Web Application Vulnerability Scanner Report

Introduction

The Web Application Vulnerability Scanner is a Python-based tool designed to identify security weaknesses in websites. It targets common vulnerabilities like XSS (Cross-Site Scripting), SQL Injection, and missing security headers. This project provides a simple interface to scan websites, log vulnerabilities, and display results in a user-friendly manner.

Abstract

This project demonstrates the development of a web application security scanner using Python and Flask. By automating the detection of common web vulnerabilities, it helps developers and security enthusiasts understand and mitigate potential risks in web applications. The scanner crawls web pages, tests forms for XSS and SQL Injection, checks HTTP security headers, logs vulnerabilities in a database, and presents the results clearly through a web interface.

Tools Used

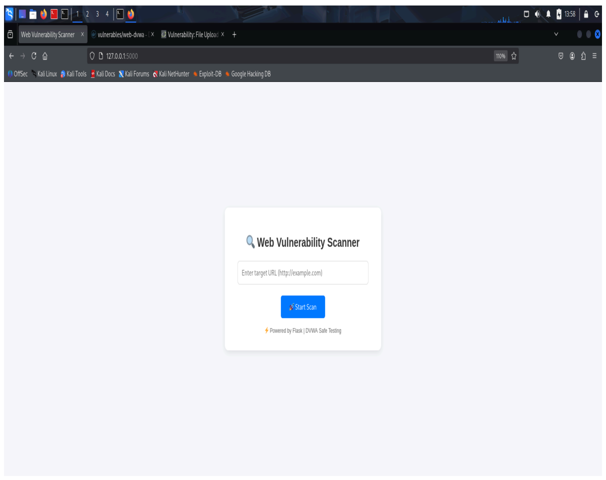
- **Python 3**: Programming language for backend logic.
- **Flask**: Framework for building the web interface.
- **Requests**: To send HTTP requests to target websites.
- **BeautifulSoup4**: To parse HTML and extract forms.
- **SQLite3**: To store vulnerability results.
- **HTML/CSS**: For creating the web-based UI.

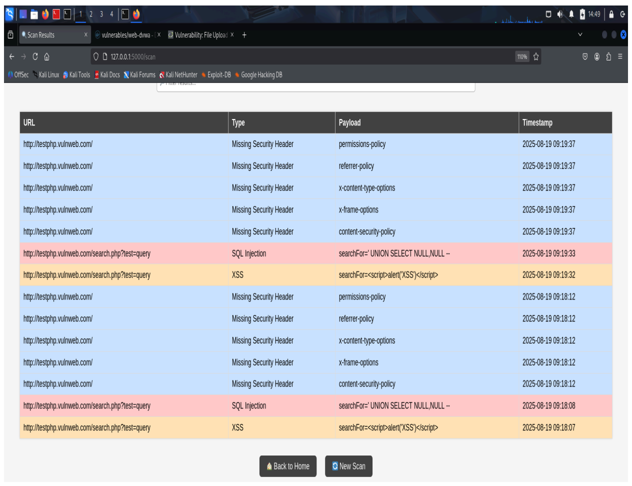
Steps Involved in Building the Project

1. **Setup Environment**: Install Python, create a virtual environment, and install required packages (Flask, Requests, BeautifulSoup4).

2. **Design the Web Interface**: Create HTML templates for input (index.html) and results display (results.html).
3. **Crawler Implementation**: Build a module to crawl target URLs and extract all forms for testing.
4. **Scanner Implementation**: Develop scripts to:
   - Test each form input for XSS and SQL Injection vulnerabilities.
   - Check for missing critical HTTP security headers.
   - Log any found vulnerabilities into an SQLite database.
5. **Integrate Modules**: Connect the crawler and scanner modules with the Flask application to run scans upon user input.
6. **Display Results**: Retrieve logged vulnerabilities from the database and present them in a clear table on the results page.
7. **Testing**: Run the scanner on known test sites like http://testphp.vulnweb.com to verify detection and logging of vulnerabilities.

Screenshots


Web Vulnerability Scanner
Enter target URL (http://example.com)
Start Scan
Powered by Flask | DVWA Safe Testing



URL	Type	Payload	Timestamp
http://testphp.vulnweb.com/	Missing Security Header	permissions-policy	2025-08-19 09:19:37
http://testphp.vulnweb.com/	Missing Security Header	referrer-policy	2025-08-19 09:19:37
http://testphp.vulnweb.com/	Missing Security Header	x-content-type-options	2025-08-19 09:19:37
http://testphp.vulnweb.com/	Missing Security Header	x-frame-options	2025-08-19 09:19:37
http://testphp.vulnweb.com/	Missing Security Header	content-security-policy	2025-08-19 09:19:37
http://testphp.vulnweb.com/search.php?test=query	SQL Injection	searchFor=' UNION SELECT NULL,NULL --	2025-08-19 09:19:33
http://testphp.vulnweb.com/search.php?test=query	XSS	searchFor=<script>alert('XSS')</script>	2025-08-19 09:19:32
http://testphp.vulnweb.com/	Missing Security Header	permissions-policy	2025-08-19 09:18:12
http://testphp.vulnweb.com/	Missing Security Header	referrer-policy	2025-08-19 09:18:12
http://testphp.vulnweb.com/	Missing Security Header	x-content-type-options	2025-08-19 09:18:12
http://testphp.vulnweb.com/	Missing Security Header	x-frame-options	2025-08-19 09:18:12
http://testphp.vulnweb.com/	Missing Security Header	content-security-policy	2025-08-19 09:18:12
http://testphp.vulnweb.com/search.php?test=query	SQL Injection	searchFor=' UNION SELECT NULL,NULL --	2025-08-19 09:18:08
http://testphp.vulnweb.com/search.php?test=query	XSS	searchFor=<script>alert('XSS')</script>	2025-08-19 09:18:07

Conclusion

The Web Application Vulnerability Scanner provides an automated way to identify critical security issues in web applications. It serves as an educational tool for understanding common vulnerabilities and demonstrates the integration of Python, web scraping, and database logging. Future improvements can include expanding vulnerability coverage, adding visual graphs for results, and providing export options for reports.