Web Application Vulnerability Scanner – Internship Project Report

# Introduction

This project is a Python-based Web Application Vulnerability Scanner designed to identify common web vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS). It simulates the basic behavior of a real-world automated scanner and serves as a foundational security testing tool.

# Abstract

The scanner takes a user-provided URL, crawls all reachable internal pages, identifies HTML forms, and automatically injects malicious payloads to detect vulnerabilities. It reflects real offensive security workflows and demonstrates how vulnerability detection can be automated using Python. A user-friendly web interface was also created using Flask.

# Tools Used

- Programming Language: Python  
- Libraries: requests, BeautifulSoup, Flask, re, datetime  
- Technologies: HTML, HTTP methods, Regex pattern matching  
- Others: File I/O, Flask templates (Jinja2), Manual testing via test environments

# Steps Involved in Building the Project

1. Crawling:  
 Implemented a crawler that recursively explores URLs and avoids duplicates.  
  
2. Form Extraction:  
 Parsed all forms on each page using BeautifulSoup to collect input fields, methods, and actions.  
  
3. Payload Injection:  
 Injected test payloads (e.g., <script>alert(1)</script> and ' OR '1'='1) into form fields and submitted them using GET/POST.  
  
4. Vulnerability Detection:  
 Analyzed server responses using regex and pattern matching to detect XSS or SQL error signatures.  
  
5. Logging:  
 Logged all results (vulnerability type, evidence, severity, form info) into a structured log file for further analysis.  
  
6. Flask UI:  
 Developed a basic Flask web app that allows scanning through a browser, with visual output.

# Conclusion

The Web Vulnerability Scanner is capable of identifying reflected XSS and basic SQL injection vulnerabilities. It simulates a simplified version of scanners like Burp Suite or OWASP ZAP. This project improved my understanding of HTTP requests, security testing, and web automation. Future enhancements could include support for CSRF, more payloads, deeper crawling, and full PDF report generation.