**WEB APPLICATION VULNERABILITY SCANNER**

PROJECT REPORT

**1.Introduction**

Web applications are increasingly targeted by attackers exploiting common vulnerabilities such as Cross-Site Scripting (XSS), SQL Injection (SQLi), and Cross-Site Request Forgery (CSRF). This project aims to build a lightweight, automated vulnerability scanner to detect these risks, helping developers and organizations secure their web apps proactively.

**2.Abstract**

The scanner uses Python libraries such as requests and BeautifulSoup to crawl web pages, extract input fields and URLs, and inject test payloads to identify potential XSS, SQLi, and CSRF vulnerabilities. The results are analyzed with pattern matching and presented through a simple Flask web interface. This tool serves educational and practical purposes for early vulnerability assessment with room for enhancement.

**3.Tools Used**

* Python 3.11: Core development language
* Flask: Web framework for UI and routing
* requests: HTTP client for crawling and sending payloads
* BeautifulSoup: HTML parsing for identifying inputs and links
* Regex: Pattern matching in responses to detect vulnerabilities
* WSL Ubuntu & Visual Studio Code: Development environment

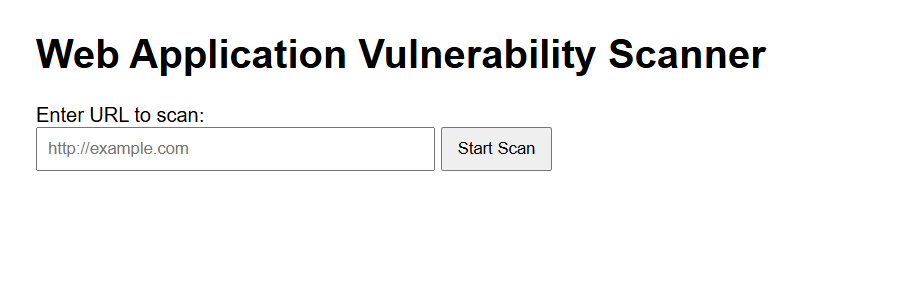
**4.Steps Involved in Building the Project**

1. Environment Setup: Installed Python 3.11 on WSL, created a virtual environment, and installed required libraries.
2. Crawling Mechanism: Developed a crawler that requests the target URL and parses the HTML to find all input fields and links using BeautifulSoup.
3. Payload Injection & Detection: Implemented basic payloads to test for XSS, SQLi, and CSRF, analyzing server responses for evidence of vulnerability.
4. Flask UI Development: Built web forms to input scan URLs and dynamic result pages to display findings including detected inputs, links, and vulnerability status.
5. Testing: Conducted scans on demo vulnerable websites such as testphp.vulnweb.com, verified correct crawling and vulnerability identification workflow.
6. Logging & Reporting: Designed basic logging of vulnerabilities and scan metadata for record keeping and audit purposes.

**5.Conclusion**

This project successfully demonstrates automated detection of common web vulnerabilities leveraging Python’s ecosystem and Flask for a user-friendly interface. It strengthens understanding of web security concepts and practical scanning techniques. Future improvements include expanding tested vulnerability types, refining payload injection methods, improving asynchronous task handling, and generating detailed reports with remediation guidance.

**SAMPLE:**



**RESULT PAGE:**

