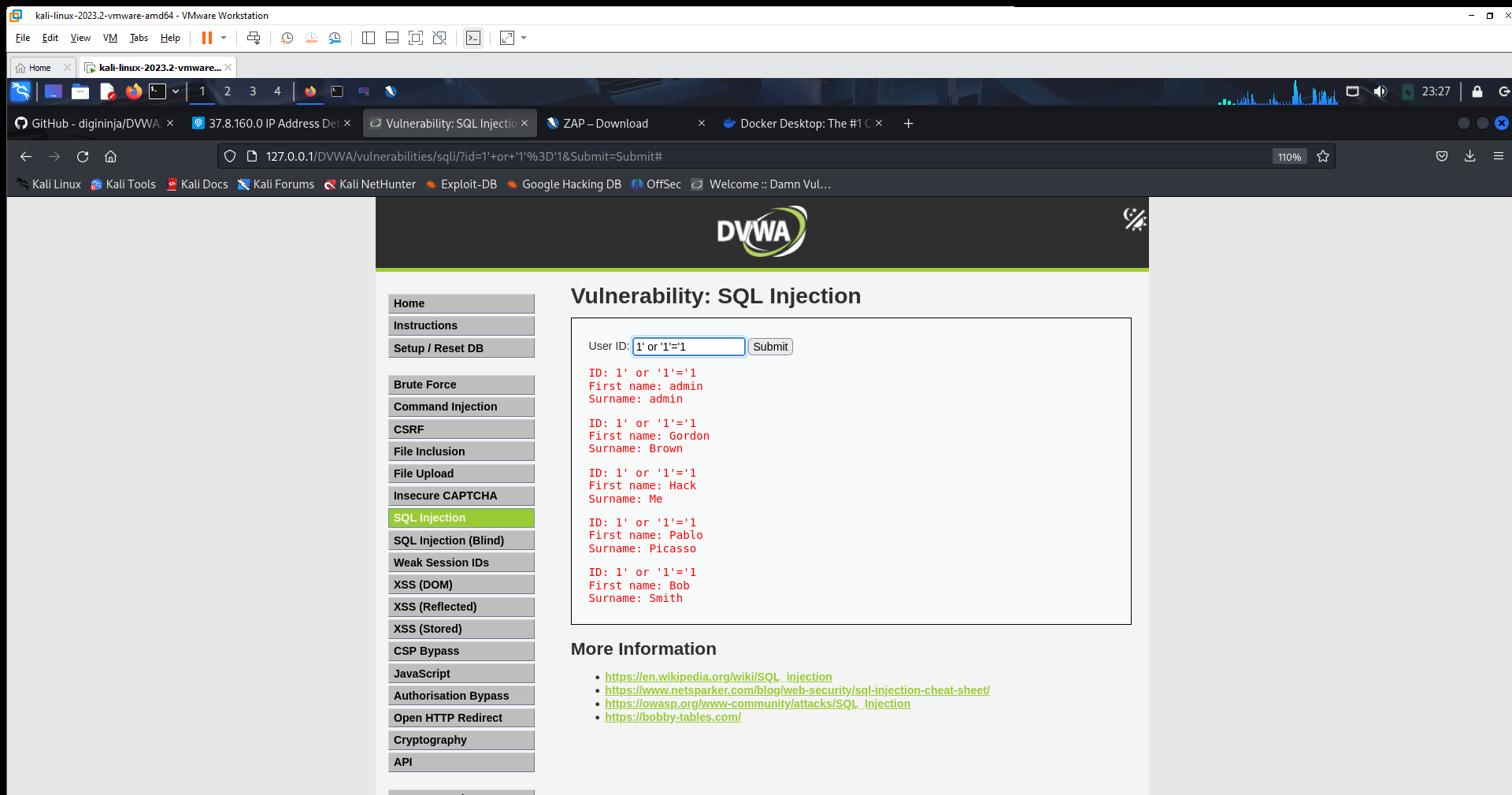
Add Content Security Policy (CSP) headers.

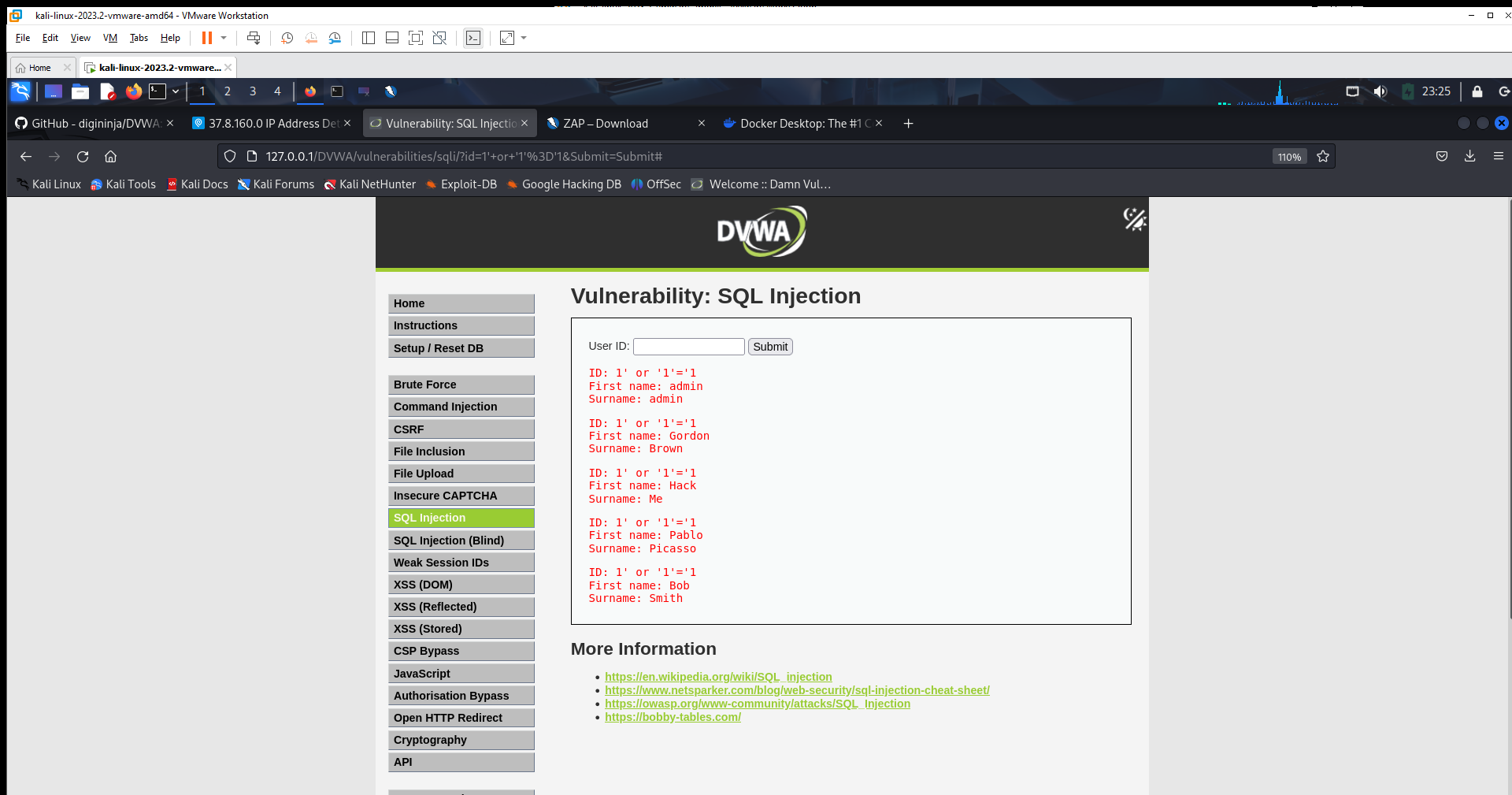
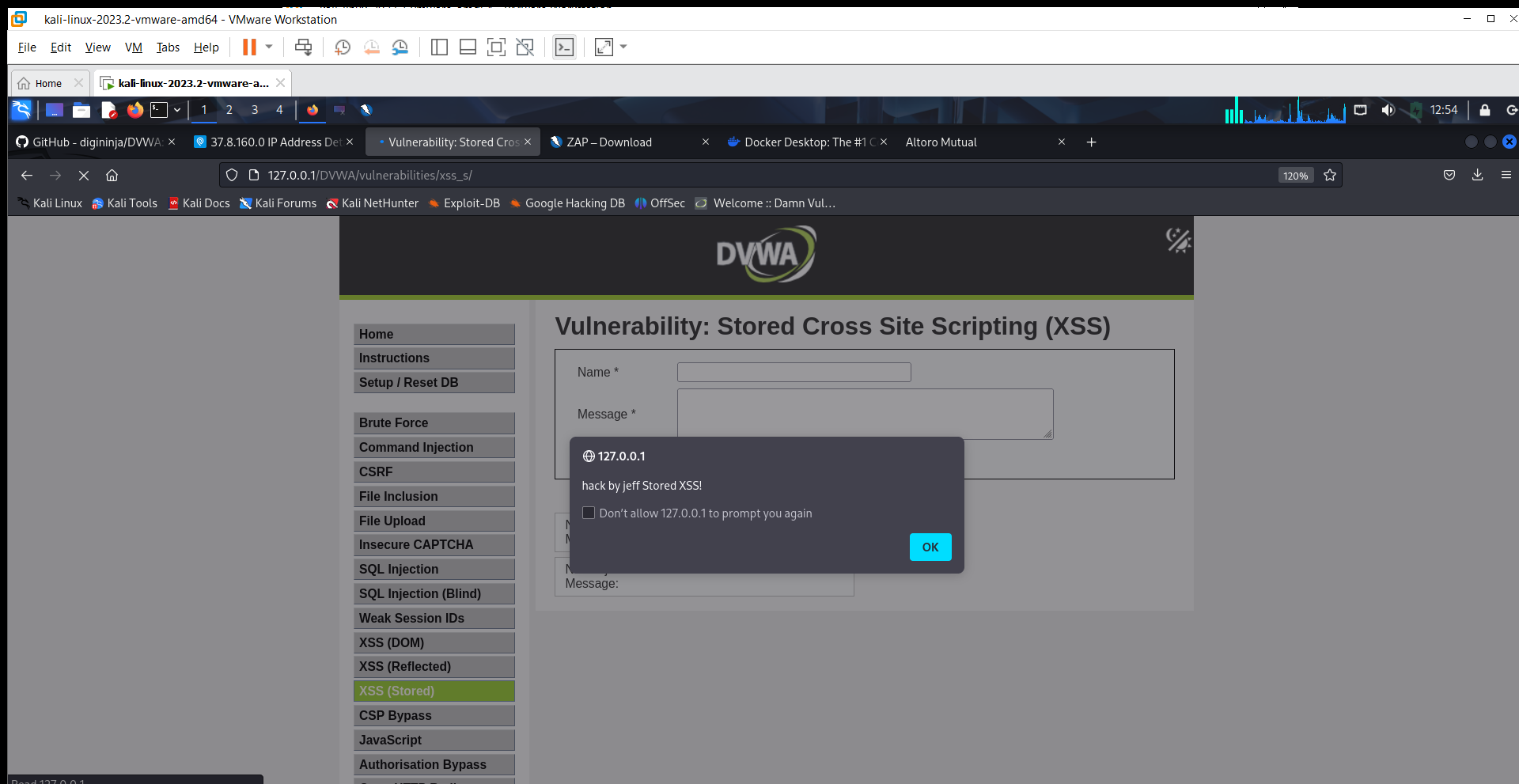
## **Conclusion**

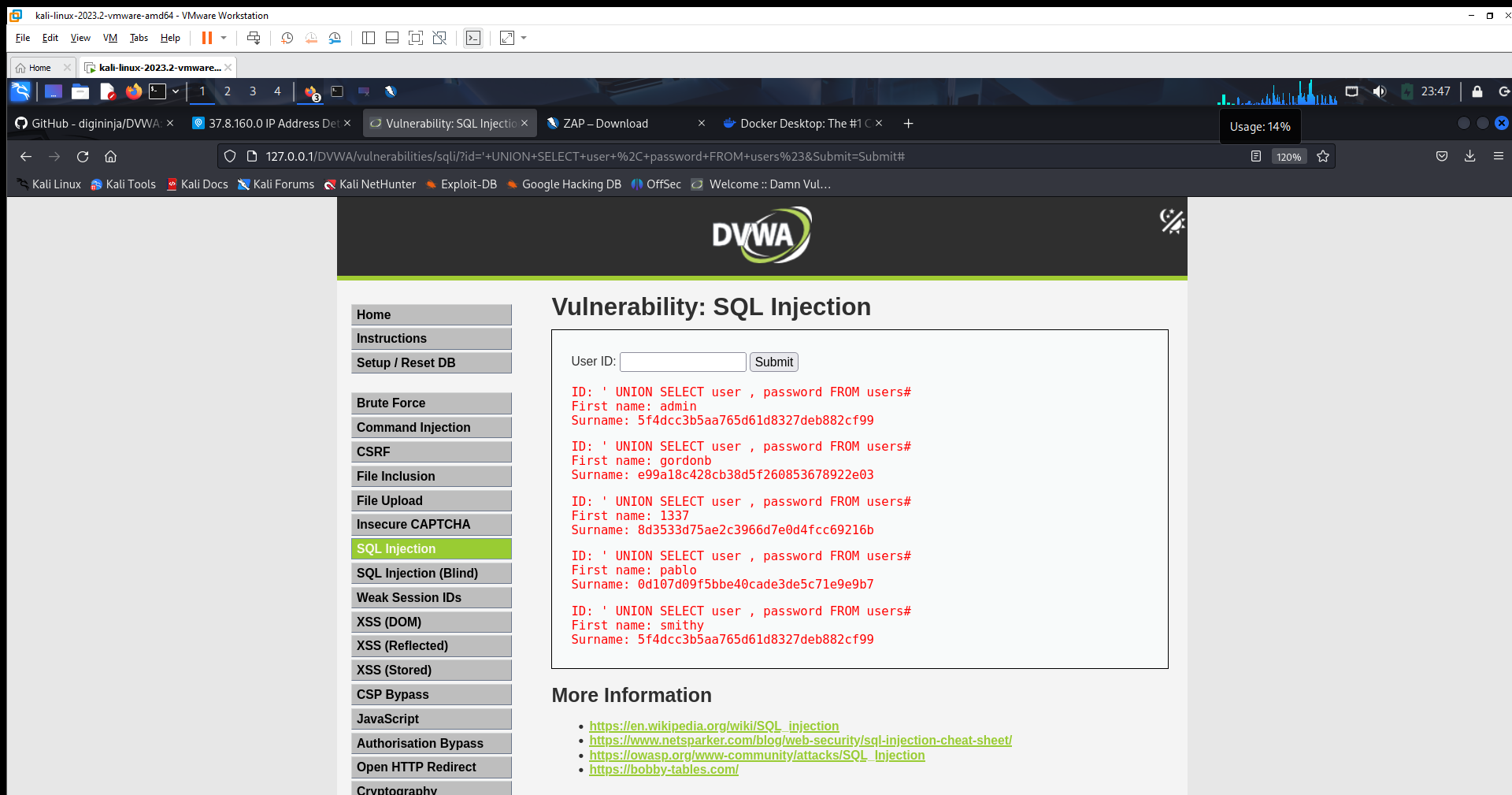
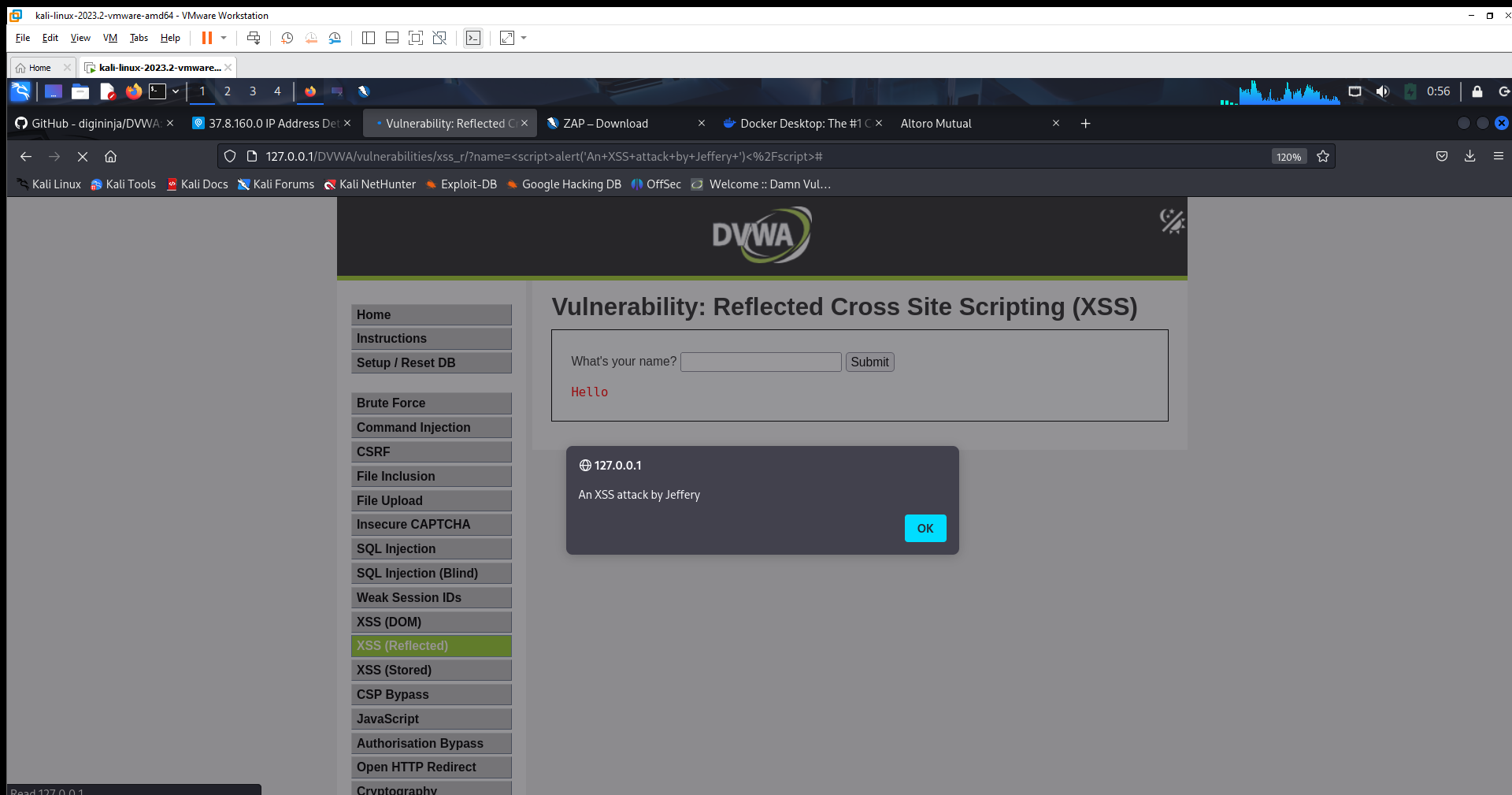
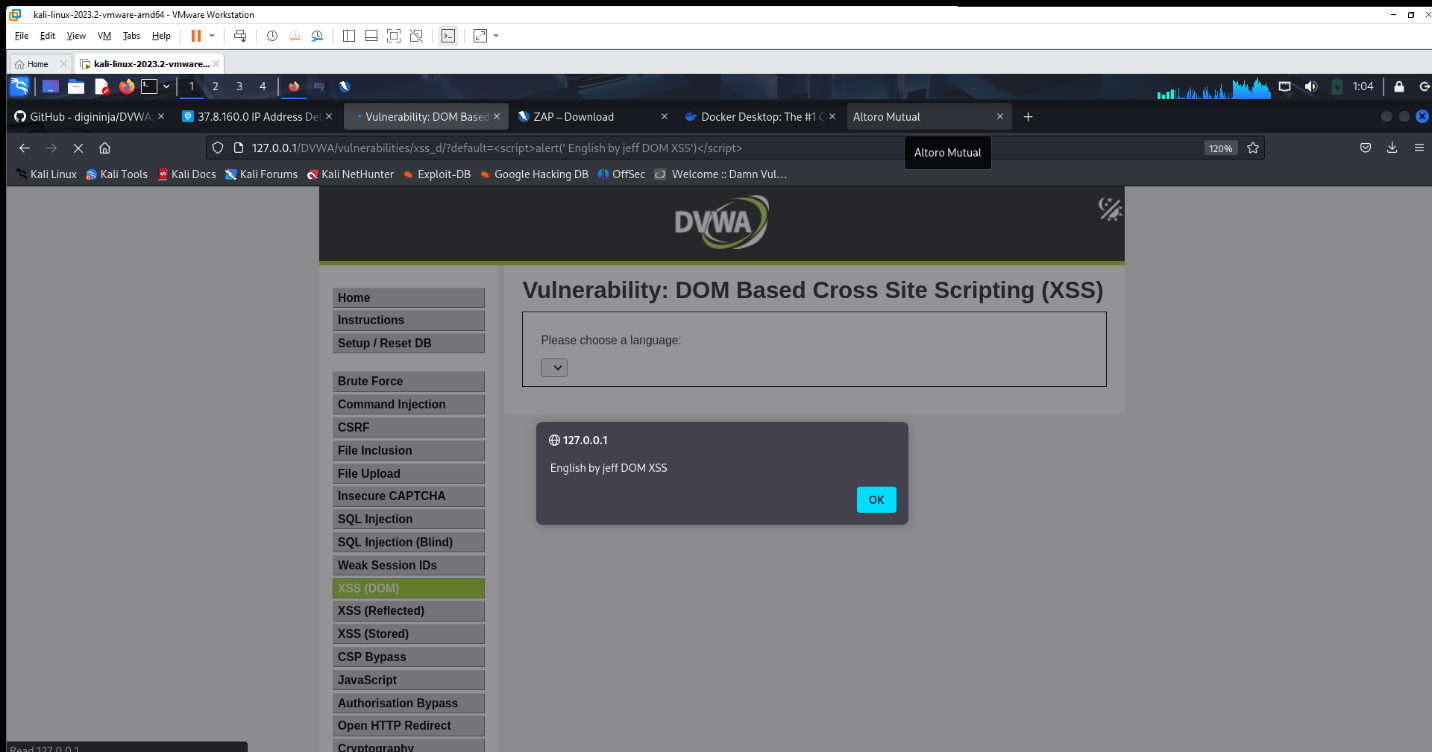
The web application contains multiple critical security vulnerabilities, including SQL Injection, XSS, authentication flaws, and broken access control. Immediate remediation is necessary to enhance security and protect user data. Implementing secure coding practices, proper input validation, and strong authentication mechanisms will mitigate these risks effectively.

NB: -Test conducted on DVWA (Damn Vulnerable Web App)

-No real users were affected

Appendix A: Screenshots





Appendix B: Payloads Used

• SQLi:

' OR '1'='1

' UNION SELECT user, password FROM users #

• XSS:

<script>alert('An XSS attack by Jeffery')</script>

<script>alert('Hack by Jeff Stored XSS’)</script>

<script>alert('English by Jeff DOM XSS')</script>