**Proyecto: Plataforma de Seguridad Unificada (PSU) con "Toquen" de Antesala**

Equipo: 4 personas | Horas totales: 320h | Duración estimada: 8-10 semanas

1. Valor Añadido y Posicionamiento en el Mercado

🔍 Problema Actual

Las herramientas de seguridad están fragmentadas (ej: Nessus para vulnerabilidades, Burp Suite para web, AWS Inspector para cloud).

Falta un sistema que analice archivos antes de almacenarlos ("toquen de antesala") y los sanitice.

Subdominios expuestos en buscadores generan riesgos.

🚀 Nuestra Propuesta de Valor

Competencia Nuestra Ventaja

Herramientas separadas (OWASP ZAP, Qualys) Centralización en una sola plataforma

Soluciones que solo detectan Mitigación automática con "toquen"

Subdominios indexables en Google Protección activa (como Office365)

Escaneo tradicional Pre-análisis de archivos antes de guardar

👉 Posicionamiento:

"El guardián previo al almacenamiento" (evita infecciones por archivos maliciosos).

"Centralización inteligente de seguridad" (web + cloud + mitigación).

"Subdominios invisibles para Google" (sin indexación no deseada).

2. Alcance Realista (MVP en 320h)

📌 Módulos Prioritarios

Módulo Horas Estimadas Responsable

Toquen de Antesala (análisis previo a almacenamiento) 100h Dev 1 + Dev 2

Escáner Web Básico (SQLi, XSS, CORS misconfig) 80h Dev 3

Protección de Subdominios (no-indexación + auth) 60h Dev 4

Dashboard Unificado (visualización de resultados) 50h Dev 1

Integración Cloud Básica (AWS/Azure Security Center) 30h Dev 2

📌 Features Descartadas (por tiempo)

❌ Análisis avanzado de malware (requiere más tiempo).

❌ Auto-remediación completa (solo mitigaciones básicas).

❌ Soporte multi-cloud completo (solo AWS/Azure inicial).

3. Arquitectura Modular

text

1. \*\*Frontend (Dashboard)\*\* → React/Vue (50h)

2. \*\*Backend (API)\*\* → Python/FastAPI (60h)

3. \*\*Toquen Manager\*\* →

- \*\*Toquen Maestro\*\* (identifica tipo de archivo)

- \*\*Toquen Especializados\*\* (PDF, EXE, Office)

- \*\*Toquen Sanitizador\*\* (elimina macros/scripts peligrosos)

4. \*\*Escáner Web\*\* → Integración con OWASP ZAP API

5. \*\*Protección Subdominios\*\* →

- robots.txt + X-Robots-Tag

- Autenticación básica (OAuth2 opcional)

4. Plan de Trabajo Colaborativo

🔹 Semana 1-2: Diseño y Setup

Definir arquitectura del "toquen".

Configurar API y frontend básico.

🔹 Semana 3-5: Desarrollo Módulos Clave

Implementar toquen de antesala (análisis de archivos).

Escáner web básico (SQLi/XSS).

🔹 Semana 6-7: Integración y Protección Subdominios

Conectar dashboard con escáner.

Implementar no-indexación + auth básica.

🔹 Semana 8-10: Pruebas y Documentación

Testear sanitización de archivos.

Ajustar protección subdominios.

5. Entregables Finales

✅ Aplicación funcional con:

Toquen de antesala (evita almacenar archivos peligrosos).

Escaneo web básico + mitigaciones.

Subdominios no indexables en Google.

Dashboard unificado.

✅ Documentación técnica (arquitectura, flujo de toquen).

✅ Video demostrativo (muestra valor único).

6. ¿Qué Priorizar si Falta Tiempo?

Toquen de antesala (nuestro diferencial).

Protección subdominios (segundo diferencial).

Dashboard (para demostrar resultados).

🎯 Conclusión

Este MVP demuestra el valor único de la plataforma:

✔ Previene infecciones (toquen de antesala).

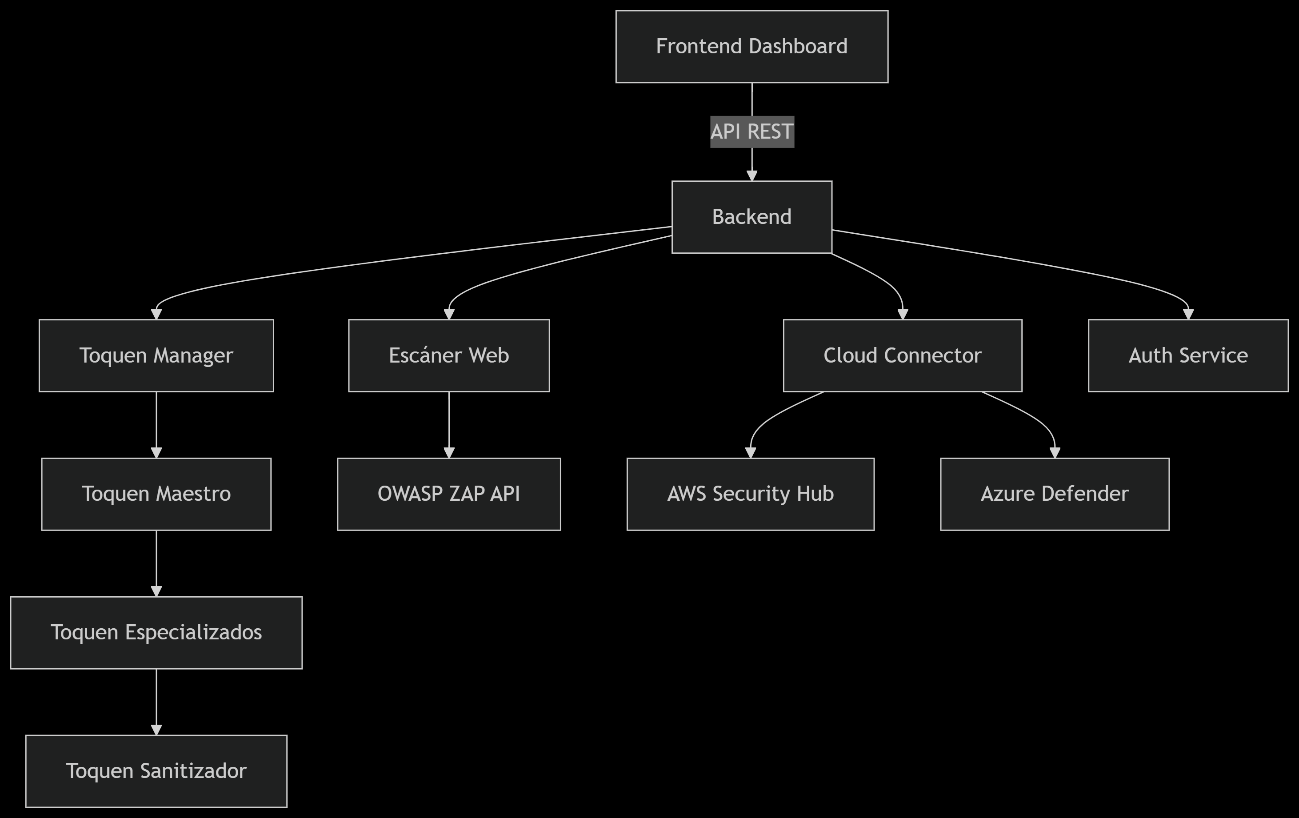
✔ Centraliza seguridad web + cloud.

✔ Oculta subdominios críticos.

**Arquitectura Técnica Detallada**

**Plataforma de Seguridad Unificada (PSU) con Toquen de Antesala**

**1. Diagrama de Arquitectura General**



flowchart TB

A[Frontend Dashboard] -->|API REST| B[Backend]

B --> C[Toquen Manager]

C --> D[Toquen Maestro]

D --> E[Toquen Especializados]

E --> F[Toquen Sanitizador]

B --> G[Escáner Web]

B --> H[Cloud Connector]

B --> I[Auth Service]

G --> J[OWASP ZAP API]

H --> K[AWS Security Hub]

H --> L[Azure Defender]

**2. Componentes Clave**

**🔹 Frontend (Dashboard)**

* **Tecnología:** React + TailwindCSS
* **Funciones:**
  + Visualización de vulnerabilidades
  + Configuración del "toquen"
  + Gestión de subdominios (no-indexación)
* **Comunicación:** API REST (JSON)

**🔹 Backend (API Core)**

* **Tecnología:** Python (FastAPI) + PostgreSQL
* **Endpoints clave:**
  + /api/scan (inicia escaneo web/cloud)
  + /api/token-process (gestión de toquen)
  + /api/subdomains (protección subdominios)

**🔹 Toquen Manager**

python

class ToquenMaestro:

def \_\_init\_\_(self, file):

self.file = file

self.file\_type = self.\_detect\_type() # Magic Numbers + ML básico

def \_detect\_type(self):

# Implementar con libmagic o custom rules

pass

def route\_to\_specialized\_token(self):

if self.file\_type == "PDF":

return PDFToquen(self.file)

elif self.file\_type == "EXE":

return EXEToquen(self.file)

**Flujo del Toquen**

1. **Entrada:** Archivo subido por usuario/app.
2. **Toquen Maestro:** Identifica tipo (PDF, EXE, ZIP).
3. **Toquen Especializado:**
   * **PDFToquen:** Busca JS embebido/macros.
   * **EXEToquen:** Verifica firma digital/sandbox básico.
4. **Toquen Sanitizador:**
   * Elimina componentes sospechosos (ej: macros de Office).
   * Registra acción en log de seguridad.

**3. Módulo de Escaneo Web**

* **Integración:** OWASP ZAP (API REST)
* **Vulnerabilidades detectadas:**
  + SQL Injection (SQLi)
  + Cross-Site Scripting (XSS)
  + Cabeceras CORS mal configuradas
* **Flujo:**

python

def scan\_website(url):

zap = ZAPv2(apikey="API\_KEY")

scan\_id = zap.spider.scan(url)

results = zap.core.alerts()

return generate\_mitigation(results) # Sugiere soluciones

**4. Protección de Subdominios**

**Técnicas Implementadas**

| **Método** | **Implementación** |
| --- | --- |
| **robots.txt** | Bloquea rastreadores en /subdominios/ |
| **X-Robots-Tag** | Cabecera HTTP noindex, nofollow |
| **Autenticación Básica** | Middleware en backend (API Key/OAuth2) |

**Código Ejemplo (FastAPI Middleware)**

python

@app.middleware("http")

async def protect\_subdomains(request: Request, call\_next):

if "subdominio-secreto" in request.url.path:

if not valid\_api\_key(request.headers.get("X-API-Key")):

return JSONResponse({"error": "Acceso no autorizado"}, 403)

return await call\_next(request)

**5. Cloud Connector (AWS/Azure)**

* **Tecnología:** Boto3 (AWS) / Azure SDK
* **Funciones:**
  + Revisa configuraciones inseguras (S3 públicos, NSG abiertos).
  + Prioriza hallazgos con CVSS > 7.0.

python

def check\_aws\_security():

client = boto3.client('securityhub')

findings = client.get\_findings(SeverityThreshold=7)

return parse\_cloud\_findings(findings)

**6. Stack Tecnológico**

| **Capa** | **Tecnologías** |
| --- | --- |
| **Frontend** | React, Chart.js, TailwindCSS |
| **Backend** | FastAPI (Python), Uvicorn |
| **Base de Datos** | PostgreSQL (SQLAlchemy) |
| **Seguridad** | OWASP ZAP, libmagic, Sandboxing básico |
| **DevOps** | Docker, GitHub Actions |

**7. Despliegue y Escalabilidad**

* **Contenedores:** Docker para Toquen Manager y Backend.
* **Escalabilidad:**
  + Toquen workers en Kubernetes (futuro).
  + Redis para colas de tareas (escaneos largos).

dockerfile

# Ejemplo Dockerfile (Backend)

FROM python:3.9

COPY requirements.txt .

RUN pip install -r requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0"]

**8. Pruebas y Seguridad**

* **Unitarias:** Pytest (cobertura >80%).
* **Pentesting:** OWASP ZAP contra el propio dashboard.
* **Hardening:**
  + Todos los toquen se ejecutan en sandbox (Firejail).
  + API Keys con rotación automática.

**9. ¿Qué Falta por Definir?**

1. **Modelo de datos exacto** para PostgreSQL.
2. **Reglas de sanitización** por tipo de archivo.
3. **Priorización de vulnerabilidades** en el dashboard.

**Stack Tecnológico Exacto para el Proyecto PSU**

1. Desarrollo del "Toquen de Antesala" (Núcleo del Proyecto)

Función Herramientas Elegidas Razón

Procesamiento de Archivos Python + python-magic (detección de tipos) + PyMuPDF (PDFs) + oletools (Office) Ligero, amplia compatibilidad con análisis de archivos

Sanitización - PDFs: pdf-redact-tools (elimina JS/embeds)

- Office: oletools (macros) Soluciones OSS probadas en entornos forenses

Despliegue Docker (contenedorización de cada "toquen") Aislamiento de procesos y escalabilidad

python

# Ejemplo de código para el Toquen Maestro

import magic

from pdf\_token import PDFSanitizer

from office\_token import OfficeAnalyzer

class MasterToken:

def \_\_init\_\_(self, file\_bytes):

self.file\_type = magic.from\_buffer(file\_bytes, mime=True)

def route(self):

if self.file\_type == "application/pdf":

return PDFSanitizer().sanitize()

elif "office" in self.file\_type:

return OfficeAnalyzer().remove\_macros()

2. Escaneo de Vulnerabilidades Web/Cloud

|  |  |  |
| --- | --- | --- |
| Módulo | Herramientas/APIs | Integración |
| Escáner Web | OWASP ZAP (API REST) + Nuclei (plantillas personalizadas) | Automatizable, cubre OWASP Top 10 |
| Cloud (AWS/Azure) | - AWS: boto3 (Security Hub) + Scout Suite  - Azure: azure-mgmt-security | Acceso directo a reportes de seguridad nativos |
| API Backend | FastAPI (Python) + Swagger/OpenAPI | Rápido desarrollo y documentación automática |

bash

# Comando para ejecutar OWASP ZAP automáticamente

docker run -t owasp/zap2docker zap-baseline.py -t https://example.com -r report.html

3. Frontend y Visualización

Componente Tecnología Ventaja

Dashboard Vue.js (Composition API) + Tailwind CSS Reactividad + diseño modular

Grágicos Apache ECharts (alternativa ligera a D3.js) Visualización interactiva de vulnerabilidades

Autenticación Auth0 (gratis para <7k usuarios) o Firebase Auth Solución lista para producción

javascript

// Ejemplo de llamada API desde Vue

fetch('https://api.psu/zap-scan', {

method: 'POST',

body: JSON.stringify({ url: targetUrl })

})

.then(response => response.json())

.then(data => this.vulns = data);

4. Protección de Subdominios

Técnica Implementación Exacta

No Indexación - X-Robots-Tag: noindex en cabeceras HTTP

- robots.txt con Disallow: /

Autenticación Nginx Basic Auth o Keycloak (OSS)

DNS Privado Cloudflare (proxy DNS) o AWS Route53 (zonas privadas)

nginx

# Configuración Nginx para autenticación y no-indexación

server {

listen 80;

server\_name subdominio.secreto.com;

add\_header X-Robots-Tag "noindex, nofollow";

auth\_basic "Restringido";

auth\_basic\_user\_file /etc/nginx/.htpasswd;

location / {

proxy\_pass http://app\_interna;

}

}

5. Entorno de Desarrollo y Colaboración

Área Herramientas

Control Versiones Git + GitHub (Projects para Kanban)

CI/CD GitHub Actions (para tests automáticos)

Documentación MkDocs (Markdown) + Swagger (APIs)

Comunicación Discord/Slack + Miro (diagramas arquitectura)

6. Alternativas por si Fallan las Herramientas Principales

Análisis de archivos:

Alternativa a python-magic: filetype (Python)

Escaneo web:

Si OWASP ZAP es complejo: Nikto (más simple pero menos features)

Dashboard:

Si Vue.js requiere mucha curva de aprendizaje: Svelte (más sencillo)

🔍 ¿Por qué Este Stack?

✔ Python → Ideal para procesamiento rápido de archivos (toquen) e integración con herramientas de seguridad.

✔ Vue.js → Menos overhead que React para un equipo pequeño.

✔ OWASP ZAP → El escáner web más usado (con API bien documentada).

✔ Docker → Permite aislar cada "toquen" y escalar en futuro.