Web Application Vulnerability Scanner

Introduction

In the evolving world of cybersecurity, web applications are a prime target for attackers due to improper input validation and outdated code practices. This project focuses on developing a lightweight web vulnerability scanner that detects common vulnerabilities like Cross-Site Scripting (XSS) and SQL Injection (SQLi). The goal is to provide a proof-of-concept tool that demonstrates vulnerability detection using automated payload injections and response analysis.

Abstract

The Web Application Vulnerability Scanner is a Python-based tool designed to crawl websites, extract forms, inject test payloads, and identify vulnerabilities like XSS and SQLi. It also provides a web interface built with Flask to make scanning user-friendly. This project simulates real-world attack scenarios ethically and can serve as a learning tool for beginners and a foundational component in penetration testing pipelines.

Tools Used

- Python – Main scripting language
- BeautifulSoup – HTML parsing and DOM crawling
- Requests – For sending HTTP GET and POST requests
- Flask – For building the web-based UI
- Linux Terminal – For running the scanner and serving Flask app
- Text Files – Payload storage (xss.txt, sqli.txt)
- Custom Log File – vuln_log.txt to store scan results

Steps Involved in Building the Project

1. **Web Crawler Creation**
   Built a crawler to recursively visit all pages in the same domain and collect form inputs and URLs using BeautifulSoup.
2. **Payload Injection**
   Read test payloads for XSS and SQLi from external text files and injected them into form fields and URL parameters.

3. **Vulnerability Detection**
   Analyzed server responses for payload reflection (for XSS) or SQL error patterns (for SQLi) to identify vulnerabilities.
4. **Logging Mechanism**
   Created a log system to write timestamped vulnerability entries to vuln_log.txt and summarized them in scan_report.txt.
5. **Flask Web UI**
   Developed a web interface using Flask that allows users to input target URLs and view results in the browser.
6. **Reporting**
   Extracted and organized all scan data into a human-readable summary format, making it suitable for demonstration or auditing.

Conclusion

This project offered valuable hands-on experience in ethical hacking, web application security, and automation using Python. It mimics real-life vulnerability assessment workflows and highlights how attackers exploit basic input flaws. The tool, while simple, provides a solid foundation to build more advanced scanning systems and has improved my understanding of OWASP vulnerabilities, input sanitization, and secure coding practices.

Project Repository: https://github.com/lscdgetexit/webappscanner/tree/main
Submission Date: 25 June 2025
Submitted by: Kunal Jagdish Bhenwal