SQL Injection Playground with Detection Engine

Date: June 27, 2025

Submitted by: Your Name Here

Abstract

This project demonstrates a basic SQL Injection (SQLi) playground using a vulnerable Flask web application and a detection engine written in Python. The application is intentionally designed to be insecure in order to simulate SQL injection attacks and help understand how such vulnerabilities occur and how to detect and prevent them. It also includes a detector script to simulate various SQLi payloads and log suspicious behaviors like SQL errors and response delays.

Tools and Technologies Used

- Python 3
- Flask Framework
- SQLite Database
- Requests Library
- HTML Templates
- FPDF for PDF generation

System Overview

The system is composed of two main parts:

1. Vulnerable Flask Web App:

- A login form that uses raw SQL queries vulnerable to SQL injection.
- User credentials are validated using unparameterized SQL queries.

2. SQLi Detector (Python Script):

- Sends malicious SQL payloads to the login form.

- Detects SQL errors or long delays indicating vulnerability.

- Logs results to a file (sqli_log.txt).

How It Works

1. The Flask app runs locally at http://127.0.0.1:5000/.

2. Users enter username and password in the login form.

3. If the form uses vulnerable SQL, SQLi payloads can bypass authentication.

4. The detector script tests multiple payloads and logs if the app is vulnerable.

Sample Output and Logs

Example Log Entries:

[SQLi Detected] Payload: ' OR '1'='1

[TIME-BASED SQLi] Payload: ' OR sleep(5)-- | Delay: 5.02s

Conclusion

This project effectively shows how SQL Injection vulnerabilities work and how to detect them using automated tools. It also emphasizes the importance of secure coding practices such as using parameterized queries to defend against SQLi attacks.