**Plan del Proyecto MVP de Aplicación de Ciberseguridad**

**Resumen del Proyecto**

Desarrollar una aplicación web para un escaneo básico de vulnerabilidades en páginas web, con un sistema simplificado de "toquen de antesala" para analizar archivos subidos antes de procesarlos. El MVP se centra en funcionalidades esenciales, diseño modular e integración de herramientas existentes para ajustarse a un período de desarrollo de 320 horas por un equipo de cuatro personas.

**Objetivos**

* Implementar un escáner básico de vulnerabilidades web usando una herramienta existente (OWASP ZAP).
* Desarrollar un toquen de antesala simplificado para analizar archivos en busca de amenazas básicas.
* Proporcionar una interfaz amigable con resultados de escaneos y sugerencias de mitigación básicas.
* Asegurar modularidad para futuras expansiones.

**Alcance**

**Incluido**

* Escaneo de vulnerabilidades web para problemas comunes (por ejemplo, XSS, CSRF).
* Toquen de antesala para análisis básico de archivos (por ejemplo, validación de tipo de archivo, comprobaciones simples de malware).
* Panel web para iniciar escaneos, subir archivos y ver resultados.
* Recomendaciones de mitigación básicas para vulnerabilidades detectadas.

**Excluido (para el MVP)**

* Escaneo de entornos cloud.
* Toquens especializados para diferentes sistemas.
* Neutralización avanzada de amenazas o sistema colaborativo de toquens.
* Integración de inteligencia de amenazas en tiempo real.

**Funcionalidades Clave**

1. **Escáner de Vulnerabilidades Web**:
   * Integra OWASP ZAP vía API para escanear URLs proporcionadas por el usuario.
   * Detecta vulnerabilidades comunes (por ejemplo, XSS, CSRF, cabeceras inseguras).
   * Genera un informe con los hallazgos.
2. **Toquen de Antesala**:
   * Analiza archivos subidos para validar tipos MIME (por ejemplo, PDF, PNG) y firmas de malware básicas usando ClamAV.
   * Pone en cuarentena o rechaza archivos maliciosos antes de su almacenamiento.
   * Registra resultados de análisis para revisión del usuario.
3. **Panel de Control**:
   * Interfaz web para iniciar escaneos de URLs y subir archivos.
   * Muestra resultados de escaneos y registros de análisis de archivos.
   * Proporciona consejos de mitigación estáticos (por ejemplo, "Habilitar CSP para mitigar XSS").

**Arquitectura Técnica**

* **Frontend**: React con Tailwind CSS (alojado en CDN) para un panel simple.
* **Backend**: Python (FastAPI) para puntos finales de API y lógica de toquens.
* **Base de Datos**: SQLite para almacenar resultados de escaneos y metadatos de archivos (ligero para el MVP).
* **Herramientas de Seguridad**: OWASP ZAP (escaneo web), ClamAV (escaneo de archivos).
* **Despliegue**: Servidor local para la demo (sin contenerización para el MVP).

**Roles del Equipo y Distribución de Tareas**

Horas totales: 320 (80 horas por persona). Equipo de 4: Desarrollador A (Frontend), Desarrollador B (Backend), Desarrollador C (Herramientas de Seguridad), Desarrollador D (Pruebas/Documentación).

**Desarrollador A: Frontend (80 horas)**

* Diseñar e implementar el panel de React (40 horas).
* Integrar Tailwind CSS para estilos (10 horas).
* Conectar frontend con APIs de backend (20 horas).
* Pruebas de usuario y refinamiento de la interfaz (10 horas).

**Desarrollador B: Backend (80 horas)**

* Configurar servidor FastAPI y puntos finales de API (25 horas).
* Implementar lógica de toquen de antesala para análisis de archivos (30 horas).
* Configurar base de datos SQLite y esquema (15 horas).
* Asistir con pruebas de integración de APIs (10 horas).

**Desarrollador C: Herramientas de Seguridad (80 horas)**

* Integrar OWASP ZAP para escaneo web (30 horas).
* Integrar ClamAV para escaneo de archivos (25 horas).
* Desarrollar lógica básica de recomendaciones de mitigación (15 horas).
* Apoyar pruebas de funcionalidades de seguridad (10 horas).

**Desarrollador D: Pruebas/Documentación (80 horas)**

* Escribir pruebas unitarias e integrales (30 horas).
* Realizar pruebas de extremo a extremo (20 horas).
* Crear documentación de usuario e informe del proyecto (20 horas).
* Coordinar entregables del equipo y configuración de la demo (10 horas).

**Plan de Desarrollo**

**Fase 1: Configuración y Diseño (Semana 1, 40 horas)**

* **Tareas**:
  + Configurar repositorio en GitHub y estructura del proyecto (Dev B, 5 horas).
  + Diseñar wireframes de la interfaz (Dev A, 10 horas).
  + Definir puntos finales de API y esquema de base de datos (Dev B, 10 horas).
  + Investigar integración de OWASP ZAP/ClamAV (Dev C, 15 horas).
* **Entregables**: Configuración del proyecto, wireframes, diseño de API.

**Fase 2: Desarrollo Principal (Semanas 2-3, 160 horas)**

* **Tareas**:
  + Construir panel de React (Dev A, 40 horas).
  + Desarrollar backend FastAPI y toquen de antesala (Dev B, 45 horas).
  + Integrar OWASP ZAP y ClamAV (Dev C, 45 horas).
  + Escribir pruebas iniciales y documentación (Dev D, 30 horas).
* **Entregables**: Panel funcional, APIs de backend, herramientas integradas.

**Fase 3: Pruebas y Refinamiento (Semana 4, 100 horas)**

* **Tareas**:
  + Conectar frontend con APIs de backend (Dev A, 20 horas).
  + Probar y depurar lógica de backend/toquen (Dev B, 20 horas).
  + Validar salidas de herramientas de seguridad (Dev C, 20 horas).
  + Realizar pruebas de extremo a extremo y finalizar documentación (Dev D, 40 horas).
* **Entregables**: MVP completamente probado, guía de usuario, informe del proyecto.

**Fase 4: Preparación de la Demo (Semana 5, 20 horas)**

* **Tareas**:
  + Configurar entorno de demo local (Dev B, 5 horas).
  + Preparar guion y presentación de la demo (Dev D, 10 horas).
  + Ensayar la demo (Todos, 5 horas).
* **Entregables**: Aplicación lista para demo, presentación.

**Fragmentos de Código de Ejemplo**

**Toquen de Antesala (Backend, Python/FastAPI)**

from fastapi import FastAPI, UploadFile

from clamav import scan\_file

app = FastAPI()

class ToquenAntesala:

async def analizar\_archivo(self, archivo: UploadFile):

tipos\_permitidos = ["application/pdf", "image/png"]

if archivo.content\_type not in tipos\_permitidos:

return {"estado": "rechazado", "razon": "Tipo de archivo inválido"}

resultado\_escaneo = scan\_file(await archivo.read())

return {"estado": "seguro" if resultado\_escaneo["limpio"] else "en cuarentena"}

@app.post("/subir")

async def subir\_archivo(archivo: UploadFile):

toquen = ToquenAntesala()

resultado = await toquen.analizar\_archivo(archivo)

return resultado

**Componente del Panel (Frontend, React)**

import React, { useState } from 'react';

function Escaner() {

const [url, setUrl] = useState('');

const [resultados, setResultados] = useState(null);

const escanearUrl = async () => {

const respuesta = await fetch('/api/escanear', {

method: 'POST',

body: JSON.stringify({ url }),

headers: { 'Content-Type': 'application/json' },

});

setResultados(await respuesta.json());

};

return (

<div className="p-4">

<input

type="text"

value={url}

onChange={(e) => setUrl(e.target.value)}

className="border p-2"

placeholder="Ingresa URL"

/>

<button onClick={escanearUrl} className="bg-blue-500 text-white p-2 ml-2">

Escanear

</button>

{resultados && <pre>{JSON.stringify(resultados, null, 2)}</pre>}

</div>

);

}

**Recomendaciones de Mitigación**

* **XSS**: Usar Content Security Policy (CSP) y sanitizar entradas.
* **CSRF**: Implementar tokens anti-CSRF.
* **Archivos Maliciosos**: Restringir tipos de archivo y escanear con ClamAV.

**Riesgos y Mitigación**

* **Riesgo**: Tiempo limitado para integración de herramientas.
  + **Mitigación**: Usar APIs bien documentadas y centrarse en funcionalidad básica.
* **Riesgo**: Problemas de coordinación del equipo.
  + **Mitigación**: Usar GitHub para control de versiones y reuniones semanales de sincronización.
* **Riesgo**: Falsos positivos en escaneo de archivos.
  + **Mitigación**: Permitir revisión manual de archivos en cuarentena.

**Herramientas y Tecnologías**

* **Frontend**: React (CDN: <https://cdn.jsdelivr.net/npm/react>), Tailwind CSS (CDN).
* **Backend**: Python, FastAPI, SQLite.
* **Herramientas de Seguridad**: OWASP ZAP, ClamAV.
* **Control de Versiones**: GitHub.

**Próximos Pasos**

1. Asignar roles a los miembros del equipo (A, B, C, D).
2. Configurar repositorio en GitHub y entornos de desarrollo locales.
3. Programar reuniones semanales de sincronización para seguir el progreso.
4. Comenzar tareas de la Fase 1 (configuración y diseño).