📃 Web Application Vulnerability Scanner - Project Report

📄 Page 1 of 2

🔹 Project Overview	
Project Title	Web Application Vulnerability Scanner
Technology	Python, Flask, BeautifulSoup4, Requests, lxml
Duration	2 Weeks (Internship Project)
Domain	Cybersecurity – Web Application Testing

💡 1. Introduction

In today's interconnected digital landscape, web applications are frequently targeted by malicious actors. Identifying and mitigating vulnerabilities is crucial for maintaining data integrity, user privacy, and system availability. This project aimed to develop a **"Web Application Vulnerability Scanner"** capable of automatically detecting common web vulnerabilities such as Reflected Cross-Site Scripting (XSS), SQL Injection (SQLi), and missing security headers. The scanner is designed as a Python-based tool with a user-friendly Flask web interface, providing a foundational understanding of automated vulnerability assessment.

🧠 2. Abstract

The Web Application Vulnerability Scanner is a Python 3 application leveraging **requests** for HTTP interactions, **BeautifulSoup4** for HTML parsing, and **Flask** for its graphical interface. It crawls a target web app, identifies forms, and injects non-destructive payloads to test for vulnerabilities. Detected issues (XSS, SQLi, missing headers) with severity and evidence are stored in a structured JSON report. Emphasis is placed on **ethical scanning practices** — polite delays and clear disclaimers ensure responsible educational use.

⚙️ 3. Tools Used

Tool / Library	Purpose
🐍 Python 3	Primary programming language
🌐 Flask	Web framework for the scanner's UI (app.py)
📡 requests	Handles HTTP requests for scanning
🧩 BeautifulSoup4	HTML parser to extract forms and links
⚡ lxml	High-performance XML/HTML parsing
🔎 re	Regular expressions for payload pattern matching

📃 json	Report generation and data structuring
🗂️ os	File and directory operations

🧩 4. Steps Involved in Building the Project

- 🔧 **Environment Setup:** Created a virtual environment and installed Flask, requests, bs4, and lxml.
- 🧱 **Core Logic (core.py):**
  - 🚀 **Request Handling:** Implemented GET/POST with exception handling and request throttling.
  - 🔗 **URL Crawling:** Recursively discovered same-domain links up to depth limits.
  - 📝 **Form Extraction:** Parsed HTML to locate form actions, methods, and fields.
- 🧪 **Testing Modules:**
  - ⚔️ **XSS Detection:** Injected safe `<script>` and `<img>` payloads; detected reflection in responses.
  - 💉 **SQL Injection:** Used harmless payloads like `' OR '1'='1` to detect SQL error messages or anomalies.
  - 🧰 **Header Analysis:** Checked for missing headers such as Content-Security-Policy and X-Frame-Options.
  - 💾 **Reporting:** Saved results as JSON with URL, payload, severity, and evidence snippet.

📄 Page 2 of 2 (Continued)

🌍 Web Interface Development (app.py, templates/)

- ⏳ **Flask Application:** Served as the front-end to accept target URLs and display results.
- 📃 **Input Form:** index.html allowed users to input a website URL for scanning.
- ⚙️ **Scan Execution:** /start-scan route invoked the core scanner and saved a JSON report.
- 📊 **Dynamic Reporting:** report.html presented findings in tables showing type, severity, payload, and evidence.
- ⚠️ **Ethical Disclaimer:** Displayed clear warning to scan only authorized systems.
- 🧪 **Example Target:** Built a simple vulnerable Flask app (search/login forms) for demo testing.

🏁 5. Conclusion

The **Web Application Vulnerability Scanner** successfully automates the detection of common web flaws such as XSS, SQLi, and missing security headers. By integrating crawling, form parsing, and safe payload injection within a Flask UI, the project demonstrates the fundamentals of web vulnerability assessment. It reinforces key cybersecurity principles — automation, ethical testing, and secure coding awareness. Future enhancements may include deeper scan logic (DOM-based XSS, blind SQLi), authentication support, and visual analytics dashboards.

✅ **Submitted By:** [FATEHALI ABBASALI MAKNOJIYA] 🎓 **Internship Project Duration:** 2 Weeks
📅 **Year:** 2025

Mentor: [ELEVATE LABS]