About My Program: Vulnerability Scanner

Overview:
My program, the Vulnerability Scanner, is designed to assess the security of websites by scanning for critical security headers, checking for open ports, evaluating SSL/TLS settings, and retrieving server information. The program leverages Python's built-in libraries such as `requests`, `socket`, `ssl`, and `BeautifulSoup` to carry out its scanning tasks. This tool is useful for security professionals, web developers, and anyone interested in evaluating the security configuration of publicly accessible websites.

Program Components:

My program consists of two Python files:
1. vuln_scanner.py - This file contains the core functionality of the vulnerability scanner. It includes functions to check security headers, scan ports, evaluate SSL settings, and retrieve server information.
2. test_vulnerability_scanner.py - This file contains test functions designed to ensure the correctness of the program. It uses pytest to verify that the program functions as expected.

Steps to Set Up and Run My Program:

1. Set Up a Virtual Environment (venv):

First, ensure that you have Python 3.6 or later installed on your system.
Create a virtual environment to isolate your dependencies:
```bash
python3 -m venv myenv
```

Activate the virtual environment:
- On Windows:
```bash
myenv\Scripts\activate
```

- On macOS/Linux:
```bash
source myenv/bin/activate
```

2. Install Required Dependencies:

With the virtual environment activated, install the required Python packages:
```bash
pip install requests beautifulsoup4 pytest
```

3. Download the Program Files:

Download the following Python files and place them in a folder:
- vuln_scanner.py: The main scanner program.
- test_vulnerability_scanner.py: The test file that includes pytest test functions.

4. Run the Program:

To run the vulnerability scanner, execute the vuln_scanner.py script:
```bash
python vuln_scanner.py
```

The program will prompt you to enter the URL of the website you wish to scan (e.g., `https://facebook.com`).

5. Run the Tests with pytest:

To ensure that the program functions correctly, you should run the test file using pytest. This will validate that all the functions in your program work as expected.
Run the following command to execute the tests:
```bash
pytest test_vulnerability_scanner.py
```

pytest will execute all the test functions and provide feedback on whether each test passes or fails.

6. Program Execution:

Once you input the website URL, the program will start performing the following scans:
- Security Header Check: Verifies if essential security headers like `Strict-Transport-Security`, `Content-Security-Policy`, and others are present.
- Port Scan: Scans for open ports on the website's domain (e.g., HTTP, HTTPS, SSH).
- SSL/TLS Evaluation: Checks the website's SSL/TLS certificate details.
- Server Information: Retrieves the server type and title of the website.

Example of the output:
```bash
{
    'headers': {'Strict-Transport-Security': 'max-age=31536000', ...},
    'open_ports': [443, 80],
    'ssl_info': {...},
    'server_info': {'server': 'nginx', 'title': 'Facebook'}
}
```

7. Analyze the Results:

The program will output a dictionary containing the results of the scan. Review the results to identify potential security vulnerabilities, such as missing headers, open ports, or issues with the SSL/TLS configuration.

8. Test Function Coverage:

The test_vulnerability_scanner.py file includes multiple test functions to verify the correctness of the scanner. Each test function is designed to test specific functionalities of the scanner, such as checking for security headers or performing port scans.
Each test function includes at least two calls and asserts to ensure that the functions behave as expected.

Program Output Example:

```bash
{
    'headers': {
        'Strict-Transport-Security': 'max-age=31536000',
        'Content-Security-Policy': 'default-src 'self';',
        'X-Frame-Options': 'DENY',
        'X-Content-Type-Options': 'nosniff',
        'Referrer-Policy': 'strict-origin-when-cross-origin'
    },
    'open_ports': [443, 80],
    'ssl_info': {
        'subject': '...',
        'issuer': '...',
        'valid_from': '...',
        'valid_to': '...'
    },
    'server_info': {'server': 'nginx', 'title': 'Facebook'}
}
```

Conclusion:
My Vulnerability Scanner is a useful tool for anyone looking to assess the security of a website. By verifying security headers, scanning open ports, checking SSL/TLS configurations, and retrieving server information, the program helps identify potential vulnerabilities. Running the program through pytest ensures that the scanner is working correctly, and the results can be analyzed to improve the security posture of the target website.