**Title**: Web Application Vulnerability Scanner  
**Domain**: Cyber Security  
**Student Name**: *Aarya Suresh*  
**Submission Date**: 8th September 2025

**1. Introduction**

In the modern IT industry, web applications are the backbone of businesses. However, they are also common targets for cyberattacks such as SQL Injection (SQLi) and Cross-Site Scripting (XSS).  
This project focuses on building a **lightweight Web Application Vulnerability Scanner** that identifies basic security flaws in web applications. The scanner helps developers and security analysts detect potential vulnerabilities early, thereby improving the overall security posture of applications.

**2. Abstract**

The Web Application Vulnerability Scanner is a Python-based tool with a simple **Flask web interface**.  
It performs:

* **Web crawling** to identify input fields and URLs.
* **SQL Injection testing** by injecting common payloads and detecting error patterns.
* **Cross-Site Scripting testing** by inserting malicious scripts and checking reflections.

The system provides results in a **tabular dashboard** and highlights vulnerable endpoints. This scanner can be used for educational and training purposes to demonstrate common web application weaknesses.

**3. Tools Used**

* **Programming Language**: Python
* **Libraries**: requests, BeautifulSoup, Flask
* **Frontend**: HTML, CSS (Flask templates)
* **Platform**: Runs locally on any OS with Python installed
* **Version Control**: GitHub (for code submission & documentation)

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

1. **Requirement Analysis**: Studied OWASP Top 10 vulnerabilities, focused on SQLi & XSS.
2. **Environment Setup**: Installed Python, Flask, and dependencies.
3. **Crawling Module**: Implemented a crawler using requests + BeautifulSoup to extract links and forms.
4. **Vulnerability Detection**:
   * Injected SQLi payloads (' OR '1'='1) and checked for error messages.
   * Injected XSS payload (<script>alert(1)</script>) and verified reflections.
5. **Flask Dashboard**: Built a simple UI for URL input and results display.
6. **Testing & Validation**: Ran scanner on test web pages and confirmed detection.

**5. Conclusion**

This project successfully demonstrates the process of scanning web applications for common vulnerabilities. While limited to SQLi and XSS detection, it forms a foundation for future expansion into other OWASP vulnerabilities such as CSRF, command injection, and insecure direct object references.

