Web Application Security Scanner Using Python

Amrita Vishwa Vidyapeetham - School of Computing, Chennai

Vaanusha, Sangamithra

Introduction

In the rapidly evolving digital era, web applications are frequent targets for cyberattacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Remote Code Execution (RCE), and Server-Side Template Injection (SSTI). Manual vulnerability assessments, while thorough, are time-consuming and not scalable for complex applications. To address this, the project introduces a Python-based Web Application Security Scanner that automates the detection of critical vulnerabilities using libraries like Requests, BeautifulSoup, and Regex. This lightweight tool is designed to be easy to use, capable of simulating real-world attack patterns, and generating detailed reports with actionable insights. It aims to support both novice and experienced users by offering a fast, efficient, and accessible solution for enhancing the security of web applications, making it particularly useful for small businesses and individual developers.

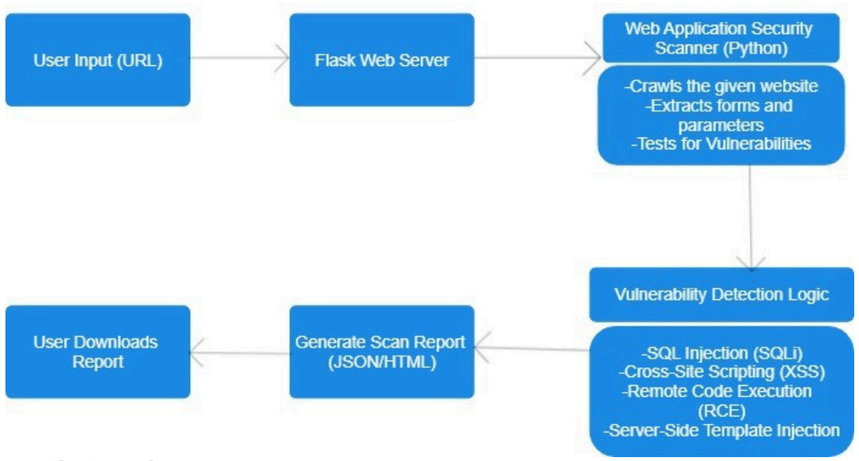
Problem statement

Web applications today handle sensitive data and are frequent targets for cyberattacks due to issues like poor input validation, misconfigurations, and outdated components. Traditional manual testing methods are time-consuming and require expertise, while many automated tools are either expensive or limited in scope. This project introduces a lightweight, Python-based web vulnerability scanner that detects common threats such as SQL Injection, XSS, and sensitive data exposure. It offers an affordable, easy-to-use solution for developers, small businesses, and security enthusiasts. Initially terminal-based, the tool will later be extended to a web-based version hosted on AWS for greater accessibility and usability.

Objective

The objective of this project is to develop a lightweight and effective web application security scanner using Python that can automatically detect common vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and sensitive data exposure. Given the increasing complexity of web technologies and rising cyber threats, the scanner aims to simplify and automate security testing, making it accessible to developers and small organizations without requiring deep expertise in penetration testing. The tool is designed to be user-friendly, extensible for future enhancements, and capable of improving the overall security posture of web applications. Initially built as a terminal-based tool, it has been extended to a web-based interface hosted on AWS for broader accessibility and ease of use.

Architecture Diagram



Tech Stack

- Frontend: Developed using HTML, CSS and FontAwesome icons to create a web interface where users can input target URLs and trigger the scan process.
- Backend & APIs: Python-based backend engineered to handle vulnerability scans through HTTP request handling, pattern matching, and multithreaded execution. Designed for command-line use and easily extendable to API deployment using frameworks like Flask or FastAPI and AWS for Cloud Deplyment.
- Detection Modules: Implements detection mechanisms for SQL Injection, Cross-Site Scripting (XSS), and Sensitive Data Exposure using crafted payloads and regex-based pattern extraction from web responses.
- Libraries & Tools: requests for sending HTTP requests, BeautifulSoup for DOM parsing and crawling, re for regex-based sensitive data scanning, urllib for URL parsing and handling, concurrent.futures for multithreaded scanning, and json for structured reporting.
- Execution Model: Optimized using multithreading to speed up scanning of multiple parameters or pages simultaneously, improving performance for larger web applications.

Results

The security scanning process across multiple web applications revealed several critical vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and Sensitive Information Exposure. These issues stem from common security flaws such as improper input validation, lack of output encoding, and insufficient access controls. SQL Injection allows attackers to manipulate backend databases, while XSS enables the injection of malicious scripts into web pages. Sensitive Information Exposure indicates that some applications unintentionally reveal system or user data. These findings highlight the importance of secure coding practices, regular vulnerability assessments, and the implementation of defensive mechanisms to enhance overall application security.

Real-Time UI Demonstration