**Web Application Vulnerability Scanner Report**

**Introduction:**

* Web applications are frequent targets for cyberattacks due to vulnerabilities from poor coding practices and lack of input validation.
* This project focuses on creating a scanner to detect common vulnerabilities using automated techniques.
* The objective is to simulate real-world attacks using headless browser automation.
* The tool features a web interface for usability and provides detailed scan reports for further analysis.

**Abstract:**

* This tool is built using Python, combining Flask, Selenium, and BeautifulSoup libraries.
* It focuses on detecting three major OWASP Top 10 vulnerabilities: XSS, SQL Injection, and CSRF.
* It crawls target web applications, extracts forms, injects malicious payloads, and monitors for abnormal behaviour.
* The scanner mimics attacker behaviour in a safe and controlled environment.
* Reports are generated in both HTML and JSON formats to support documentation and auditing.
* This project encourages practical learning in cybersecurity and enhances offensive security skills.

**Tools Used:**

* Programming Language: Python 3.x
* Web Framework: Flask (for web interface)
* Automation Framework: Selenium with Firefox (Geckodriver)
* HTML Parsing: BeautifulSoup4
* Network Requests: Requests library
* Data Handling: JSON and urllib
* Test Target: http://testphp.vulnweb.com (intentionally vulnerable site)

**Steps Involved in Building the Project:**

1. **Setup Development Environment**:
   * Installed Python and required packages in a virtual environment.
   * Installed external tools like Firefox and Geckodriver.
2. **Designed the Core Modules**:
   * Developed a crawler using BeautifulSoup to locate forms and extract parameters.
   * Developed a detection engine to test for XSS, SQLi, and CSRF vulnerabilities.
3. **Implemented Detection Logic**:
   * XSS: Injected JavaScript payloads and checked reflection in page content.
   * SQLi: Manipulated query strings with SQL payloads and detected error patterns.
   * CSRF: Verified presence or absence of CSRF tokens in POST forms.
4. **Simulated Browser Behaviour**:
   * Used Selenium in headless mode to mimic real user interaction and form submissions.
5. **Developed the Web Interface**:
   * Used Flask to build a UI for entering URLs, initiating scans, and displaying results.
   * Designed dark-themed UI with categorized output for readability.
6. **Added Report Generation**:
   * Implemented HTML report export for submission.
   * Added JSON logging for audit and debugging.
7. **Final Testing and Optimization**:
   * Tested the tool on known targets to validate vulnerability detection.
   * Optimized code structure, input validation, and performance.

**Conclusion:**

* The scanner accurately detects major web vulnerabilities using real attack payloads.
* It offers a realistic simulation of attacker behaviour in a secure environment.
* Results are presented clearly in a categorized format for professional use.
* The project enhances knowledge in both offensive security and secure development.
* Future improvements may include login-authenticated scans, PDF exports, and advanced evasion techniques.
* Overall, the tool meets the goal of being an efficient, user-friendly, and educational security scanner.