Web Application Vulnerability Scanner

Project Report

Introduction

In the ever-evolving world of cybersecurity, identifying software vulnerabilities early is essential. This project aims to automate the detection of security flaws in web applications, network configurations, and codebases using pattern-matching techniques. Designed for ethical hackers, developers, and students, this scanner enhances awareness and response to common vulnerabilities.

Abstract

This project presents a lightweight yet effective Python-based vulnerability scanner. It uses static analysis and pattern-matching logic to detect a wide range of security issues such as SQL Injection, Cross-Site Scripting (XSS), insecure API usage, leaked credentials, and misconfigured ports. The system includes both a terminal interface and a GUI for better user experience. The scanner analyzes uploaded source code and folders to identify known vulnerability patterns using regular expressions.

Its modular architecture makes it extensible, and multithreading improves its scanning performance. The final output highlights vulnerabilities with file names, line numbers, and matched context. The goal is to help users proactively fix issues before exploitation.

Tools Used

- Programming Language: Python

- GUI Framework: Tkinter

- Regex Module: re (for pattern matching)

- Networking: socket, requests

- File Handling: os, filedialog

- Scanning Modules:

- scan_web.py – Web-related vulnerabilities

- scan_ports.py – Open port scanning

- scan_software.py – Code/static analysis

- scan_network.py, scan_api.py – Additional layers

- ZIP Upload Support for scanning offline codebases

Steps Involved in Building the Project

1. Pattern Definition

Created a file patterns.py that contains known vulnerability patterns categorized as web, software, network, or API types. Each pattern includes regex, severity, and description.

2. Scanner Module Development

Built separate modules (scan_web.py, scan_software.py, etc.) that accept folders/files and scan them for pattern matches using regular expressions.

3. Multithreading for Performance

Used Python's threading module to perform multiple scans in parallel to reduce overall scan time.

4. GUI Interface with Tkinter

Designed a simple GUI using Tkinter allowing users to upload folders or ZIPs, run scans, and view results in a user-friendly window.

5. Result Display and Context Extraction

Matched vulnerabilities are shown with line number and file path. For each match, a few lines of context are also displayed to help understand the issue.

6. Final Integration

All modules were integrated into a main GUI-based application. The terminal-based version also supports colorful output using colorama.

Conclusion

This project demonstrates how static analysis and regular expression-based pattern matching can effectively detect common vulnerabilities. It provides both terminal and GUI interaction modes for accessibility and ease of use. Though not based on machine learning, its rule-based engine is extensible and ideal for educational and lightweight scanning needs. In future iterations, real-time monitoring, email alerts, and advanced plugins can be added for even greater functionality.