**Vulnerability Scanner Documentation**

Overview

The Vulnerability Scanner is a Python script designed to identify security vulnerabilities in web applications. The scanner currently supports two types of vulnerabilities: Cross-Site Scripting (XSS) and SQL Injection (SQLi).

Approach

**General Approach**

1. Form Detection: The script utilizes BeautifulSoup to parse HTML and locate all **<form>** elements on a given webpage.
2. Form Details Extraction: For each form, the script extracts relevant details such as action, method, and input fields.
3. Payload Injection: For XSS scanning, the user provides JavaScript code that is injected into text and search input fields of each form. For SQLi scanning, predefined SQL injection payloads are injected into various input fields.
4. HTTP Request Submission: The script sends HTTP requests to the target URL with the injected payloads.
5. Response Analysis: The responses are analyzed to identify potential vulnerabilities.

**XSS Scanning**

* The user inputs JavaScript code for XSS scanning.
* The script injects the provided JavaScript code into text and search input fields of each form.
* If the injected code is found in the response content, the script reports a potential XSS vulnerability.

**SQL Injection Scanning**

* Predefined SQL injection payloads are used to construct malicious input.
* The script injects these payloads into various input fields of each form.
* If common SQL error messages are found in the response content, the script reports a potential SQL injection vulnerability.

Code Structure

**Functions**

1. get\_forms(url):

Uses BeautifulSoup to find and return all <form> elements on the provided webpage.

1. form\_details(form):

Extracts details (action, method, inputs) from a given form element.

1. vulnerable(response, vulnerabilities):

Checks if a given HTTP response contains specific vulnerabilities.

1. scan\_xss(url, js\_code):

Performs XSS scanning by injecting user-provided JavaScript code into text and search input fields.

1. scan\_sqli(url):

Performs SQL injection scanning using predefined SQL injection payloads.

1. get\_form\_details(form):

Extracts form details specifically for XSS scanning.

1. submit\_form(form\_details, url, value):

Submits a form with a given payload.

**Main Execution**

* User inputs the target URL.
* User chooses the scan type (1 for XSS, 2 for SQLi).
* For XSS scanning, user inputs JavaScript code.
* The script calls the appropriate scanning function based on user choice.

Key Decisions

1. User Input for XSS Code:

To make the script versatile, the user is allowed to input the JavaScript code for XSS scanning. This allows customization based on specific testing requirements.

1. Predefined SQL Injection Payloads:

Common and effective SQL injection payloads are predefined to cover a range of scenarios. Users can extend this list if needed.

1. Dynamic HTTP Request Submission:

The script dynamically constructs and submits HTTP requests for each form, allowing flexibility in handling various web applications.

1. Clear User Interaction:

The script prompts the user for essential information, such as the target URL, scan type, and XSS code. This enhances user interaction and customization.

1. Structured Code with Comments:

The code is structured into functions with meaningful names, promoting readability. Comments are added to explain the purpose of each section.

Future Enhancements

1. Additional Payloads:

Add more predefined payloads for SQL injection to increase coverage.

1. Automated Payload Generation:

Implement a mechanism to automatically generate diverse payloads for both XSS and SQL injection testing.

1. Interactive Web Interface:

Develop a simple web-based interface for easier usability.

Conclusion

The Vulnerability Scanner script provides a foundation for identifying XSS and SQL injection vulnerabilities in web applications. Its modular structure, user-friendly interaction, and customizable aspects make it suitable for security testing in various scenarios. Further development and enhancements can be implemented to increase its capabilities and usability.