Web Security Scanner

Overview

Web Security Scanner is a **Flask-based web application** that scans websites for security vulnerabilities. It helps identify **missing security protections, firewall presence, CAPTCHA detection, and common vulnerabilities like XSS and SQL Injection**. The scanner runs **asynchronously** for faster results and generates a **security report** for further analysis.

Features

- **Security Headers Analysis** – Checks for missing security headers like `Content-Security-Policy` and `X-Frame-Options`.
- **WAF Detection** – Identifies if a website has a **Web Application Firewall (WAF)**.
- **CAPTCHA Detection** – Determines whether CAPTCHA is implemented to prevent bot attacks.
- **Custom Payload Injection** – Tests for vulnerabilities like **XSS and SQL Injection**.
- **Asynchronous Scanning** – Uses `aiohttp` for fast and efficient scanning.
- **Real-time Results & Report Generation** – Displays scan results instantly and saves them in a downloadable **JSON report**.

Technologies Used

- **Python** – Core programming language.
- **Flask** – Web framework for building the application.
- **aiohttp & asyncio** – For asynchronous HTTP requests.
- **BeautifulSoup** – For parsing HTML responses.
- **Logging** – To track scan activity.
- **JSON** – Stores scan results in a structured format.

Installation & Setup

1. **Clone the Repository**

   ```
   git clone https://github.com/Wave-kun/web-security-scanner.git
   cd web-security-scanner
   ```

2. **Install Dependencies**

   ```
   pip install -r requirements.txt
   ```

3. **Run the Application**

   ```
   python Web_Security_Scanner.py
   ```

4. **Access the Web Interface** Open `http://127.0.0.1:5000/` in your browser.

Usage

1. Enter a website URL in the input field.
2. Click the "Scan" button.

3. View the results in real time.
4. Download the detailed security report.

File Structure

```
web-security-scanner/
|-- static/
|   ├── styles.css   # Frontend styling
|-- templates/
|   ├── index.html   # Web interface
|-- Web_Security_Scanner.py  # Main backend logic
|-- scanner.log  # Logs scan activities
|-- security_report.json  # Stores scan results
|-- README.md  # Project documentation
```

Code Explanation

Frontend (index.html & styles.css)

- `index.html` : Displays the user interface for entering a website URL and viewing scan results.
- `styles.css` : Provides a futuristic hacker-style theme for better usability.

Backend (Web_Security_Scanner.py)

Key Components:

- **Flask App Initialization:**

```
app = Flask(__name__, template_folder="templates", static_folder="static")
```

- **Security Headers Check:**

```
async def check_security_headers(url):
    headers_to_check = {"Content-Security-Policy": "Missing", "X-Frame-Options": "Missing"}
    async with aiohttp.ClientSession() as session:
        async with session.get(url) as response:
            for header in headers_to_check:
                if header in response.headers:
                    headers_to_check[header] = response.headers[header]
    return headers_to_check
```

- **WAF & CAPTCHA Detection:**

```
async def detect_waf(url):
    async with aiohttp.ClientSession() as session:
        async with session.get(url) as response:
            return "WAF Detected" if "Server" in response.headers else "No WAF detected."
```

- **Custom Payload Injection (Testing for XSS, SQL Injection):**

```
async def custom_payload_injection(url, payloads):
    async with aiohttp.ClientSession() as session:
```

```
        for param in ["id", "query", "search"]:
            for payload in payloads:
                test_url = f"{url}?{param}={payload}"
                async with session.get(test_url) as response:
                    if payload in await response.text():
                        return f"Possible vulnerability detected on {param}"
    return "No vulnerabilities detected."
```

- **Generating and Downloading Security Report:**

```
@app.route('/download_report')
def download_report():
    return send_file("security_report.json", as_attachment=True)
```

Future Improvements

- **AI-powered detection** to improve scanning accuracy.
- **Expanded payload testing** for more attack types.
- **Integration with a vulnerability database** for deeper insights.

License

This project is licensed under the MIT License.

Author

Ralph Zaw – GitHub