CYBERSECURITY INTERNSHIP PROJECT

Title:
Web Application Vulnerability Scanner using Python and Flask

Introduction:
With increasing threats in the web landscape, vulnerability scanners are essential tools in cybersecurity. This project aims to create a lightweight, browser-accessible web vulnerability scanner that detects two of the most common web application threats — **XSS (Cross-Site Scripting)** and **SQL Injection**. The tool allows users to input a URL, scans for forms, injects crafted payloads, and analyze the responses to detect potential vulnerabilities.

Abstract:
The Web Application Vulnerability Scanner is a Python-based tool with a Flask-powered web interface. It automates:

- Crawling target pages for forms
- Submitting crafted payloads to simulate attacks
- Analyzing the responses for indicators of security flaws

It provides a simple UI, logs the results in .txt format, and demonstrates the core scanning logic required in basic vulnerability assessment tools.

Tools Used:

- **Languages:** Python, HTML
- **Frameworks:** Flask, Bootstrap (CDN)
- **Libraries:** requests, BeautifulSoup, colorama
- **Others:** VS Code, Flask development server

CYBERSECURITY INTERNSHIP PROJECT

Steps Involved in Building the Project:

1. **Setup and Environment**
    - Installed Python and Flask in a virtual environment
    - Created folder structure: app.py, templates/, payloads.py, etc.
2. **Form Detection and Crawling**
    - Used BeautifulSoup to detect HTML forms from the target page
3. **Payload Injection**
    - Injected test XSS (<script>alert(1)</script>) and SQLi (' OR '1'='1) payloads into detected form fields
4. **Response Analysis**
    - Checked if the payload appeared in response (XSS) or caused SQL-related errors (SQLi)
5. **Web Interface**
    - Built a Flask web app with index.html to input URLs and results.html to display scan output
6. **UI Styling**
    - Integrated Bootstrap for a clean interface layout
7. **Scan Logging**
    - Implemented auto-saving of scan results to a .txt file inside the logs/ folder

Conclusion:

This project demonstrates the real-world implementation of an automated vulnerability scanner focused on identifying basic input-based threats. It combines both backend logic and frontend simplicity, providing a solid foundation for further enhancements like CSRF detection, database logging, or PDF exports.

This tool also reinforces key cybersecurity principles such as input validation, output sanitization, and secure coding practices.