Web Vulnerability Scanner – Project Report

Abstract

This project is about building a Web Vulnerability Scanner in Python. The tool helps in finding security weaknesses in websites by automatically checking for issues like SQL Injection, Cross-Site Scripting (XSS), and Directory Traversal. Instead of manually testing every page, the scanner crawls through the site, injects test payloads, and reports any potential vulnerabilities. It also provides both a command-line option and a web interface to make it easy to use.

Introduction

Websites today are exposed to different kinds of cyber-attacks. Attackers often take advantage of weak spots in web applications, and finding these flaws manually can take a lot of time. This project was built to solve that problem by creating an automated tool that can check websites for common vulnerabilities. The aim is to make the process faster, reliable, and more systematic while also giving the user clear results in the form of reports.

Tools Used

- **Python 3** – main programming language.
- **Requests** – to send and receive web requests.
- **BeautifulSoup (bs4)** – to read and extract links from web pages.
- **ThreadPoolExecutor** – to scan multiple links at the same time.
- **Flask** – to provide a simple web interface.
- **Regular Expressions (re)** – to detect error messages in responses.
- **OS and Sys** – for running system and command-line tasks.

Steps Involved in Building the Project

1. **Crawling the Website**
   The scanner first collects links from the target site using BeautifulSoup. It makes sure to stay within the same domain and avoids visiting the same page twice.
2. **Injecting Payloads**
   Test inputs (payloads) are added into the website's URL parameters to check how the site responds. This helps in spotting vulnerabilities like SQL Injection, XSS, and Directory Traversal.

3. **Detecting Vulnerabilities**
   The tool looks for signs such as error messages, script execution, or sensitive file content to confirm if a vulnerability exists.
4. **Parallel Scanning**
   To make the process faster, the scanner checks multiple URLs at the same time using multithreading.
5. **Reporting the Results**
   The results are displayed in the terminal and also saved as an HTML report. A simple Flask-based web page is included where users can enter a URL, start the scan, and see the results in their browser.

Conclusion

The Web Vulnerability Scanner works as an automated way to find common security issues in websites. It can detect SQL Injection, XSS, and Directory Traversal vulnerabilities, which are among the most frequent web threats. The tool also shows how security testing can be made faster and more user-friendly with automation. While this project covers only a few basic checks, it can be extended further to detect more advanced security risks. Overall, it highlights the importance of testing web applications regularly to keep them safe.