**Documentación Técnica Profundizada para SentinelCore**

**1. Identificación de Módulos Funcionales**

**Módulos Principales y Subcomponentes**

| **Módulo** | **Submódulos** | **Descripción Técnica** |
| --- | --- | --- |
| **Toquen Manager** | Toquen Maestro | Clasifica archivos por tipo (magic numbers) y delega a toquens especializados |
|  | Toquen PDF | Sanitiza PDFs (elimina JS/embeds) con PyMuPDF |
|  | Toquen Office | Detecta macros maliciosas con oletools |
| **Escáner Web** | OWASP ZAP Integrator | Controla escaneos mediante API REST |
|  | Parser de Resultados | Transforma JSON de ZAP a informe estructurado |
| **Cloud Connector** | AWS Security Hub Fetcher | Recoge hallazgos de vulnerabilidades via boto3 |
|  | Azure Defender API Client | Consulta recomendaciones de seguridad |
| **Auth & Subdomains** | Nginx Config Generator | Genera reglas de no-indexación y autenticación básica |
|  | Google Search Console Cleaner | Remueve URLs indexadas mediante API |
| **Dashboard** | Frontend (Vue.js) | Muestra vulnerabilidades y estado de toquens |
|  | API Backend (FastAPI) | Expone endpoints para escaneos y gestión |

**2. Interfaces y Protocolos de Comunicación**

**Diagrama de Flujo de Comunicación**

**Especificación de APIs Clave**

**API Toquen Manager**

python

# Endpoints (FastAPI)

@app.post("/analyze")

async def analyze\_file(file: UploadFile):

"""

Input: File bytes

Output: { "status": "clean|infected", "actions": ["removed\_js"] }

"""

master = MasterToken(await file.read())

return master.route()

@app.get("/toquens/status")

async def get\_toquens():

"""

Devuelve estado de toquens activos

"""

return {"toquens": ["pdf", "office", "static\_analysis"]}

**Comunicación entre Toquens (Protocolo Interno)**

python

# Ejemplo: Toquen Maestro → Toquen PDF

class PDFSanitizer:

def sanitize(self, file\_bytes):

import fitz # PyMuPDF

doc = fitz.open(stream=file\_bytes)

for page in doc:

page.\_removeJs() # Elimina JavaScript

return doc.tobytes()

**3. Asignación de Responsables**

| **Módulo** | **Responsable** | **Tareas Críticas** |
| --- | --- | --- |
| **Toquen Manager** | Dev 1 (Python) | - Implementar clasificación de archivos - Integrar sanitizadores |
| **Escáner Web** | Dev 2 (DevOps) | - Dockerizar OWASP ZAP - Desarrollar parser de resultados |
| **Cloud Connector** | Dev 3 (Cloud) | - Configurar AWS/Azure APIs - Mapear vulnerabilidades a formato unificado |
| **Auth & Subdomains** | Dev 4 (SecOps) | - Generar configs Nginx automáticas - Implementar auth básica |
| **Dashboard** | Dev 1 + Dev 3 | - Frontend Vue.js - API FastAPI |

**4. Documento de Arquitectura Modular**

**Arquitectura Técnica**

plaintext

sentinelcore/

├── core/ # Lógica principal

│ ├── main.py # Punto de entrada

│ ├── dispatcher.py # Orquestación de toquens

├── toquens/ # Toquens especializados

│ ├── pdf.py # Sanitización PDF

│ ├── office.py # Análisis Office

├── webscanner/ # Escaneo web

│ ├── zap\_integration.py # Control de OWASP ZAP

├── cloud/ # Conectores cloud

│ ├── aws\_security.py # AWS Security Hub

├── auth/ # Protección subdominios

│ ├── nginx\_config.py # Generador de configs

└── frontend/ # Dashboard

├── src/ # Vue.js

**Principios Clave**

1. **Desacoplamiento**: Cada toquen es un microservicio independiente (Docker container).
2. **Comunicación**: REST para APIs externas, gRPC para comunicación interna entre toquens.
3. **Extensibilidad**: Nuevos toquens se registran via registry.json.

json

// Ejemplo registry.json

{

"toquens": [

{

"name": "pdf",

"endpoint": "http://pdf-toquen:8000",

"file\_types": ["application/pdf"]

}

]

}

**5. Validación con el Equipo**

**Checklist de Validación**

1. **Revisión Arquitectural**:
   * ¿Cada módulo tiene dependencias claras?
   * ¿Los protocolos de comunicación son seguros (HTTPS/gRPC+TLS)?
2. **Pruebas de Concepto**:
   * Toquen PDF sanitiza correctamente archivos de prueba.
   * OWASP ZAP detecta vulnerabilidades en entorno controlado (DVWA).
3. **Feedback del Equipo**:

plaintext

Preguntas clave:

- ¿Algún módulo es demasiado complejo para el tiempo disponible?

- ¿Tenemos acceso a todas las APIs necesarias (AWS, ZAP, etc.)?

**🔧 Mejoras Adicionales Propuestas**

**Para Comunicación entre Toquens**

* **Usar Apache Kafka**: Para colas de procesamiento asíncrono.

python

from kafka import KafkaProducer

producer = KafkaProducer(bootstrap\_servers='kafka:9092')

producer.send('toquen\_events', b'PDF recibido para analizar')

**Para Seguridad**

* **Firmar Archivos Procesados**:

python

from cryptography.hazmat.primitives import hashes

digest = hashes.Hash(hashes.SHA256())

digest.update(file\_bytes)

file\_hash = digest.finalize().hex()

**Para el Dashboard**

* **Visualización con Grafana** (si hay tiempo):

yaml

# docker-compose.yml

grafana:

image: grafana/grafana

ports:

- "3000:3000"

**📌 Conclusión**

Este documento proporciona:

1. **Módulos bien definidos** con responsabilidades acotadas.
2. **Protocolos estandarizados** (REST/gRPC) para comunicación.
3. **Asignación clara de tareas** según habilidades del equipo.
4. **Arquitectura validable** mediante PoCs y feedback.

**Siguientes pasos**:

* Realizar una reunión de alineación técnica con el equipo.
* Empezar con el setup del entorno de desarrollo (Docker + GitHub).