Project Title: SQL Injection Scanner
Name: S. Krishna Kumar
Platform / Context: Cybersecurity / Web Application Security Testing (Authorized Test Environments – DVWA, Localhost)

1. Overview

This project is a basic SQL Injection (SQLi) scanner designed for educational and ethical security testing. This document provides a comprehensive guide to a Python-based SQL Injection (SQLi) Scanner script. The script is designed for educational and ethical penetration testing purposes only. It probes web inputs for common SQL injection vulnerabilities by sending crafted HTTP requests and analyzing responses for indicators of vulnerability (e.g., SQL error messages or unexpected data dumps).

It probes HTTP GET parameters for common SQL injection patterns and reports potential vulnerabilities based on database error messages in server responses.

The scanner is intended only for authorized targets, such as:

- DVWA (Damn Vulnerable Web Application)
- Local test applications

2. Objectives

The main objectives of this project are:

- Identify potential SQL injection points in web applications
- Automate testing using common SQLi payloads
- Detect error-based SQL injection indicators
- Log and report findings
- Demonstrate basic concurrency and rate limiting
- Follow legal and ethical security testing practices

3. Scope and Limitations

In Scope

- Error-based SQL injection detection
- GET parameter testing
- Local or intentionally vulnerable targets
- Basic concurrency using threads

Out of Scope

- Authentication bypass
- Blind SQL injection
- Time-based SQL injection
- Data extraction or exploitation
- POST parameter testing
- Security bypass techniques

This scanner does not exploit vulnerabilities; it only detects indicators.

4. Ethical and Legal Considerations

⚠ **Important**

- This tool must be used only on systems you own or have permission to test
- DVWA and local labs are explicitly allowed targets
- Unauthorized testing on real websites is illegal and unethical

The user is fully responsible for ensuring proper authorization. This tool must only be used on targets you own or have explicit permission to test, such as local applications, Damn Vulnerable Web Application (DVWA), or other intentionally vulnerable testing environments. Unauthorized use on production systems or third-party websites is illegal and unethical. Always comply with laws like the Computer Fraud and Abuse Act (CFAA) in the US or equivalent regulations elsewhere. The author and xAI disclaim any responsibility for misuse.

5. System Requirements

Operating System

- Kali Linux (recommended)
- Any Linux distribution with Python 3

Software

- Python 3.8+
- requests library

Installation

sudo apt update

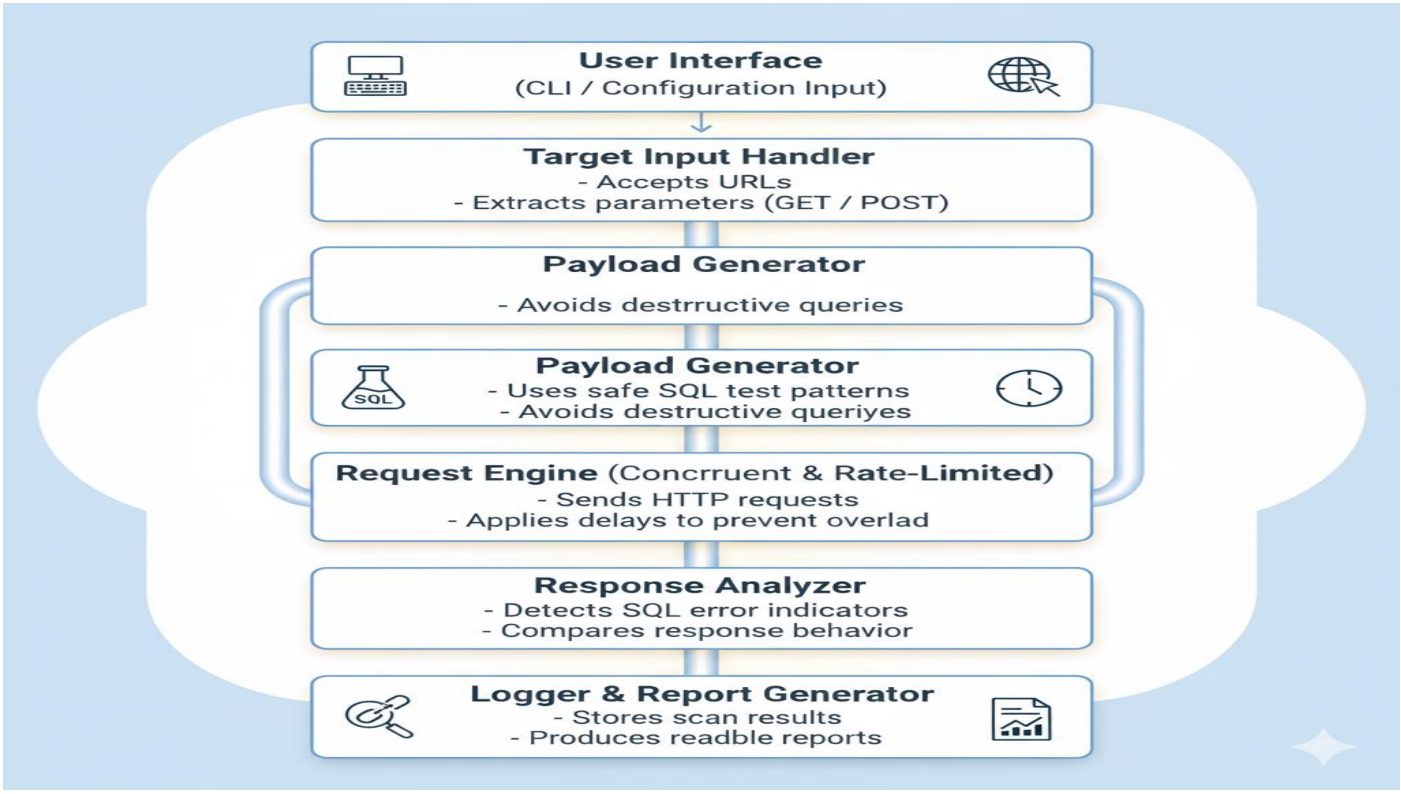
sudo apt install python3-requests

6. File Structure

sql_scanner.py # Main scanner script

payloads.txt # SQL injection payload list

results.log # Scan results and findings



7. Payload File (payloads.txt)

This file contains common SQL injection payloads, one per line.

Example:

'

"

' OR '1'='1

" OR "1"="1

--

#

Payloads are appended to existing GET parameter values during testing.

```
1 OR 1=1
1\' OR \'1\'=\'1
1\'1
1 EXEC XP_
1 AND 1=1
1\' AND 1=(SELECT COUNT(*) FROM tablenames); --
1 AND USER_NAME() = \'dbo\'
\\\'; DESC users; --
1\\\'1
1\' AND non_existant_table = \'1
```

8. How the Scanner Works

Step 1: Input Target URL

The user provides a URL containing **GET parameters**, for example:

http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit

Step 2: Parameter Enumeration

All GET parameters are extracted and tested individually.

Step 3: Payload Injection

Each payload is appended to the original parameter value and sent as a new HTTP request.

Step 4: Response Analysis

The scanner searches the response body for known SQL error patterns such as:

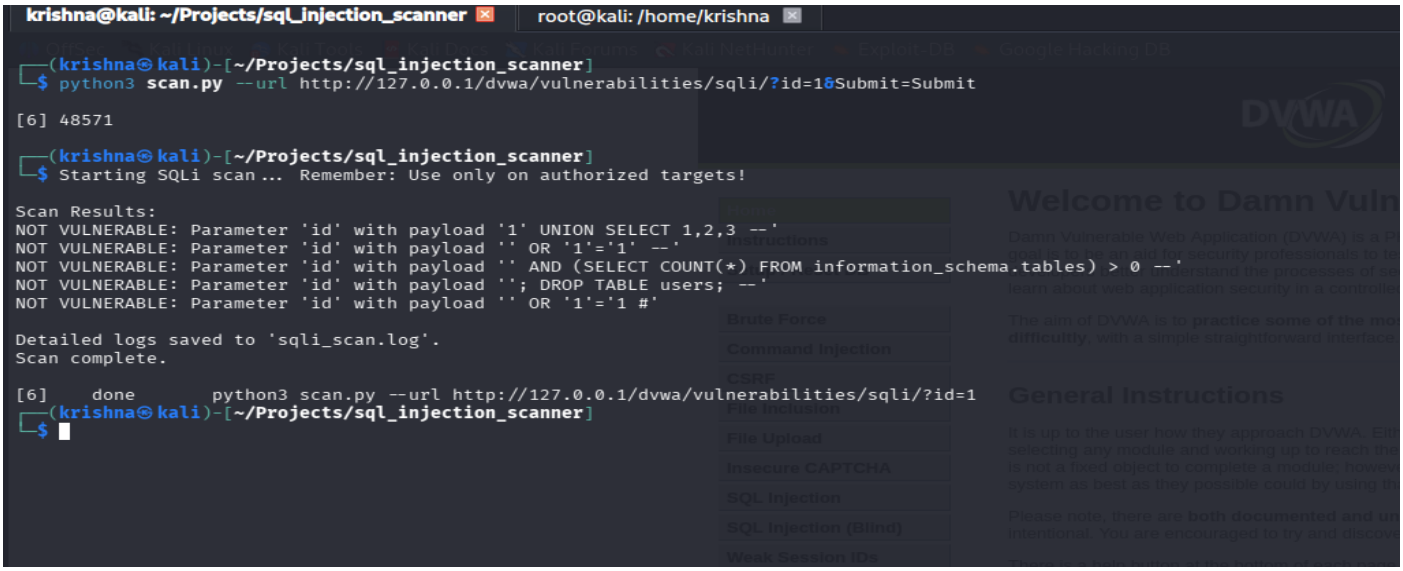
- SQL syntax errors
- MySQL warnings
- Database exception messages

Step 5: Reporting

If an error indicator is found:

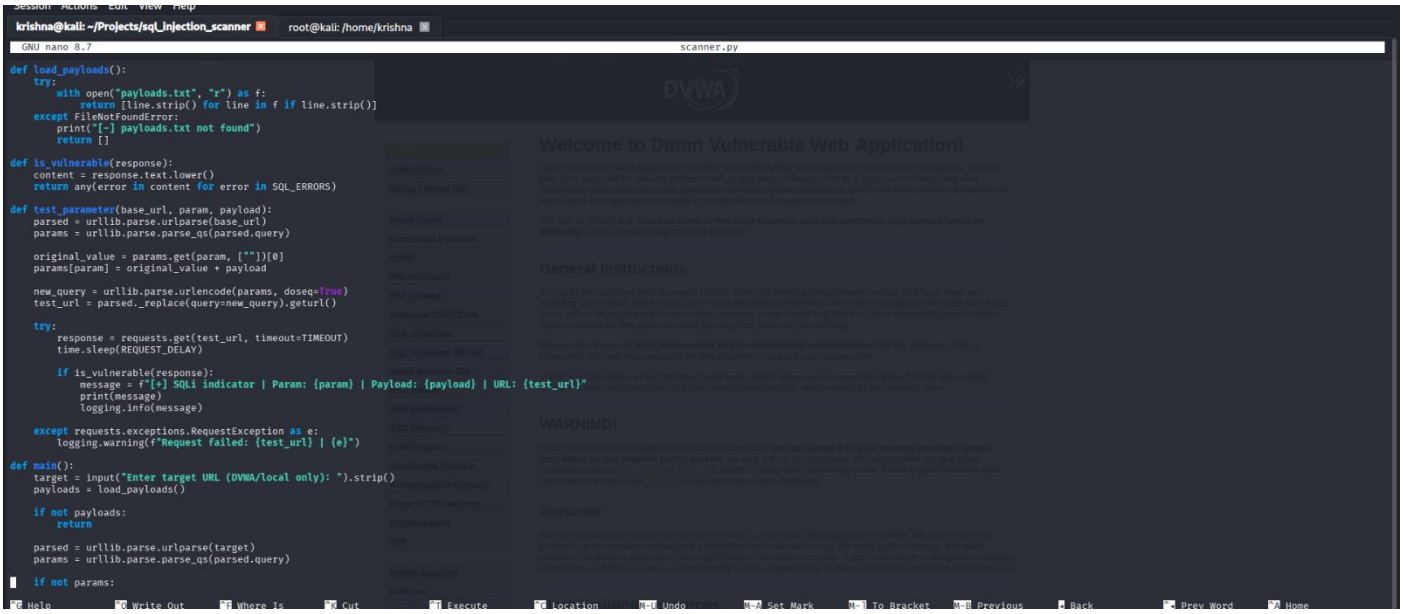
- The parameter name
- Payload used
- Full test URL

are printed to the console and logged to results.log.



9. Concurrency and Rate Limiting

- The scanner uses a ThreadPoolExecutor with a configurable number of worker threads
- A fixed delay (REQUEST_DELAY) is applied between requests to prevent flooding the server
- This ensures polite and controlled scanning behavior



10. Logging and Output

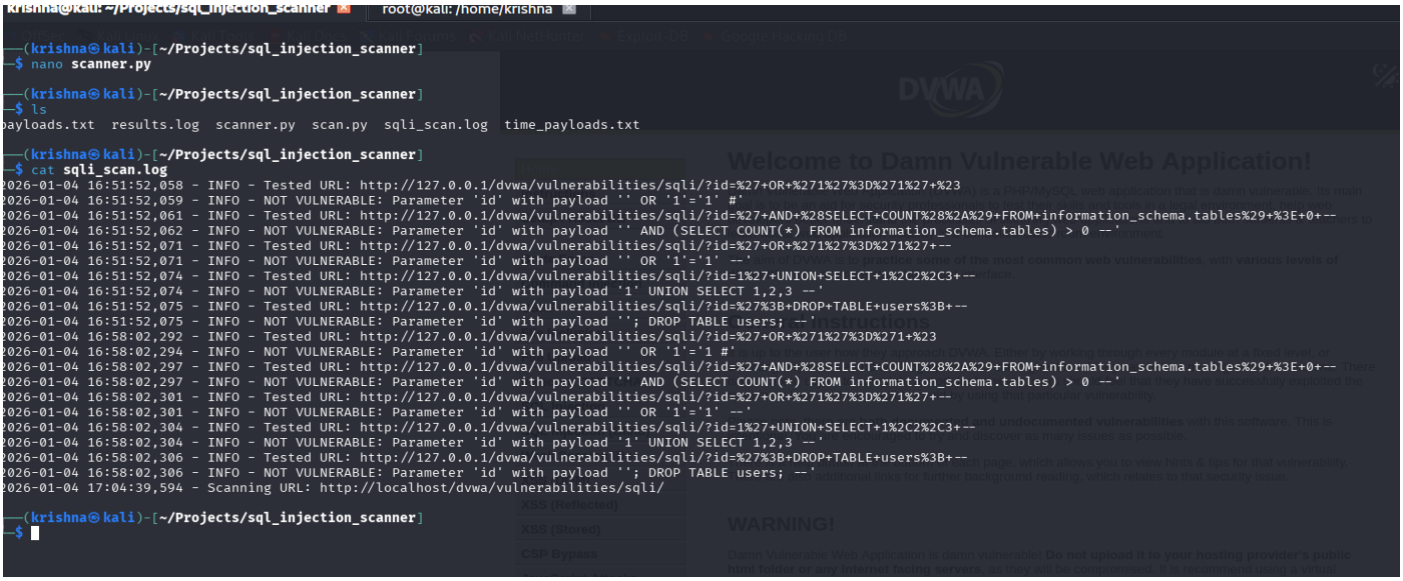
Console Output

[+] SQLi indicator | Param: id | Payload: ' | URL: http://...

Log File (results.log)

2026-01-04 12:30:45 - SQLi indicator | Param: id | Payload: ' | URL: ...

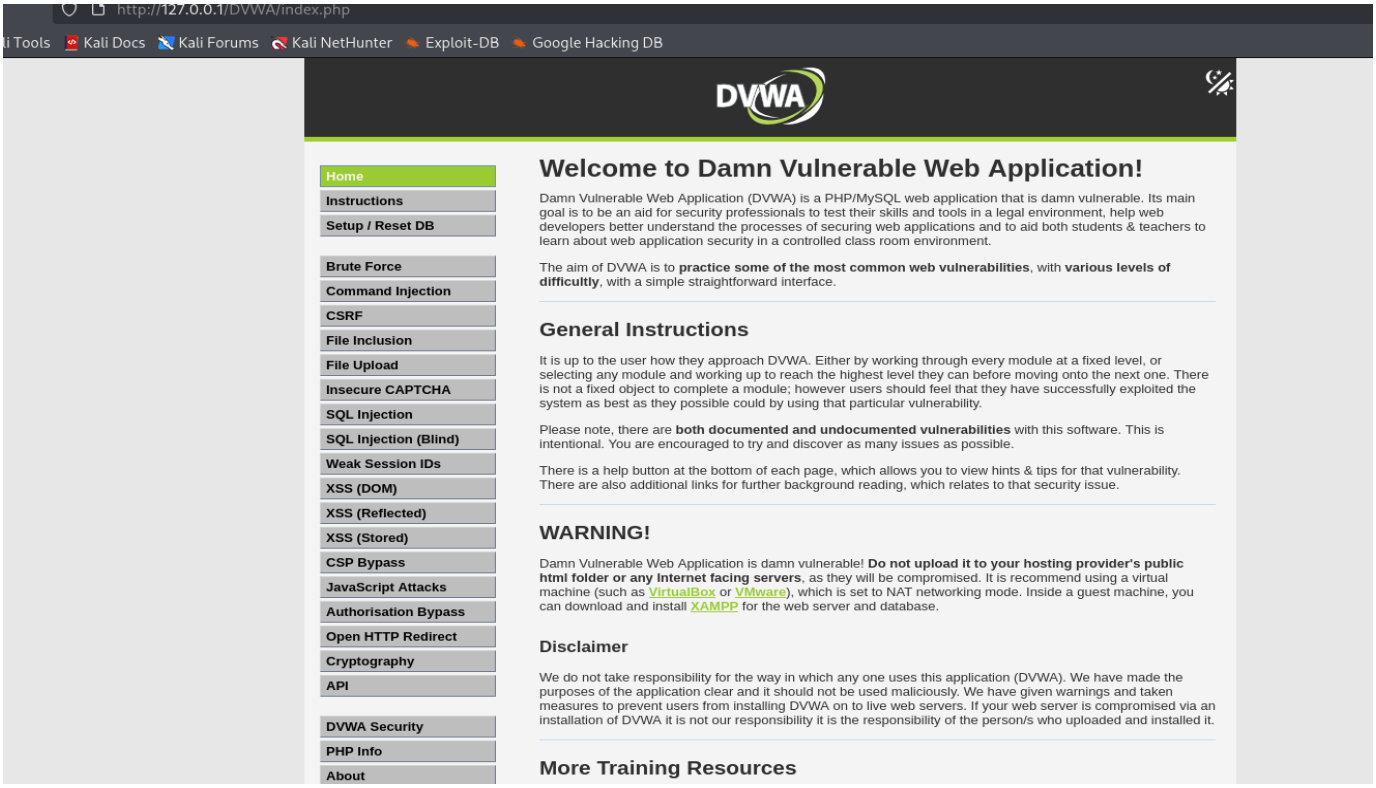
Logs provide traceability and reproducibility.



11. DVWA-Specific Notes

- DVWA requires user authentication
- The security level must be set to Low
- Without a valid session, the scanner may receive redirects to the login page
- This can result in false negatives

This behavior is expected and does not indicate scanner failure.



12. Known Limitations

- Only error-based SQL injection is detected
- URL encoding may affect payload behavior
- No session handling (cookies)
- No POST request testing

These limitations are intentional to keep the scanner simple and educational.

13. Conclusion

This SQL Injection Scanner demonstrates the fundamental concepts of automated vulnerability scanning while respecting ethical boundaries.

It is suitable for:

- Academic projects
- Introductory penetration testing labs
- Understanding how SQL injection detection works

It is **not a replacement** for professional security tools.