UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

PWASP SCANNER – Sistema de Detección de Vulnerabilidades Web

Curso: *Patrones de Software*

Docente: Ing. Patrick Jose Cuadros Quiroga

Integrantes:

Ccalli Chata, Joel Robert ***(2017057528)***

Jarro Cachi, Jose Luis ***(2020067148)***

Tacna – Perú

2025

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	JCC	JRPC	JRPC	30/10/2024	Versión Original

PWASP SCANNER – Sistema de Detección de Vulnerabilidades Web

FD04 - Documento de Arquitectura de Software

Versión 1.0

ÍNDICE GENERAL

1. Introducción

1.1 Propósito del Documento

Este documento tiene como objetivo describir de manera detallada y formal la arquitectura del software del sistema **PWASP SCANNER – Sistema de Detección de Vulnerabilidades Web**. Su propósito es establecer una visión técnica clara sobre los componentes que conforman el sistema, sus relaciones, flujos de información, capas de arquitectura, modelos de datos, seguridad, patrones de diseño y decisiones tecnológicas. La información contenida en este documento servirá como guía principal para el equipo de desarrollo, asegurando la coherencia estructural y funcional del sistema en todas sus etapas de implementación y mantenimiento.

1.2 Alcance del Sistema

PWASP SCANNER es una herramienta web diseñada para detectar vulnerabilidades comunes en aplicaciones web, tales como inyecciones SQL, XSS, CSRF, entre otras, utilizando técnicas de escaneo automatizado y heurísticas basadas en el top 10 de OWASP. El sistema está orientado a profesionales de seguridad, desarrolladores y testers que deseen fortalecer la seguridad de sus plataformas digitales. La plataforma permite registrar proyectos, ejecutar análisis sobre URLs específicas, obtener reportes clasificados por nivel de riesgo, y recibir sugerencias de mitigación.

1.3 Audiencia Destinada

Este documento está dirigido a los siguientes grupos de interés:

- **Equipo de desarrollo**: Para implementar correctamente la arquitectura propuesta.
- **Ingenieros de software**: Para verificar la factibilidad técnica y asegurar la calidad del diseño.
- **Auditores de sistemas**: Para evaluar la solidez estructural y la seguridad del sistema.
- **Administradores de red**: Para conocer los componentes de integración con infraestructura.
- **Usuarios avanzados (pentesters y DevSecOps)**: Para entender el funcionamiento interno del escáner.

1.4 Definiciones y Acrónimos

- **PWASP**: Plataforma Web de Análisis de Seguridad y Penetración.
- **SAD**: Software Architecture Document.
- **XSS**: Cross Site Scripting.
- **CSRF**: Cross Site Request Forgery.
- **SQLi**: SQL Injection.
- **OWASP**: Open Web Application Security Project.

2. Representación Arquitectónica

2.1 Estilo Arquitectónico

El estilo arquitectónico adoptado para PWASP SCANNER es **arquitectura en capas (layered architecture)**, complementado con el enfoque de **microservicios modulares**, permitiendo independencia entre componentes, escalabilidad, seguridad reforzada y mantenimiento eficiente.

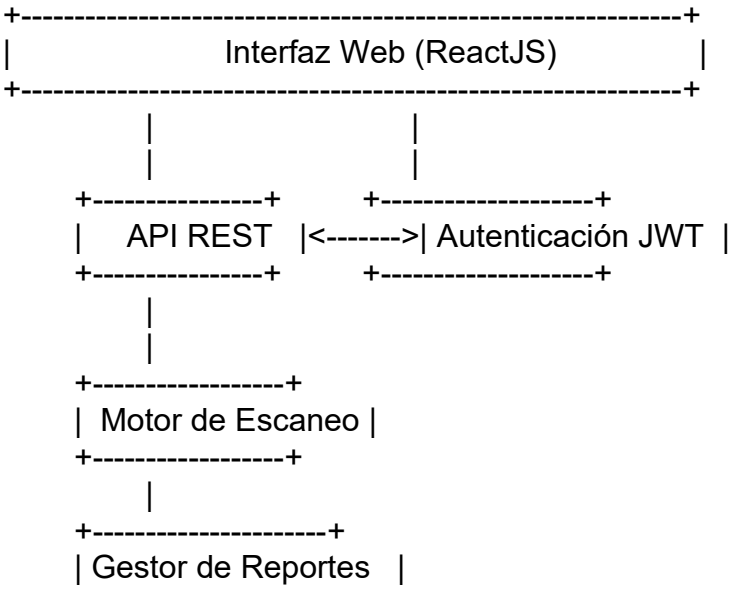
2.2 Descripción General de Componentes

El sistema se compone de los siguientes módulos principales:

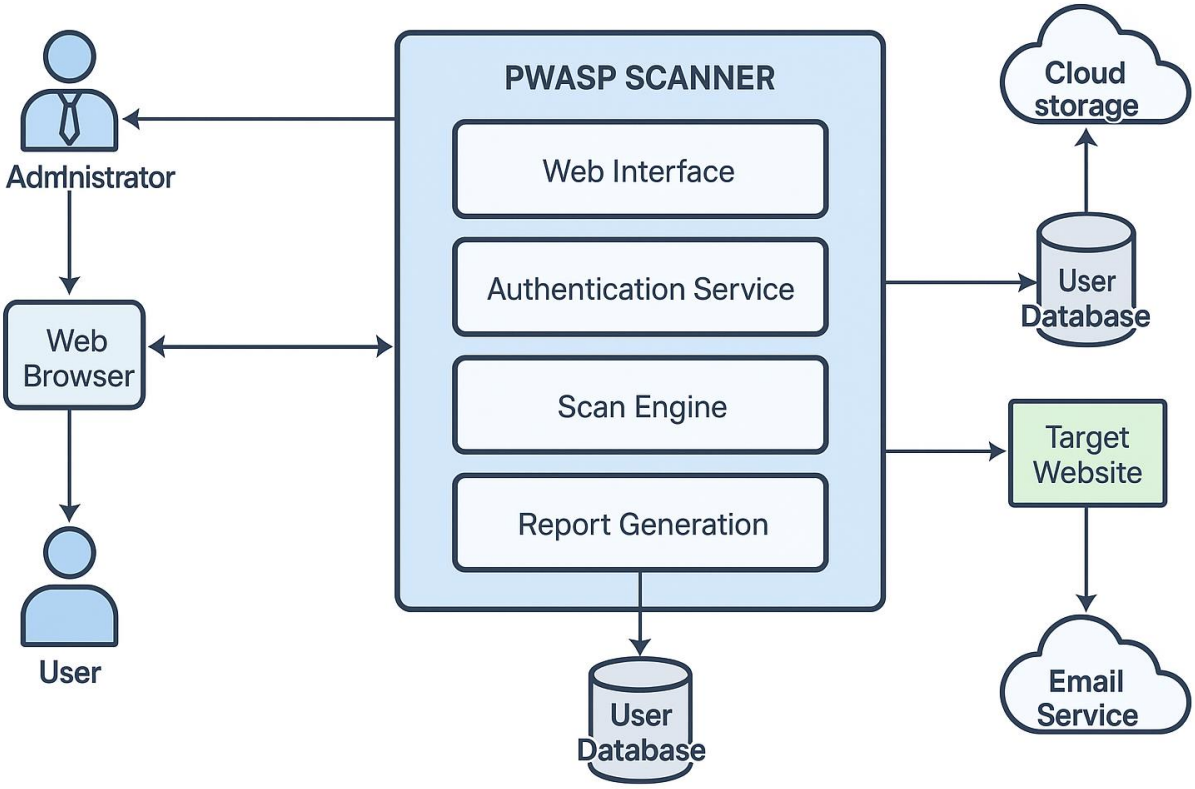
1. **Interfaz Web (Frontend)**: Plataforma de interacción con el usuario, desarrollada en ReactJS o Angular, que permite configurar escaneos, visualizar resultados y gestionar cuentas.
2. **Módulo de Autenticación**: Gestiona login, logout, tokens JWT, control de sesiones y permisos.
3. **Motor de Escaneo (Scanner Core)**: Componente central que ejecuta las técnicas de detección de vulnerabilidades, basado en Python con bibliotecas como requests, BeautifulSoup, OWASP ZAP API.
4. **Gestor de Reportes**: Se encarga de clasificar los hallazgos, generar informes técnicos y visuales en PDF o JSON.
5. **Base de Datos**: Almacena resultados de escaneos, registros de usuarios, configuraciones y logs.
6. **API REST**: Interfaz de comunicación entre frontend y backend, implementada en ASP.NET Core.
7. **Módulo de Seguridad**: Filtro de peticiones, detección de anomalías, validaciones y firewall lógico.
8. **Sistema de Alertas**: Notifica al usuario sobre escaneos completados o hallazgos críticos vía email o notificaciones push.

3. Diagrama de Arquitectura del Sistema

A continuación, se representa un esquema lógico del sistema (descripción textual, ya que no se puede mostrar una imagen directamente aquí):



```
+---------------------+
        |
+---------------------+
|     Base de Datos    |
+---------------------+
        |
+---------------------+
| Sistema de Alertas   |
+---------------------+
```



4. Componentes y Sus Responsabilidades

4.1 Interfaz Web (Frontend)

- Proporciona una experiencia gráfica amigable al usuario.
- Permite lanzar escaneos, ver resultados, configurar perfiles de análisis.
- Integración con gráficos (Chart.js) para resultados dinámicos.

4.2 API REST

- Puente entre la interfaz web y el backend.
- Gestiona operaciones CRUD, lanzamientos de escaneo, autenticación y obtención de reportes.

4.3 Motor de Escaneo

- Realiza crawling sobre las URLs ingresadas.

- Ejecuta pruebas de inyección automatizadas.
- Evalúa resultados según reglas definidas en base a OWASP.
- Soporta plugins externos de análisis.

4.4 Gestor de Reportes

- Clasifica hallazgos en niveles: alto, medio, bajo e informativo.
- Genera informes descargables en múltiples formatos.
- Incluye recomendaciones para mitigación.

4.5 Módulo de Seguridad

- Validación de entradas.
- Implementación de CORS, CSRF protection y X-Content-Type Options.
- Prevención de ataques comunes como DoS, enumeration y brute force.

4.6 Sistema de Alertas

- Configurable por usuario.
- Notificaciones ante hallazgos críticos o errores en escaneos.
- Integración con servicios de correo (SMTP) o API de notificaciones.

5. Modelo de Datos

El modelo se basa en una base de datos relacional SQL Server y contempla las siguientes entidades:

- **Usuario** (id, email, hash_password, rol)
- **Proyecto** (id, nombre, url_objetivo, usuario_id)
- **Escaneo** (id, fecha, estado, tiempo_total, proyecto_id)
- **Hallazgo** (id, tipo, severidad, descripción, recomendación, escaneo_id)
- **Log** (id, evento, fecha_hora, ip_origen, usuario_id)

Cada entidad está normalizada en 3FN y tiene índices optimizados para búsquedas por fechas y severidad.

6. Decisiones Tecnológicas

- **Frontend**: ReactJS con Tailwind CSS
- **Backend/API**: ASP.NET Core 8.0
- **Motor de escaneo**: Python 3.11 con OWASP ZAP API y herramientas propias
- **Base de datos**: SQL Server Express
- **Host**: Plataforma cloud (Azure o VPS con Docker)
- **Seguridad**: JWT para autenticación, HTTPS forzado, cifrado AES para datos sensibles

7. Requisitos de Seguridad

- Acceso protegido por roles (Administrador, Analista, Usuario estándar).
- Tokens JWT con expiración y renovación.
- Logs de acceso y auditoría.
- Escaneo de URLs internas o protegidas restringido por configuración.
- Saneamiento de inputs y outputs.

8. Patrones de Diseño Utilizados

- **MVC**: Separación lógica entre controladores, vistas y modelos.
- **Repositorio**: Abstracción de acceso a datos.
- **Singleton**: En el motor de escaneo para evitar instancias múltiples.
- **Factory**: Para construcción de diferentes tipos de reportes.

9. Requisitos de Rendimiento y Disponibilidad

- Escalabilidad horizontal con balanceador de carga.
- Respuesta promedio del API: < 300ms.
- Procesamiento de escaneo en background (concolic concurrency control).
- Alta disponibilidad mediante contenedores Docker y orquestación con Kubernetes (opcional).

10. Consideraciones Futuras

- Implementación de escaneo pasivo y análisis estático de código (SAST).
- Integración con CI/CD para DevSecOps.
- Traducción multilingüe para interfaz y reportes.
- Módulo de inteligencia artificial para priorización de amenazas.

11. Conclusión

La arquitectura propuesta para el sistema PWASP SCANNER proporciona una base sólida, modular, escalable y segura para la detección de vulnerabilidades web. El diseño en capas con microservicios independientes facilita su mantenimiento, mientras que el motor de escaneo basado en estándares OWASP garantiza efectividad en la identificación de riesgos. La utilización de tecnologías modernas y patrones de diseño adecuados permite al sistema evolucionar de forma ágil, adaptándose a nuevas amenazas y necesidades del usuario final.