**Proyecto: Plataforma Integral de Seguridad Informática**

**Descripción General**

**Aplicación destinada a centralizar y ordenar herramientas de seguridad informática para el análisis de vulnerabilidades en páginas web y entornos cloud. El sistema no solo detecta riesgos, sino que también proporciona información práctica para mitigarlos y se apoya en una arquitectura innovadora basada en “toquen de antesala”.**

**Clave Características**

* **Centralización de herramientas: Permite gestionar desde una sola interfaz múltiplos módulos de análisis de seguridad (escáner web, análisis cloud, escaneo de malware, etc.).**
* **Arquitectura modular con “toquen de antesala”:**
  + **Cada “toquen de antesala” analiza la información y los archivos antes de que se almacenen en disco.**
  + **Determina el tipo de archivo y lo dirige al “toquen” específico encerrado del sistema/aplicación correspondiente.**
  + **Antes del alojamiento definitivo, el archivo es tratado para asegurar que es inocuo.**
  + **Los “toquen” se generan dinámicamente durante la instalación y colaboración entre sí para maximizar la protección.**
* **Mitigación proactiva: Tras la detección de vulnerabilidades, la plataforma de recepción guías y acciones sugeridas para mitigarlas.**
* **Valor añadido/Posicionamiento:**
  + **Foco en la prevención activa gracias al análisis anterior (“antesala”).**
  + **Modularidad y extensibilidad para integrar nuevas herramientas facilmente.**
  + **Orientación a entornos multinube y web.**
  + **Facilidad de uso incluido para equipos no expertos.**

**Primeros Pasos del Proyecto**

1. **Definir el MVP (Producto Mínimo Viable):**
   * **Módulo centralizador (panel de control)**
   * **Sistema básico de “toquen de antesala” para análisis de archivos entrantes**
   * **Integración con al menos una herramienta de análisis web y una nube**
   * **Generación de informes y recomendaciones de mitigación**
2. **Análisis de mercado y benchmarking:**
   * **Comparar con plataformas como Nessus, Qualys, Burp Suite, etc.**
   * **Identificar el valor diferencial: análisis proactivo, modularidad, colaboración de “toquen”, integración sensacila de nuevas fuentes y sistemas.**
3. **Diseño de la arquitectura:**
   * **Diagrama de módulos y fluido de información entre “toquen de antesala”, “toquen específicos” y el núcloo.**
   * **Especificar API interna para comunicación entre “toquen”.**
4. **Backlog inicial de tareas:**
   * **Especificar el protocolo de análisis del “toquen de anteala”**
   * **Implementar un prototipo sencillo de “toquen de antesala”**
   * **Web integral primer escáner**
   * **Documentar los requisitos para integrar de nuevas herramientas (API de “toquen”)**
   * **Crear maquetas del tablero**

**Notas de Innovación**

* **El sistema de “toquen de antesala” permite analizar y segmentar datos antes de que accedan al almacenamiento local, lo que minimiza el riesgo de infecciones o ataques persistentes.**
* **Cada “toquen” es autónomo pero puede pedir ayuda a otros para combatir amenazas completas, fomentando una defensa colaborativa.**
* **El enfoque en la mitigación directa y las guías prácticas diferencia la plataforma de otras que solo reportan problemas.**
* **La arquitectura modular facilita la evolución del producto y su adaptación a nuevos tipos de amenazas o entornos tecnológicos.**

**Próximos pasos**

1. **Validar hipótesis de valor añadido con potenciales usuarios (CISOs, equipos de IT, pentesters).**
2. **Priorizar las tareas del backlog y definir hoja de ruta.**
3. **Establecer criterios de éxito para el MVP.**

**Especificaciones Técnicas Mínimas por Módulo**

**1. Panel Centralizado (Frontend)**

**Tecnología sugerida: Reaccionar (alternativa: Vue) Requisitos mínimos:**

* **Pantalla de inicio con resumen de análisis recientes y su estado.**
* **Formulario para lanzar nuevos análisis (selección de tipo: web/cloud, subida de archivo si aplica).**
* **Sección de resultados: lista de vulnerabilidades detectadas, nivel de riesgo y recomendaciones de mitigación.**
* **Visualización de logs/salidas de herramientas de análisis.**
* **Comunicación con backend vía API REST.**
* **Diseño responsive básico.**

**2. Backend Core + API**

**Tecnología sugerida: Python (FastaPI/Flask), Node.js (Express) Requisitos mínimos:**

* **API REST para:**
  + **Lanzar análisis (recibe tipo, archivo, URL, etc.)**
  + **Consultar resultados y estado de análisis.**
  + **Obtenedor recomendaciones asociadas.**
* **Organizador que gestione la comunicación entre frontend, “toquen de antesala” y los módulos de análisis.**
* **Gestión de usuarios opcional (si hay tiempo).**
* **Registros de actividad.**

**3. Módulo “Toquen de Antesala”**

**Tecnología azucarida: Python (microservicio o clase independiente) Requisitos míos:**

* **Intercepta archivos antes de alojamiento o análisis.**
* **Analiza tipo de archivo (por extensión/MIME).**
* **Redirige el archivo al módulo “toquen” específico segundo el tipo (web, nube, script, binario, etc.).**
* **Realiza inspección mía para detectar elementos sospechosos (simulación: escándalo por firma, tamaño, nombre, etc.).**
* **Marca como “inocuo” o “sospechoso” y permite/deniega paso.**
* **Registra eventos de análisis.**

**4. Integración de Herramientas de Análisis**

**Tecnología azucarida: Scripts de Python, wrappers CLI para herramientos externos (ej: OWASP ZAP, trivy, etc.) Requisitos míos:**

* **Módulos independientes para cada tipo de análisis:**
  + **Web: Lanza escaneo sobre URL objetivo, recoge resultados.**
  + **Nube: Simulación de cheque básico de configuración/nube (ej: S3 abierto).**
* **Proceso y normalización la salida a un formato común (JSON).**
* **Retorna resultados al backend core.**

**5. Motor de Recomendaciones**

**Tecnología azucarida: Python (módulo o clase) Requisitos míos:**

* **Asociación vulnerabilidades detectadas con plantas de mitigación.**
* **Desarrolla una interfaz un listado de acciones sugeridas y combina una documentación.**
* **Permite ampliar la base de conocimientos facilmente (estructura de plantas en archivos YAML/JSON).**

**6. Documentación y Soporte**

* **README claro para el despliegue local y explicación de arquitectura.**
* **Ejemplo de configuración y flujo de trabajo.**
* **Diagrama básico de arquitectura.**

**Notas de modularidad y colaboración:**

* **Cada módulo debe poder desarrollar y probar de forma independiente (API bien definidos).**
* **Estandarizar el intercambio de datos en formato JSON.**
* **Uso de control de versiones y ramas por módulo/feature.**

**Primer Sprint: MVP Inicial**

Objetivo

Desarrollar un MVP funcional con:

Un “toquen de anteala” operativo.

Integración de una herramienta de escaneo web (por ejemplo, OWASP ZAP o similar, incluido en modo simulado/mock).

Un dashboard básico para lanzar análisis, subir archivos y visualizar resultados.

Alcance y entregables

1. Módulo “toquen de antesala”

Microservicio o clase en Python.

API REST para recibir archivos.

Detectar tipo de archivo (por extensión/MIME).

Redirigir el archivo al módulo correspondiente (por ahora, solo web).

Marcar archivo como “inocuo” o “sospechoso” (reglas simples: extensión, tamaño, nombre, contenido neutral).

Retornar resultado al backend.

Registro de eventos (log básico en consola o archivo).

2. Integración de herramientas de escaneo web

Script/wrapper que reciba una URL y lance un escándalo básico (puede ser simulación si no se dispone de la herramienta real).

Recoger resultados principales (ejemplo: “XSS detectado”, “Sin vulnerabilidades”, etc.).

Estandarizar resultado en JSON.

Endpoint REST para lanzar el análisis web desde backend.

3. Panel básico (Frontend)

Interfaz web sencilla (Reaccionar, Vue, etc.).

Formulario para:

Subir archivo (que se enviará al “toquen de antesala”).

Lanzar análisis web (ingreso de URL).

Visualización de resultados de análisis: tabla/lista de vulnerabilidades y estado del análisis.

Integración con backend vía API REST.

Diseño mío, pero funcional.

División de tareas sugerida

Persona 1: “Toquen de antesala” (servicio, API, lógica de clasificación)

Persona 2: Integración herramienta web (script, wrapper, API, simulación si es necesario)

Persona 3: Núcleo del backend (orquestador, endpoints REST, conexión módulos)

Persona 4: Frontend/panel de control (formulario, resultados, API de conexión)

Criterios de éxito para el sprint

El usuario puede subir un archivo y recibir un resultado (“inocuo” o “espercoso”).

El usuario puede lanzar un análisis web de una URL y ver los resultados en el dashboard.

Comunicación fluida entre frontend, backend, “toquen de antesala” y módulo web.

Código modular y documentado (README, comentarios esenciales).

Entregables ejecutables localmente (manual de compras sencillo en README).

Pasos siguientes

Definir puntos finales y estructura JSON para la comunicación entre módulos.

Crear repositorio y ramas por módulo.

Realizar reuniones cortas de seguimiento y sincronización.