Web Application Vulnerability Scanner

Introduction

The Web Application Vulnerability Scanner project aims to develop a tool capable of automatically identifying commonly known vulnerabilities in web applications. This tool is designed with ethical hacking and security education in mind, providing users with insights into how vulnerable components of websites can be exploited. The focus of the project is primarily on detecting SQL Injection (SQLi) and Cross-Site Scripting (XSS), which are two of the most critical security flaws according to the OWASP Top 10.

Abstract

This project involves the creation of a Python-based scanner that automates the discovery and testing of input fields on web applications. It combines the functionality of a crawler, vulnerability scanner, and a user-friendly web interface built using Flask. By leveraging well-known Python libraries like requests and BeautifulSoup, the scanner navigates through the target website, collects forms and links, and applies a series of payloads to test for XSS and SQLi vulnerabilities.

The scanner uses various SQLi payloads to detect improper input sanitization that could lead to unauthorized data access or manipulation. Similarly, it tests for reflected XSS by injecting JavaScript-based payloads and checking if they are executed or reflected in the web page response. All discovered vulnerabilities are presented in a structured HTML report generated through the Flask interface.

Tools Used

- Python: Core language for scripting and logic implementation.
- Flask: Lightweight web framework used to build the user interface.
- requests: Used to send HTTP requests and simulate browser actions.
- BeautifulSoup: HTML parsing library used for crawling and form extraction.
- HTML/CSS: Used for building the web front-end interface.
- Jinja2: Templating engine used with Flask to render dynamic results.
- urllib.parse: For handling URL joins and parameter manipulations.

Steps Involved in Building the Project

1. Setting Up the Environment:
   - Initialized the Python environment and created the folder structure.
   - Installed necessary packages such as Flask, requests, and BeautifulSoup using pip.
   - Created a versioned requirements.txt file to document dependencies.

2. Web Crawler Development:
   - Developed a module to crawl websites starting from a given URL.
   - Used BeautifulSoup to extract anchor tags and form elements.
   - Stored all discovered resources for further analysis.

3. SQL Injection Scanning:
   - Created a scanner that targets forms and submits malicious payloads.

- Used different SQL dialects (MySQL, PostgreSQL, MSSQL) for comprehensive testing.
- Identified vulnerabilities based on error messages and response behavior.
- Implemented time-based payloads to simulate blind SQLi detection.

4. XSS Detection:
- Tested links and form fields for reflected input.
- Injected multiple encoded and obfuscated XSS payloads.
- Verified if the input was reflected directly or via encoding mechanisms.
- Recorded all occurrences and summarized them in the result view.

5. Web Interface Creation:
- Designed a simple front-end with Flask and HTML templates.
- Created a homepage to input the target URL.
- Built a results page to display all identified vulnerabilities.
- Integrated scan logic with the front-end via Flask routes.

Conclusion

This project served as a practical introduction to web application security, focusing on how vulnerabilities can be automatically detected. By building a scanner from scratch, I gained insights into the real-world application of HTTP protocols, form submission mechanics, and response handling.

The project successfully identified common web vulnerabilities in a controlled and ethical environment. It highlighted the importance of input validation and output encoding as first-line defenses in web security.

Future improvements could include:
- Support for CSRF and File Inclusion detection.
- Exporting results to PDF or CSV formats.
- Adding authentication bypass and session hijack tests.
- Real-time scan progress indicators and result graphs.

Overall, this project reinforced my understanding of secure coding practices and introduced me to the workflow of developing security tools used by ethical hackers and penetration testers.