Project: SQLi Sentinel - A Web Vulnerability Scanner

Phase 1: Requirements Gathering and Analysis

1. Introduction

This document outlines the functional and non-functional requirements for **SQLi Sentinel**, a command-line tool designed to detect SQL injection vulnerabilities in web applications. The primary goal is to create an educational yet effective tool that demonstrates the principles of web security scanning.
Ethical Warning: This tool must only be used on web applications that you own or have explicit, written permission to test. Unauthorized scanning of websites is illegal and unethical.

2. Functional Requirements (What the system will do)

These are the core features of our tool.
FR1: Target Specification

- The user must be able to specify a single starting URL for the scan.
- The tool will only scan pages that are part of the same domain as the starting URL (e.g., if the start is http://example.com/page1, it will scan http://example.com/page2 but not http://google.com).

FR2: Web Crawler / Spider

- The tool shall automatically discover links (from <a> tags) on the target website, starting from the initial URL.
- It will maintain a list of visited URLs to avoid redundant scanning and infinite loops.
- The crawler will explore the website up to a user-defined depth to control the scan's scope.

FR3: Vulnerability Vector Identification

- The tool must parse the HTML of each discovered page to identify potential points of injection.
- It must identify and extract all HTML forms (<form>) and their input fields (<input>, <textarea>, etc.).
- It must identify and extract all URL parameters (e.g., id from product.php?id=1).

FR4: SQL Injection Detection Engine
The core of the tool will implement multiple detection techniques:

- **FR4.1: Error-Based SQLi:** The tool will inject characters that break SQL syntax (like a single quote ') and analyze the server's response page for common database error messages (e.g., "SQL syntax," "unterminated string").
- **FR4.2: Boolean-Based (Content-Based) SQLi:** The tool will inject logical statements (' OR 1=1 and ' OR 1=2) and compare the HTML content of the resulting pages. A

significant difference in content indicates a vulnerability.

- **FR4.3: Time-Based (Blind) SQLi:** The tool will inject database-specific commands that cause a time delay (e.g., SLEEP(5)). It will measure the server's response time. A response time significantly longer than a baseline measurement indicates a vulnerability.

FR5: Reporting

- The tool will provide real-time output to the console as it works (e.g., "Crawling: [URL]", "Testing Form on: [URL]").
- At the end of the scan, it will generate a clear, concise summary report listing:
  - All URLs found to be vulnerable.
  - The specific parameter or form field that is vulnerable.
  - The type of SQL injection detected (Error-Based, Boolean-Based, etc.).

3. Non-Functional Requirements (How the system will operate)

These define the quality and usability of our tool.

NFR1: User Interface

- The tool will be a command-line interface (CLI) application. It will not have a graphical user interface (GUI).
- Input will be provided via command-line arguments (e.g., python sqli_sentinel.py --url http://test.com --depth 2).

NFR2: Performance

- The tool should be reasonably fast, but accuracy is more important than speed.
- It will send requests sequentially (one at a time) to avoid overwhelming the target server.

NFR3: Platform Compatibility

- The tool will be written in Python 3.
- It will rely on standard, well-known libraries (like requests and beautifulsoup4) to ensure it can run on Windows, macOS, and Linux.

NFR4: Documentation

- The code must be thoroughly commented to explain the logic, especially for complex parts like the detection engine.
- A README.md file will be created to explain how to install and run the tool.

4. Scope Limitations (What the system will *not* do)

- **No Exploitation:** The tool is for detection only. It will not attempt to extract data from the database, escalate privileges, or perform any other post-exploitation activities.
- **No JavaScript Rendering:** The crawler will analyze the raw HTML received from the server. It will not execute JavaScript, meaning it may not find links or forms generated dynamically on the client-side.
- **No Authentication:** The tool will not handle login forms or authenticated sessions. It will scan the website as a public, unauthenticated user.
- **Limited Payloads:** It will use a small, curated list of generic SQLi payloads. It will not have an exhaustive list for every type of database.