Building a Web Vulnerability Scanner: SQL Injection & XSS Detection

Team Member(s): Aniket Bansod
Hrushikesh Sawant
Himanshu Wankhade
Wrushali Patil

Guide: Riddhi Mirajkar

Course: Ethical Hacking and Network Security

The Growing Threat: Problem Statement

Web applications are critical to modern operations, yet they remain primary targets for cyberattacks. SQL Injection (SQLi) and Cross-Site Scripting (XSS) stand out as pervasive and highly impactful vulnerabilities.

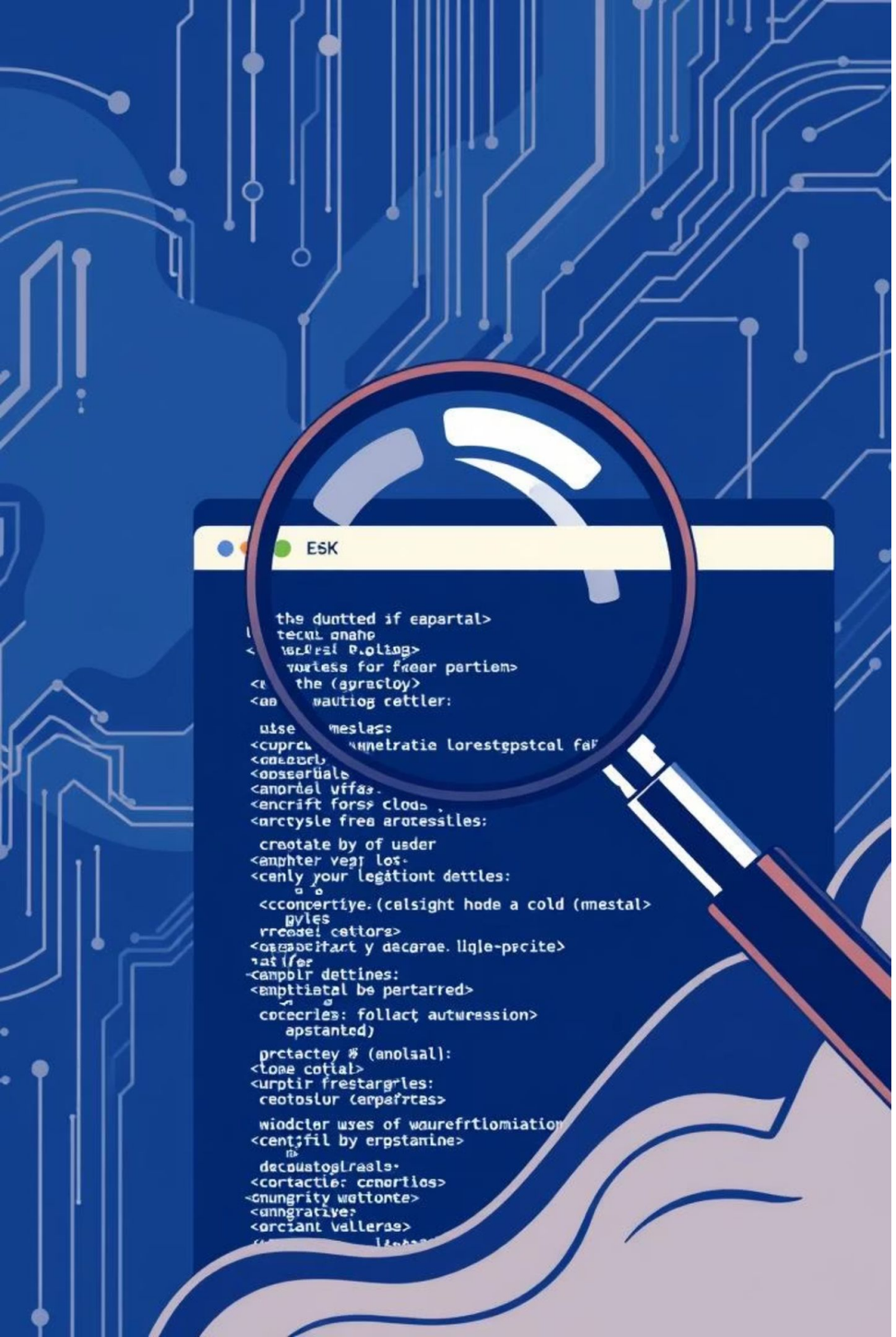
SQL Injection (SQLi)

Allows attackers to interfere with database queries, gaining unauthorized access, altering data, or even executing commands on the server. The implications range from data breaches to complete system compromise.

Cross-Site Scripting (XSS)

Enables attackers to inject malicious client-side scripts into web pages viewed by other users. This can lead to session hijacking, defacement, phishing, or malware distribution, severely compromising user trust.

Many small and medium-sized organizations lack the resources or expertise to implement robust, continuous security assessments, leaving them vulnerable to these common exploits.



Project Objectives: Empowering Web Security

Our primary objective is to develop an accessible and effective web vulnerability scanner tailored for SQL Injection and XSS. This tool aims to democratize basic web security testing, making it available to a broader audience.

1

Automated Detection

Implement algorithms to automatically identify common patterns and indicators of SQLi and XSS vulnerabilities.

2

Simplicity & Usability

Design an intuitive interface suitable for cybersecurity beginners, academic researchers, and small development teams.

3

Foundational Research

Provide a stable and extensible codebase for further academic research into automated vulnerability analysis.

Literature Survey: Foundations and Inspirations

Our project draws inspiration and methodology from established cybersecurity resources and cutting-edge research in automated vulnerability assessment.

OWASP Top 10

The OWASP Top 10 is a standard awareness document that lists the top 10 most critical web application security risks. It highlights Injection and Cross-Site Scripting (XSS) as persistent, high-risk vulnerabilities.

Relevance: Serves as the baseline for understanding and prioritizing web security threats like SQLi and XSS.

SQLMap & ZAP

Tools like **SQLMap** (automated SQL injection tool) and **OWASP ZAP** (Zed Attack Proxy) offer insights into sophisticated attack vectors and automated testing methodologies, informing our scanner's design.

Relevance: Helps in understanding how SQLi attacks are structured and automated. Inspired the scanning strategy for your project.

Academic Research

Recent papers from IEEE and IJERT, such as *"Automated Detection of SQL Injection Vulnerabilities using Machine Learning"* or *"XSS Vulnerability Detection using Hybrid Analysis"*, highlight advanced techniques and the feasibility of automated, intelligent scanning approaches.

Proposed System: Operational Flow

The scanner operates through a structured pipeline, transforming a target URL into a comprehensive vulnerability report.

1. URL Input & Pre-processing

Users provide the target web application URL. The system then parses this URL, identifies potential injection points (parameters, headers), and prepares HTTP requests.

2. Request Generation & Injection

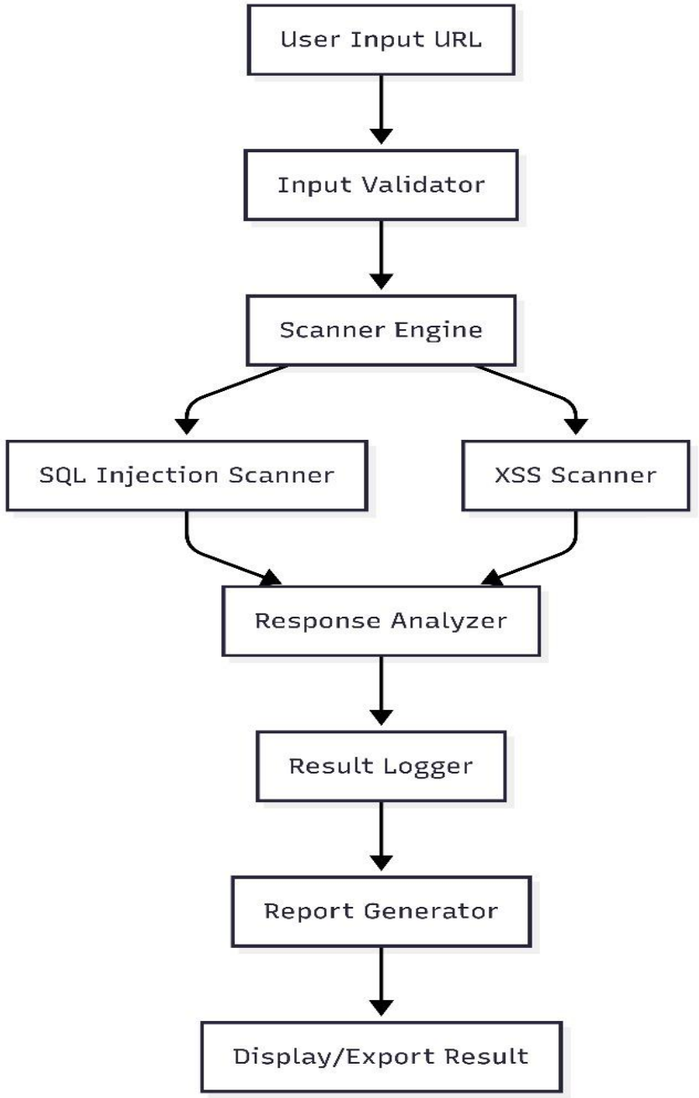
The scanner generates malicious payloads for SQLi and XSS, injecting them into identified parameters and sending modified HTTP requests.

3. Response Analysis

The system analyzes the web server's responses, looking for specific error messages (SQLi) or execution of injected scripts (XSS).

4. Vulnerability Reporting

Identified vulnerabilities are logged with details like affected URL, payload, and type of vulnerability, culminating in a user-friendly report.



Tools & Technologies: The Development Stack

Our web vulnerability scanner will be built using a robust set of Python libraries and tested against intentionally vulnerable web applications to validate its effectiveness.

Core Technologies

Python: The primary programming language for its versatility, extensive library ecosystem, and ease of scripting network operations.
Requests: A powerful HTTP library for making web requests, handling sessions, and managing parameters crucial for payload delivery.
BeautifulSoup4: For parsing HTML and XML documents, enabling the extraction of forms, links, and other elements as potential injection points.
Colorama: To enhance command-line output with colored text, improving readability and highlighting critical information.

Test Environments

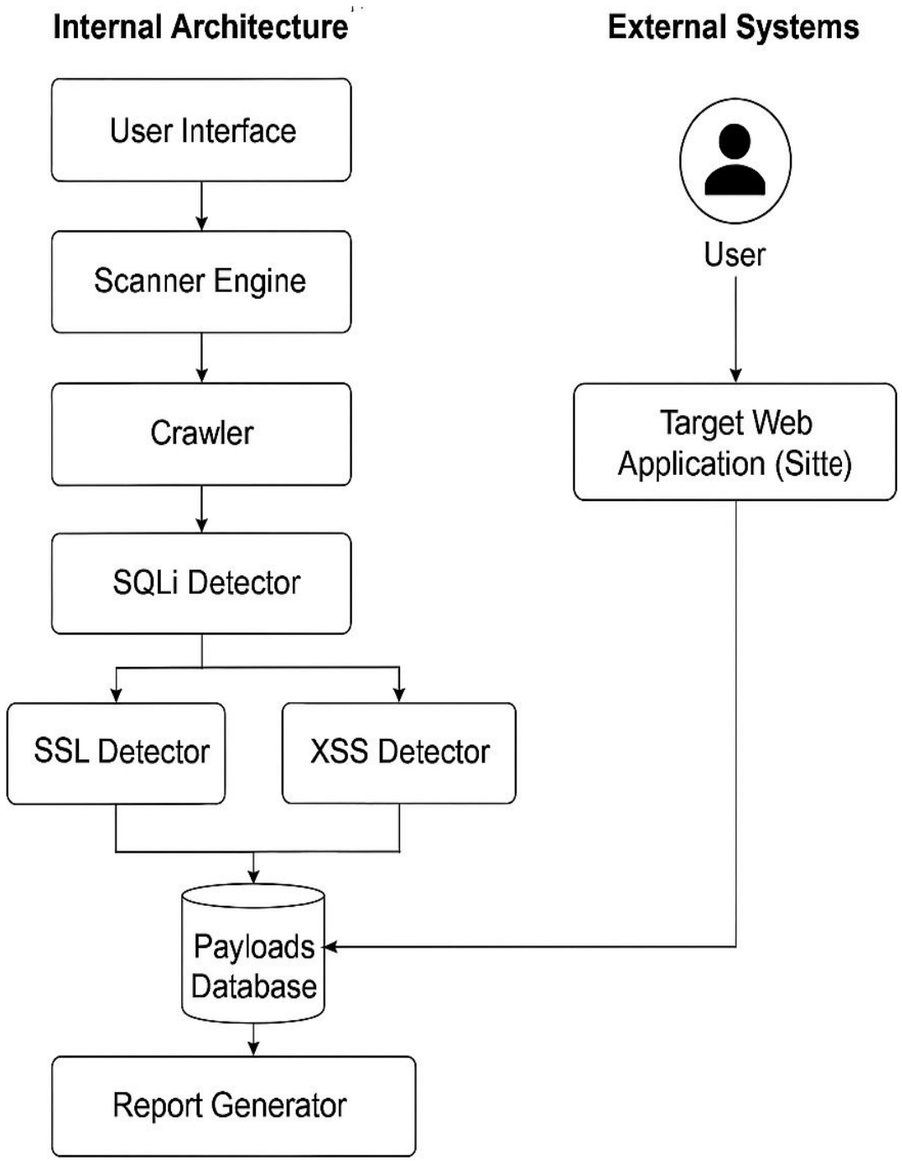
testphp.vulnweb.com: A deliberately vulnerable PHP web application, ideal for testing SQL Injection vulnerabilities.
xss-game.appspot.com: Google's XSS Game, providing a series of challenges to test and refine XSS detection capabilities.

Custom Local Environments: For controlled testing and development iterations, simulating various real-world scenarios.

System Architecture Diagram

External Components

- User Interface (CLI / Web Dashboard)
  - Allows user to input target URL(s) or payloads
  - Displays scanning results and vulnerability reports
- Target Web Application
  - The external system under test for vulnerabilities
  - Can be a live site or a locally hosted environment



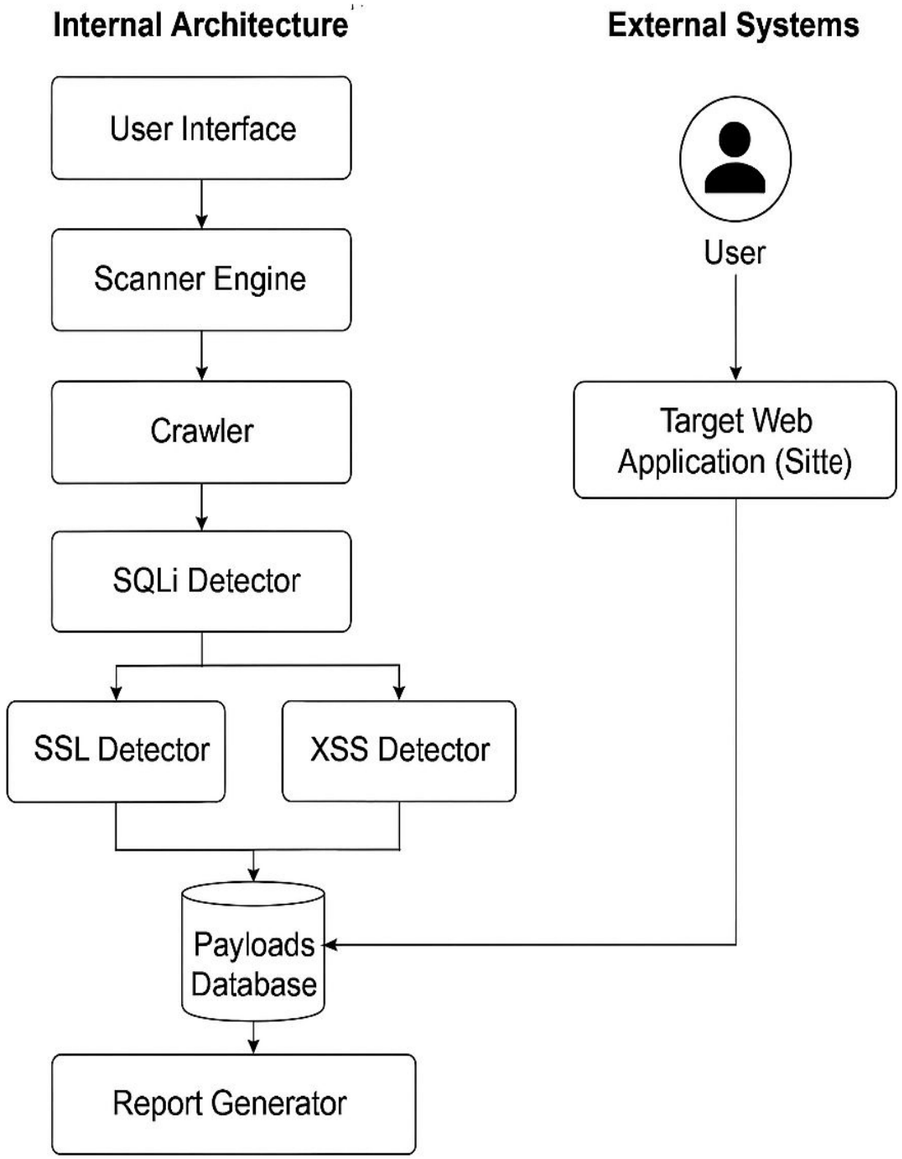
System Architecture Diagram

Internal Components

- **Crawler / URL Extractor**
  Automatically discovers internal links and parameters from the given domain

- **Payload Injector**
  Injects predefined SQLi and XSS payloads into identified inputs (GET/POST params)

- **Request Manager**
  Handles crafting and sending HTTP requests to the target
  Supports multiple request types and headers



System Architecture Diagram

- **Response Analyzer**

  Monitors HTTP responses for patterns indicating potential vulnerabilities

  Compares normal vs injected response behaviors

- **Signature & Heuristics Engine**

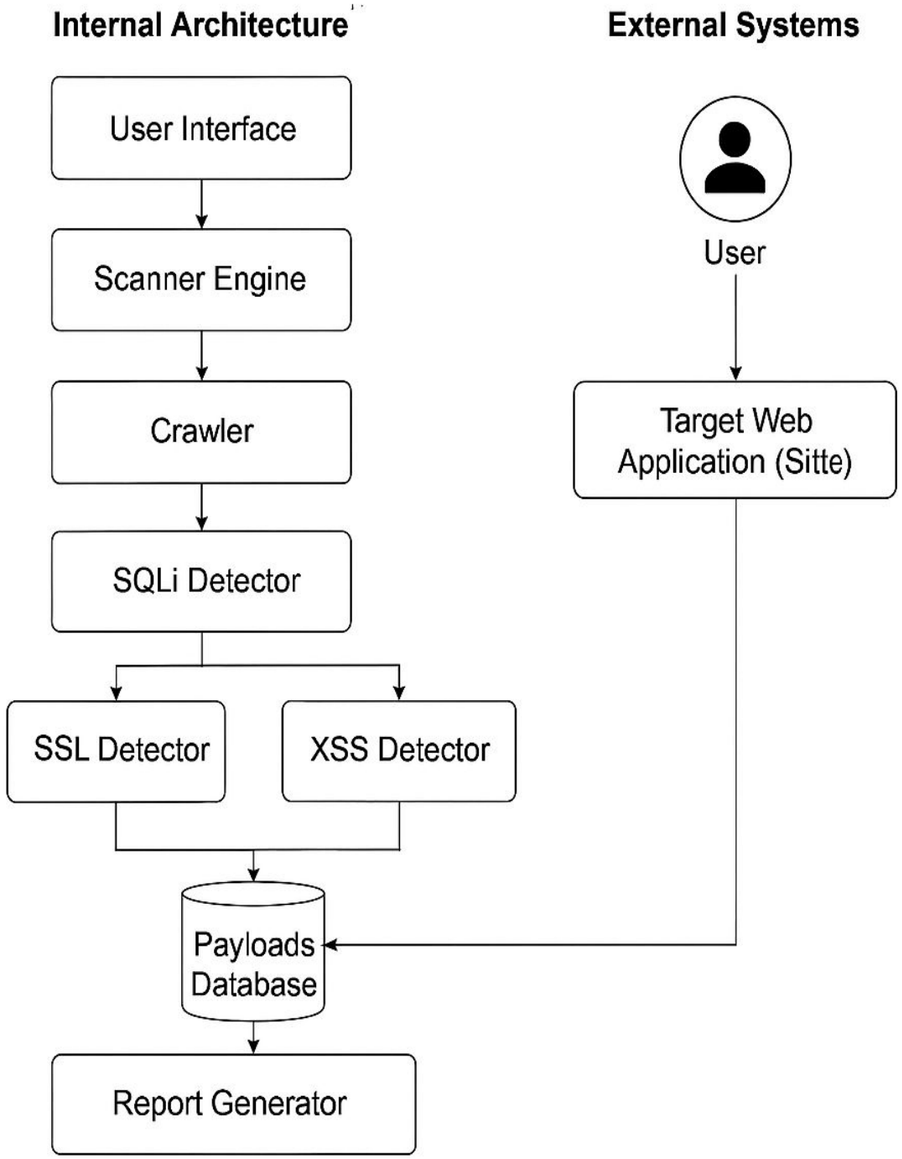
  Contains detection rules and logic for common SQLi/XSS patterns

  Uses both static signatures and dynamic anomaly detection

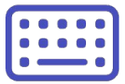
- **Logging & Reporting Module**

  Logs all actions, payloads used, responses, and detected issues

  Generates detailed vulnerability reports for each scan



Modules Overview: Core Functionalities



Input Handler

Manages user interaction, parses target URLs, and extracts initial data points for scanning. Validates and sanitizes input to prevent misuse.

XSS Detector

Deploys diverse XSS payloads (reflected, stored, DOM-based) and monitors for script execution within the browser context or server response indicators.

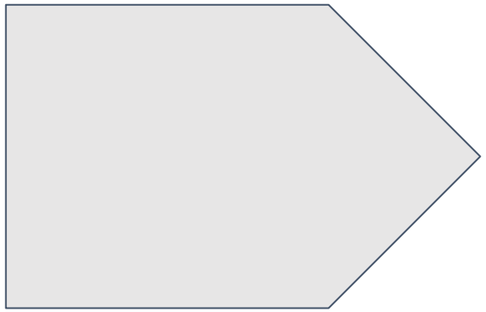
SQL Injection Detector

Crafts and injects various SQL payloads, analyzing database error messages, time-based responses, and boolean-based differences to confirm vulnerabilities.

Logger & Reporter

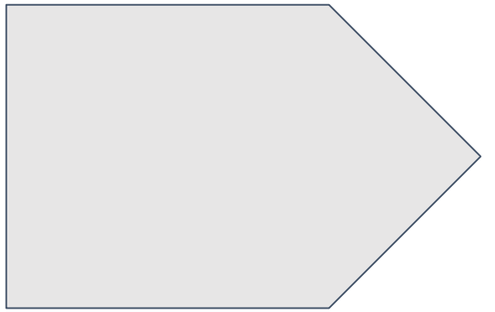
Records all scan activities, identified vulnerabilities, and generates a structured, human-readable report summarizing findings for the user.

Expected Outcomes & Impact



Functional Scanner

A robust, command-line based scanner capable of accurately identifying SQL Injection and XSS vulnerabilities on target web applications.



Detailed Reports

Detailed reports outlining detected vulnerabilities, including affected URLs, specific payloads that triggered the flaw, and remediation advice.

Future Scope: Enhancements & Evolution

In the future, we want to make our scanner smarter, easier to use, and able to catch more types of attacks.

Detect More Website Attacks

Better Design & User Experience

Use AI (Machine Learning) for Smarter Detection