Project Report: Web Application Vulnerability Scanner

Introduction

During my internship, I worked on building a web application vulnerability scanner. The main purpose of this project was to create a tool that can automatically check websites for common security issues. Many applications are vulnerable to attacks like SQL Injection and Cross-Site Scripting, and this scanner helps to find such problems before an attacker can exploit them. The project also gave me practical exposure to how vulnerabilities can be detected and reported in real time.

Abstract

The project is a Python-based scanner that crawls through a website, looks for input fields, and tries different payloads to check if the site is secure or not. It analyzes the responses and reports if something suspicious is found. A simple web interface was made using Flask so that scans can be started easily and results can be viewed in one place. The tool is lightweight and was mainly focused on the top issues listed in OWASP Top 10. Through this project I learned the process of security testing in a structured way.

Tools Used

- Python 3
- Requests (for sending HTTP requests)
- BeautifulSoup (for crawling and HTML parsing)
- Regex (for analyzing server response patterns)
- Flask (for the dashboard interface)
- Reference: OWASP Top 10 vulnerabilities

Steps Involved in Building the Project

1. Decided the scope of the scanner based on common web vulnerabilities.
2. Used Requests and BeautifulSoup to crawl a target website and collect links and forms.
3. Wrote functions to test for SQL Injection, XSS, and CSRF by injecting payloads.
4. Used regex and response checks to confirm if a vulnerability exists.
5. Added logging so that every scan result is saved with details and severity.
6. Created a Flask interface where a user can start scans and view reports.
7. Finally, tested the scanner on demo websites to check if detection was working correctly.

Conclusion

This project helped me understand how vulnerabilities are detected in web applications and why security testing is important. I gained hands-on experience with Python libraries, request handling, and response analysis. The scanner worked successfully for basic vulnerabilities and gave meaningful results during testing. In the future, it can be improved by adding more types of checks, making the reports more detailed, and integrating the tool into a continuous security testing process.