Web Security Audit Scanner

A single-page web application that performs real-time, basic security scans on websites to identify common vulnerabilities based on the OWASP Top 10. The application features a modern, interactive frontend and a powerful Python Flask backend that serves as the scanning engine.

Features

- **Modern, Interactive UI:** A clean and responsive user interface for entering target URLs and viewing results.
- **Real-Time Scanning:** Communicates with a Python backend to perform live scans, not just simulations.
- **Comprehensive Vulnerability Checks:**
  - **SQL Injection (SQLi):** Intelligently discovers and tests all forms and URL parameters.
  - **Cross-Site Scripting (XSS):** Tests forms and URL parameters for reflected XSS vulnerabilities.
  - **Broken Authentication:** Checks for insecure login forms (e.g., submitting over HTTP).
  - **Security Misconfiguration:** Detects revealing server error messages.
  - **Cross-Site Request Forgery (CSRF):** Inspects forms for the absence of anti-CSRF tokens.
- **Dynamic Reporting:** Generates an interactive report with summary cards, charts for visual analysis, and a detailed, filterable table of findings.

Project Structure

For the application to work correctly, your files must be arranged in the following structure:

```
/web-scanner-project/
├── app.py              # The Python Flask backend server
├── requirements.txt    # Python dependencies needed for the backend
└── templates/
    └── index.html      # The HTML frontend user interface
```

How to Run the Application

Follow these steps carefully to set up and run the scanner on your local machine.

Step 1: Prerequisites

- Ensure you have **Python 3.7** or newer installed on your system. You can check by opening a terminal and running:
  python --version

Step 2: Set Up the Project Folder

- Create a folder named web-scanner-project.
- Inside it, create the app.py and requirements.txt files.
- Create a sub-folder named templates, and inside it, create the index.html file.
- Copy the code from the Canvas into the corresponding files.

Step 3: Navigate to the Project Directory

- Open your terminal (Command Prompt, PowerShell, or Terminal on Mac/Linux).
- Use the cd command to navigate into the project folder you created.
  cd path/to/your/web-scanner-project

Step 4: Create and Activate a Virtual Environment

This is a best practice that keeps your project's dependencies isolated.

- **Create the environment:**
  python -m venv venv

- **Activate the environment:**
  - On **Windows** (Command Prompt): venv\Scripts\activate
  - On **Windows** (PowerShell): .\venv\Scripts\Activate.ps1
  - On **macOS / Linux**: source venv/bin/activate

After activation, you should see (venv) at the beginning of your terminal prompt.

Step 5: Install Dependencies

- With your virtual environment active, install all the required Python libraries by running:
  pip install -r requirements.txt

Step 6: Run the Backend Server

- Execute the main application file:
  python app.py

- The terminal will show output indicating that the server is running, typically on http://127.0.0.1:5000.
- **Important:** Leave this terminal window open. The server must be running for the scanner to work.

Step 7: Use the Scanner

- Open your web browser (e.g., Chrome, Firefox).
- Navigate to the following address:
  http://127.0.0.1:5000

- The Web Security Audit Scanner interface will load. You can now enter a URL to scan (e.g., http://testphp.vulnweb.com) and receive a real-time report.

Disclaimer

This tool is intended for educational purposes and for testing applications with explicit permission. Unauthorized scanning of websites is illegal. The scanning logic is basic and may not find all vulnerabilities or may produce false positives.