Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

This SRS specifies the functional and non-functional requirements for Orange Sage—an autonomous AI-driven cybersecurity assessment platform. It is intended for developers, testers, and evaluators responsible for the implementation, deployment, and validation of the system.

1.2 Scope

Orange Sage enables automated security assessments via AI agents, providing validated vulnerability findings, a web dashboard for management and reporting, and integration points for LLMs and object storage.

1.3 Definitions, Acronyms and Abbreviations

- LLM: Large Language Model

- API: Application Programming Interface

- JWT: JSON Web Token

- DB: Database (SQLite for local; PostgreSQL recommended for production)

2. Overall Description

2.1 Product Perspective

Orange Sage is a modular, Docker-friendly platform composed of a FastAPI backend, a React/Next.js frontend, and supporting services (Redis, MinIO). Agents are implemented as Python modules (e.g., pentesting_agent.py) and coordinated via Celery task queues backed by Redis.

2.2 Product Functions (high-level)

- User authentication & role-based access control (Admin, Developer, Auditor)

- Project and target management (CRUD)

- Scan orchestration and agent scheduling

- Real-time logging and WebSocket/Realtime updates

- Findings storage, triage, and report generation

2.3 User Characteristics

- Developers: run scans against their codebases or deployments

- Security Analysts: configure scans, validate findings, and perform triage

- Administrators: manage users, keys, and infrastructure

3. Specific Requirements

3.1 Functional Requirements

FR-1: User Authentication (JWT based)

FR-2: Project & Target CRUD

FR-3: AI Agents perform reconnaissance, scanning, and exploitation validation

FR-4: Real-time scan logging and progress visualization

FR-5: Findings storage with severity classification and remediation guidance

FR-6: Report generation in PDF/DOCX/HTML formats

FR-7: API endpoints under /api/v1 for all operations

3.2 Non-Functional Requirements

- Performance: Support multiple concurrent scans; maintain responsive UI under load.

- Scalability: Agents and workers should scale horizontally (Celery + Redis + multiple workers).

- Security: All endpoints authenticated and validated; sensitive keys stored in environment variables.

- Reliability: Task retries for transient failures; health checks for services.

3.3 External Interface Requirements

- REST API: Base path /api/v1; documented via Swagger UI and ReDoc.

- Frontend: React/Next.js communicates via VITE_API_BASE_URL.

- Storage: MinIO for artifacts; DB for findings.

3.4 System Features (Use Cases)

Use Case 1: Create Project and Add Target

Actors: Developer

Precondition: User authenticated

Main Flow: User creates project → adds target (URL/repo/upload) → configures scan → starts scan

Use Case 2: Run Autonomous Scan

Actors: AI Agent, System

Precondition: Target configured and reachable

Main Flow: System schedules agent job → agent performs recon, scanning, exploitation attempts → results stored → user notified

4. API Endpoints (selected)

- POST /api/v1/auth/login — Authenticate user and return JWT

- POST /api/v1/projects — Create a new project

- GET /api/v1/projects/{id}/targets — List targets for a project

- POST /api/v1/scans — Start a new scan/job

- GET /api/v1/scans/{id}/logs — Stream scan logs

- GET /api/v1/findings — List findings with filters (severity, project)

5. Data Requirements

Database schema includes tables/entities: users, projects, targets, scans, findings, reports. For local development a SQLite DB (orange_sage.db) is used; migrations and Alembic recommended for production with PostgreSQL.