Software Requirements Specification (SRS)

Project Title: Web Application Vulnerability Scanner

Version: 1.0

Date: October 2, 2024

Prepared by: Bibek Subedi

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the Web Application Vulnerability Scanner. The system will be developed to scan web applications for common vulnerabilities like SQL injection, cross-site scripting (XSS), XXE, SSRF, RCE, and more based on the OWASP Top 10 vulnerabilities. The scanner will help developers and security professionals identify and assess vulnerabilities in their web applications.

1.2 Scope

The Web Application Vulnerability Scanner will allow users to enter a URL and scan for various security vulnerabilities. The system will check for vulnerabilities across common endpoints, analyze server responses, and classify the results. Vulnerabilities will be logged and reported in a comprehensive format. Additionally, the scanner will perform brute force login attempts and anomaly detection in server responses.

1.3 Definitions, Acronyms, and Abbreviations

- **SQL Injection:** A code injection technique that exploits vulnerabilities in SQL query handling.
- **XSS (Cross-Site Scripting):** A security vulnerability that allows attackers to inject malicious scripts into webpages.
- **XXE (XML External Entity):** A vulnerability that allows attackers to inject XML entities that may point to sensitive files.
- **SSRF (Server-Side Request Forgery):** An attack that allows attackers to send requests from the vulnerable server.
- **RCE (Remote Code Execution):** A vulnerability that allows attackers to execute code on a remote server.
- **OWASP Top 10:** A list of the most critical security risks to web applications, maintained by the Open Web Application Security Project.

1.4 Overview

This SRS document contains the following sections:

- Functional and non-functional requirements
- System and user interface design details
- External interfaces

- Constraints and dependencies

---

2. Overall Description

2.1 Product Perspective

The Web Application Vulnerability Scanner will be a standalone Flask-based web application. It will allow users to input a URL, and it will scan various endpoints of that URL for known vulnerabilities. The system will make use of machine learning models to classify the type of vulnerability and detect anomalies in server responses.

2.2 Product Features

- **URL Input and Scanning:** Users can input a URL, and the system will check common endpoints for security vulnerabilities.
- **Vulnerability Classification:** The system uses a pre-trained model to classify the vulnerability types.
- **Anomaly Detection:** The system will detect anomalous server responses based on historical patterns using a pre-trained model.
- **Brute Force Login:** Simulates common username and password combinations for brute force attacks.
- **OWASP Top 10 Testing:** Tests vulnerabilities related to the OWASP Top 10 categories, such as SQL Injection, XSS, SSRF, and more.
- **Detailed Reporting:** Results of the vulnerability scan will be displayed to the user in a formatted HTML report.
- **Logging:** Vulnerability results are logged to an HTML file with timestamps and severity levels.

2.3 User Classes and Characteristics

- **Security Professionals:** Will use the tool to perform vulnerability scans on their applications.
- **Developers:** Will use the tool to test their web applications for security flaws.
- **Penetration Testers:** Will use the tool for scanning known vulnerabilities in web applications.

2.4 Operating Environment

The system will be a web-based application running in the following environment:

- **Operating System:** Windows, Linux, macOS
- **Software Requirements:** Python 3.8+, Flask, TensorFlow, Hugging Face Transformers, Scikit-learn
- **Hardware Requirements:** Minimum 4GB RAM, 10GB disk space

2.5 Design and Implementation Constraints

- The system must use pre-trained models (Hugging Face Transformers) for vulnerability classification.

- The system must ensure TensorFlow uses GPU memory on-demand, if available.
- The system must comply with ethical use standards and should not be used for unauthorized testing.

2.6 Assumptions and Dependencies

- Users must have access to the URLs they are scanning, and the tool must be used with permission.
- The system assumes the availability of an internet connection to perform live vulnerability checks.

---

3. Functional Requirements

3.1 Input URL and Scan

- **Description:** The system must accept a URL as input from the user.
- **Pre-condition:** The user must provide a valid URL.
- **Post-condition:** The system will start vulnerability scans for the provided URL.

3.2 Vulnerability Scanning

- **Description:** The system will check known paths (e.g., /wp-admin, /wp-login.php, /phpinfo.php) for vulnerabilities.
- **Pre-condition:** The user has submitted a valid URL.
- **Post-condition:** The system will perform GET and POST requests to the URL and analyze the responses for vulnerabilities.

3.3 Vulnerability Classification

- **Description:** The system will classify the type of vulnerability using a pre-trained machine learning model.
- **Pre-condition:** The system must receive a response from the server.
- **Post-condition:** The response will be classified as one of the vulnerability types (e.g., SQL Injection, XSS, CSRF, etc.).

3.4 Brute Force Login Testing

- **Description:** The system will attempt brute force login with common username and password combinations.
- **Pre-condition:** The system must identify a login page (e.g., /wp-login.php).
- **Post-condition:** The system will report if any username and password combination successfully logs in.

3.5 Logging Vulnerabilities

- **Description:** The system will log vulnerabilities and errors to an HTML file.

- **Pre-condition:** Vulnerabilities or errors have been identified.
- **Post-condition:** The vulnerability information will be appended to a log file with timestamps and severity levels.

3.6 Anomaly Detection

- **Description:** The system will detect anomalies in the server responses based on a trained anomaly detection model.
- **Pre-condition:** Responses from the server must be received.
- **Post-condition:** Anomalous responses will be flagged and reported.

---

4. Non-Functional Requirements

4.1 Performance Requirements

- The system must perform each scan within a reasonable time frame, typically under 10 seconds per vulnerability check.

4.2 Reliability

- The system should handle timeouts and server errors gracefully and continue scanning.

4.3 Security

- The system must not store or transmit sensitive data outside of the scanning session.
- The system should log only necessary information and should not store personal data.

4.4 Usability

- The system must have a simple web interface that allows users to enter URLs and view scan results easily.

4.5 Maintainability

- The codebase should be modular and well-documented for easy updates and maintenance.

---

5. External Interface Requirements

5.1 User Interfaces

- **Input:** A form for users to input a URL for scanning.
- **Output:** An HTML page showing vulnerability results, classifications, and a detailed report.

5.2 Hardware Interfaces

No specific hardware interfaces are required.

5.3 Software Interfaces

- Flask will serve as the web framework.

- TensorFlow will be used for machine learning classification.
- Hugging Face Transformers for pre-trained model integration.

---

6. System Features

Feature	**Priority**	**Description**
URL Scanning	High	Allow users to input a URL and scan for vulnerabilities.
Vulnerability Classification	High	Use a machine learning model to classify vulnerabilities found in the scan.
Anomaly Detection	Medium	Detect anomalies in server responses based on historical patterns.
Brute Force Login Testing	Medium	Perform brute force login attempts on common login pages using a predefined list of username and password combinations.
Detailed Reporting	High	Provide detailed reports in the form of HTML logs.

---

7. Other Requirements

7.1 Legal and Ethical Considerations

- The system must not be used for unauthorized penetration testing or security analysis.
- Users must ensure they have permission to scan the target URLs.

This SRS serves as a guideline for the development and implementation of the Web Application Vulnerability Scanner. Any future updates to the project will be reflected in subsequent versions of this document.