Web Application Vulnerability Scanner

Name: Prathmesh Udgiri

Date: 28-08-2025

Introduction

Web applications are increasingly targeted by attackers exploiting vulnerabilities such as XSS, SQL Injection, and misconfigured CORS. This project implements a Python-based web application vulnerability scanner that automatically crawls web pages, identifies input fields, injects malicious payloads, and detects common vulnerabilities. The tool aims to help developers and security testers identify security weaknesses efficiently.

Abstract

The scanner is designed to detect major vulnerabilities outlined in the OWASP Top 10, including XSS, SQLi, Command Injection, Path Traversal, and CORS misconfigurations. It leverages Python libraries like requests and BeautifulSoup to crawl pages, extract forms, and submit payloads. Vulnerabilities are logged with evidence and severity, and a Flask-based web interface provides a user-friendly way to manage scans and view results in real time.

Tools Used

- Python 3 – main programming language
- Flask – web interface to manage scans
- Requests – HTTP requests for crawling and form submissions
- BeautifulSoup4 – HTML parsing and form extraction
- Regex – pattern matching for error-based vulnerability detection

Steps Involved in Building the Project

1. Gathered Information

- Researched how to build a web application vulnerability scanner.

2. **Created the Crawler**

   - Used Python requests and BeautifulSoup to crawl target web pages, extract all links and forms, and identify input fields.

3. **Built the Vulnerability Scanner**

   - Implemented tests for XSS, SQL Injection, Command Injection, Path Traversal, and CORS misconfigurations.
   - Responses were analyzed using regex patterns or payload reflection to detect vulnerabilities.

4. **Tested the Scanner**

   - Ran the Python scanner code on a dummy/testing website to verify payload injection and capture vulnerability output in the console.

5. **Created the Flask Web Interface**

   - Built a web application to input target URLs and display scanner results in the browser.
   - Connected the Flask app to the scanner's main function to show live scan outputs.

6. **Final Touches**

   - Added structured logging with payload, evidence, and severity for each vulnerability.
   - Conducted final tests to confirm results display correctly both in the console and on the web interface.

Conclusion

The project successfully implements a Python-based web application vulnerability scanner with a functional web interface. It detects XSS, SQLi, Command Injection, Path Traversal, and CORS misconfigurations, logging evidence and severity for each vulnerability. This tool can serve as a foundational framework for automated web security testing and is extensible for additional vulnerability types. The Flask interface enhances usability, making it accessible for testers and developers to perform security scans efficiently.