SQL Injection Playground with Detection Engine

Abstract

This project focuses on developing a vulnerable web application integrated with a real-time detection engine to demonstrate and mitigate SQL Injection (SQLi) attacks. The playground serves as a controlled environment for security learners to understand how SQLi exploits work, how they can be detected, and what defense strategies can be implemented. By combining an intentionally flawed login/search module with a Python-based detection script, the project provides both offensive and defensive insights into SQLi vulnerabilities.

Introduction

SQL Injection remains one of the most critical security risks for web applications. Attackers manipulate improperly sanitized inputs to alter SQL queries, compromising confidentiality, integrity, and availability of data. This project aims to create a safe, educational framework where SQLi can be practiced, detected, and mitigated. Through the platform, users can:

- Experiment with vulnerable input fields.
- Analyze server responses to crafted injection attempts.
- Detect malicious SQLi patterns using a custom Python engine.
- Observe recommended defenses such as parameterized queries.

The project not only simulates real-world attack vectors but also highlights how secure coding prevents exploitation.

Tools Used

- Flask/PHP – For building the vulnerable web application.
- SQLite – Lightweight database to process queries.
- Python – For creating the SQLi detection and logging engine.
- Logging libraries – To record attempts, errors, and responses.
- Browser/Postman – For testing user inputs and injection scenarios.

Steps Involved

1. Designing the Vulnerable Application
    - Developed a simple login and search module connected to SQLite.
    - Intentionally left user input unfiltered, allowing direct query execution.
2. Simulating SQL Injection Attacks

- Crafted payloads like `' OR '1'='1` to bypass authentication.
- Observed abnormal database responses and query manipulation.

3. Building the Detection Engine
   - Implemented a Python script that sends crafted queries.
   - Detected anomalies such as error messages, time delays, or abnormal outputs.
   - Logged detailed records of successful and failed injections.
4. Integrating Logging and Analysis
   - Stored injection attempts in log files with timestamp, payload, and response.
   - Created simple analytics to show most common SQLi vectors.
5. Implementing Preventive Measures
   - Showcased secure coding practices with parameterized queries.
   - Compared results of insecure vs. secure input handling.
6. Educational Integration
   - Packaged the app and detection engine into a mini learning environment.
   - Provided examples of "attack vs. defense" to strengthen conceptual understanding.

Conclusion

The SQL Injection Playground with Detection Engine provides a comprehensive learning resource where students and professionals can explore web application vulnerabilities in a hands-on manner. It successfully demonstrates the lifecycle of an SQLi attack—exploitation, detection, and prevention. By integrating detection scripts and secure coding examples, the project reinforces the critical principle that proactive defense measures like input validation and query parameterization are the most effective ways to secure applications.