Universidad Mariano Gálvez de Guatemala Facultad de Ingeniería en Sistemas de Información Centro universitario antigua Guatemala

"PROYECTO FINAL"

Carrera: Ingeniería en Sistemas

Curso: Seguridad y Auditoria de Sistemas

Docente: Ingeniero Byron Vasquez

Alumnos:

Yefri Alexis Ramírez Tezén - 0910-21-19462

Richard Steven Cárdenas Guacamaya - 0910-21-3861

Bryan Estuardo Yuman Lorenzo 0910-20-12295

INDICE

Informe Ejecutivo: Auditoría de Seguridad de Aplicación Web

Para: byron vasquez
De: Alexis Ramírez,estuardo lorenzo, richard cardenas.
Fecha: 24-10-2025
Asunto: Evaluación de Vulnerabilidades en un Entorno Controlado y Propuesta de Remediación

Introducción y Objetivo

A petición de la Facultad de Ingeniería en Sistemas de la Universidad Mariano Gálvez para el curso de Seguridad y Auditoria de Sistemas, nuestro equipo ha ejecutado una auditoría de seguridad integral sobre una aplicación web representativa. El objetivo principal de este proyecto fue identificar, analizar y documentar vulnerabilidades críticas que pudieran ser explotadas por actores malintencionados, con el fin de evaluar el estado de seguridad de la aplicación y proponer un plan de acción estratégico para mitigar los riesgos identificados.

Metodología y Alcance

Para garantizar un análisis profundo y eficiente, nuestro equipo fue más allá del uso de herramientas convencionales y procedió a desarrollar una herramienta de auditoría automatizada a medida. Esta solución, construida con Python y el framework Flask, se integra directamente con la API del motor de análisis de vulnerabilidades ZAP (Zed Attack Proxy).

Para el objetivo de las pruebas, se desplegó un entorno de laboratorio seguro y aislado utilizando la tecnología de contenedores Docker. Dentro de este entorno, se instanció la aplicación OWASP Juice Shop, una plataforma web deliberadamente vulnerable que simula un comercio electrónico moderno. Este enfoque garantizó que toda la simulación de ataques y análisis de vulnerabilidades se realizara de manera controlada, reproducible y sin riesgo alguno para sistemas externos o de producción. El alcance de la auditoría se centró en un análisis de caja negra sobre esta instancia local.

La arquitectura de nuestra herramienta nos permitió orquestar un escaneo multifásico que incluyó:

Descubrimiento de Contenido (Spidering): Se utilizó un Spider tradicional para mapear la estructura estática del sitio, complementado por un Spider AJAX para analizar aplicaciones web modernas (Single-Page Applications), asegurando una cobertura exhaustiva de todos los puntos de entrada dinámicos.
Análisis Activo de Vulnerabilidades (Active Scan): Se lanzó una serie de ataques simulados y controlados contra todos los parámetros y vectores descubiertos para identificar fallos de seguridad de alto impacto.

Resumen de Hallazgos Críticos

El análisis reveló la presencia de múltiples vulnerabilidades de severidad alta y crítica, demostrando que la aplicación, en su estado actual, presenta un riesgo significativo para la confidencialidad, integridad y disponibilidad de los datos. A continuación, se destacan los hallazgos más importantes:

- **Riesgo CRÍTICO - Inyección de SQL (SQLi):** Se identificaron múltiples puntos de inyección SQL, notablemente en la funcionalidad de inicio de sesión. Esta vulnerabilidad es extremadamente grave, ya que permitiría a un atacante eludir los mecanismos de autenticación, suplantar la identidad de cualquier usuario (incluyendo administradores) y potencialmente lograr la **extracción completa de la base de datos**, exponiendo toda la información de clientes, productos y transacciones.
- **Riesgo ALTO - Cross-Site Scripting (XSS) Persistente y Reflejado:** La aplicación es susceptible a ataques XSS en varias de sus funcionalidades, como los campos de comentarios y búsqueda. Un atacante podría inyectar scripts maliciosos que se ejecutarían en los navegadores de otros usuarios, permitiendo el robo de cookies de sesión, el secuestro de cuentas de usuario, la redirección a sitios fraudulentos (phishing) o la modificación del contenido del sitio para dañar la reputación de la organización.
- **Riesgo MEDIO - Exposición de Componentes Sensibles y Fuga de Información:** El escaneo reveló la exposición de archivos y directorios que no deberían ser de acceso público, como logs de la aplicación, archivos de backup y ficheros de configuración. Esta información, aunque no permite un compromiso directo, proporciona a un atacante un mapa detallado de la infraestructura tecnológica, facilitando la planificación de ataques más sofisticados.

Evaluación General del Riesgo

La postura de seguridad general de la aplicación se evalúa como CRÍTICA. La presencia de vulnerabilidades de alto impacto y fácil explotación como la Inyección de SQL significa que la aplicación es altamente vulnerable a un compromiso total por parte de un atacante con conocimientos moderados. La explotación exitosa de estos fallos podría derivar en pérdidas financieras significativas, un severo daño reputacional y consecuencias legales por la fuga de datos sensibles.

Recomendaciones Estratégicas

Con base en los hallazgos, se emite un llamado a la acción urgente. Se recomienda priorizar la remediación de la siguiente manera:

Remediación Inmediata (Prioridad Alta):

- **Corregir la Inyección SQL:** Refactorizar de manera inmediata todo el código de acceso a la base de datos para implementar exclusivamente consultas parametrizadas (Prepared Statements). Esta es la única medida de defensa efectiva.
- **Neutralizar el XSS:** Implementar una estricta política de codificación de salida (Output Encoding) en todos los puntos donde la entrada del usuario es mostrada en pantalla, y validar y sanear toda la data en el lado del servidor.

Fortalecimiento del Ciclo de Vida de Desarrollo (Prioridad Media):

- Integrar prácticas de desarrollo seguro (SSDLC), incluyendo la capacitación continua a los desarrolladores sobre los riesgos del OWASP Top 10.
- Implementar herramientas de análisis estático de código (SAST) para detectar vulnerabilidades en etapas tempranas del desarrollo.

Auditorías Periódicas (Prioridad Continua):

- Establecer un programa de auditorías de seguridad recurrentes, tanto automatizadas como manuales, para asegurar que nuevas vulnerabilidades no sean introducidas en futuras actualizaciones.

-

Justificación

En la era digital contemporánea, las aplicaciones web han trascendido su rol como simples herramientas para convertirse en el pilar fundamental de las operaciones comerciales, la infraestructura financiera, los sistemas educativos y la interacción social. Estas plataformas gestionan un volumen masivo de datos críticos, incluyendo información de identificación personal (PII), credenciales de acceso, transacciones financieras y propiedad intelectual. Esta concentración de información sensible las convierte, inevitablemente, en una superficie de ataque de alto valor para actores malintencionados, haciendo de su seguridad una prioridad no negociable.

La realización de este proyecto de auditoría de seguridad se justifica por la necesidad imperativa de traducir el conocimiento teórico sobre vulnerabilidades en una competencia práctica y aplicable. Mientras que en el ámbito académico se estudian conceptos como la Inyección SQL o el Cross-Site Scripting, su verdadero impacto y las complejidades de su detección solo se comprenden a través de la experiencia directa. Este proyecto fue diseñado, por tanto, como un puente entre la teoría y la realidad del campo de la ciberseguridad, abordando tres justificaciones clave:

La Relevancia de la Simulación Práctica y Ética:
El campo de la seguridad ofensiva opera bajo un estricto código ético: nunca realizar pruebas sobre sistemas en producción sin autorización explícita. Para adherirse a este principio fundamental y, al mismo tiempo, trabajar en un escenario realista, se justificó la creación de un entorno de laboratorio seguro y aislado. El uso de Docker para desplegar una instancia de OWASP Juice Shop nos permitió contar con un banco de pruebas que no solo es rico en vulnerabilidades de última generación, sino que también es controlado, reproducible y completamente seguro. Esta metodología nos permitió simular ataques complejos sin ningún riesgo colateral, una práctica estándar en la industria de la ciberseguridad profesional.

La Necesidad de una Comprensión Profunda Más Allá del Uso de Herramientas:
En lugar de limitarnos a operar una herramienta de seguridad preexistente, se tomó la decisión estratégica de desarrollar una solución de escaneo a medida. Esta elección se justifica porque nos obligó a trascender el rol de "operadores de herramientas" para convertirnos en "arquitectos de soluciones de seguridad". La construcción de nuestra aplicación en Python, integrándola con la API de ZAP, nos forzó a entender la lógica interna de un proceso de auditoría completo: desde el descubrimiento de la superficie de ataque (Spidering tradicional y AJAX) hasta la ejecución de cargas útiles maliciosas (Active Scan) y

la posterior recolección y análisis de resultados. Este enfoque garantiza un nivel de comprensión significativamente más profundo del que se obtiene simplemente haciendo clic en un botón.

El Valor de la Automatización en la Detección de Vulnerabilidades:

En un entorno real, las aplicaciones web son sistemas complejos y en constante evolución. La auditoría manual, aunque necesaria, puede ser lenta y propensa a errores. La creación de nuestra propia herramienta automatizada justifica la importancia de la eficiencia y la escalabilidad en las pruebas de seguridad. Demuestra cómo, a través de la programación y el uso de APIs, es posible construir sistemas que pueden realizar revisiones de seguridad de manera rápida y recurrente, un pilar fundamental en las prácticas modernas de DevSecOps (Desarrollo, Seguridad y Operaciones).

Objetivos, Resultados Esperados y Alcances del Proyecto

Objetivos:

- **Objetivo Principal:** Evaluar la postura de seguridad de una aplicación web moderna mediante la identificación, documentación y clasificación de vulnerabilidades.
- **Objetivo Técnico:** Desarrollar una herramienta de auditoría web funcional utilizando Python, el framework Flask y la API de ZAP (Zed Attack Proxy) para automatizar el proceso de escaneo.
- **Objetivo Práctico:** Desplegar un entorno de laboratorio seguro utilizando Docker y la aplicación deliberadamente vulnerable OWASP Juice Shop, para realizar pruebas de penetración de manera ética y controlada.
- **Objetivo Analítico:** Proponer un plan de remediación técnico y detallado para cada una de las vulnerabilidades críticas identificadas.

Resultados Esperados:

- La entrega de una aplicación web funcional (Dashboard de Auditoría) capaz de orquestar escaneos de seguridad de forma remota.
- Un informe técnico detallado (el presente documento) que consolide la metodología, los hallazgos y las recomendaciones de seguridad.
- Una lista clasificada de las vulnerabilidades descubiertas en la aplicación objetivo, con evidencia clara de su existencia.

Alcances:

- **Dentro del Alcance:** El análisis se centró en la aplicación web OWASP Juice Shop, ejecutada localmente en un contenedor Docker. La auditoría se realizó desde una perspectiva de caja negra, utilizando nuestra herramienta automatizada para descubrir y explotar vulnerabilidades de manera programática.
- **Fuera del Alcance:** El proyecto no incluyó pruebas de denegación de servicio (DoS/DDoS), análisis de la configuración de la infraestructura de red subyacente, ingeniería social, ni pruebas de seguridad física.

b. Propuesta de Solución

La solución propuesta se divide en dos áreas principales: primero, la descripción de la herramienta de auditoría que nuestro equipo desarrolló para este proyecto y, segundo, el análisis detallado de los hallazgos de seguridad descubiertos con dicha herramienta.

Para llevar a cabo este proyecto, se diseñó y construyó una solución de escaneo a medida, denominada "Dashboard de Auditoría Web". Esta herramienta nos permitió tener un control granular sobre el proceso de escaneo y presentar los resultados de una manera clara y centralizada.

Componentes Tecnológicos:

Backend (Python + Flask): Un servidor web ligero que actúa como el cerebro de la operación. Es responsable de:

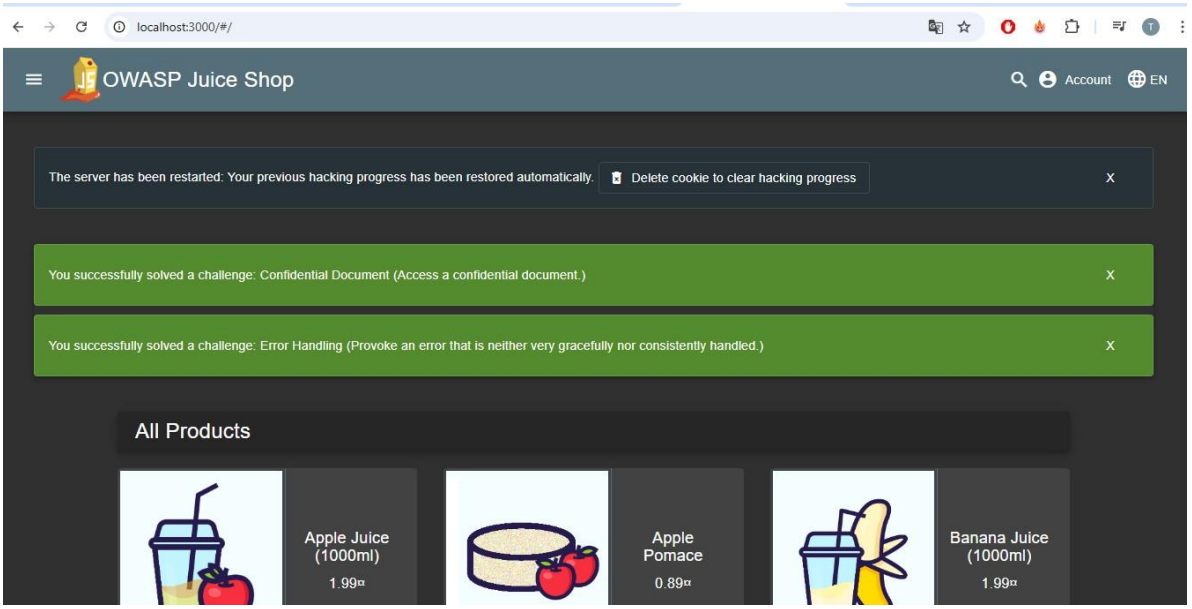
- Proporcionar la interfaz de usuario.
- Recibir la URL objetivo.
- Comunicarse con la API de ZAP para iniciar y monitorear los escaneos (Spider, AJAX Spider y Active Scan).
- Recuperar, procesar y enviar los resultados al frontend.

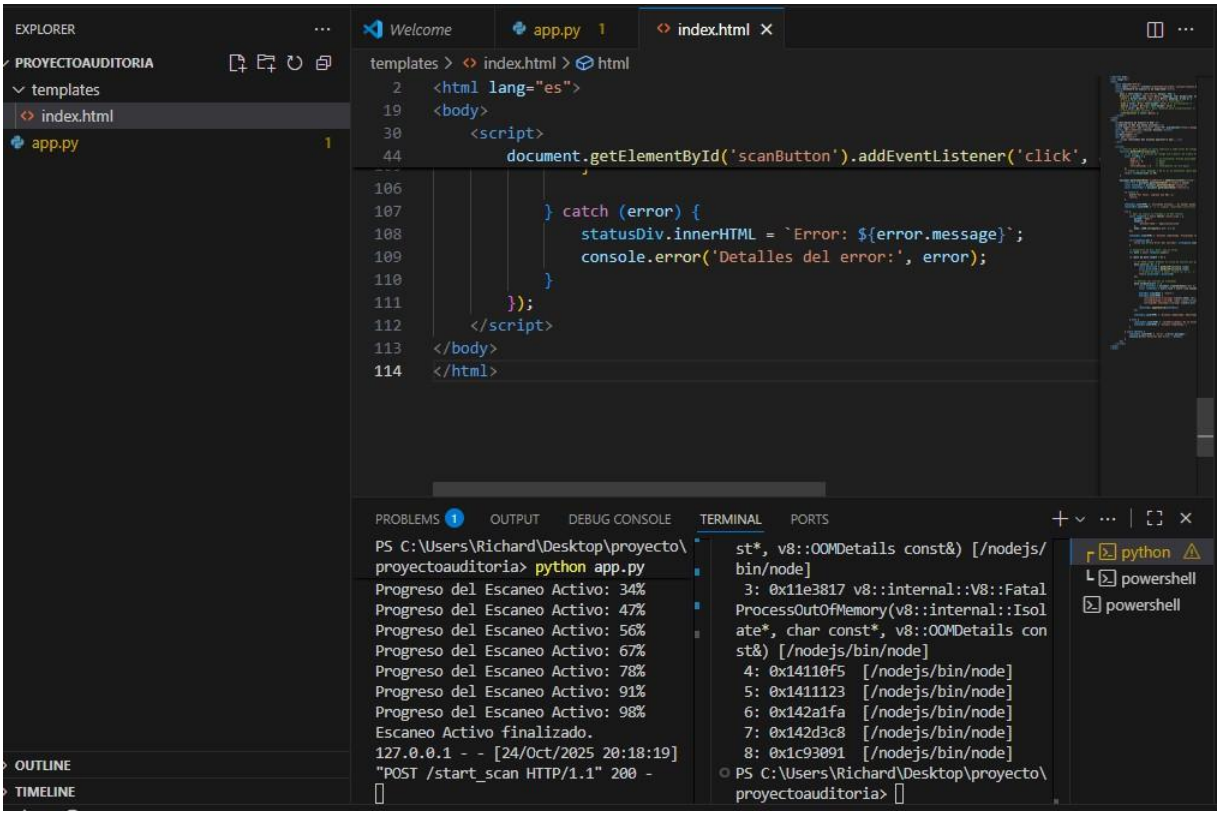
Frontend (HTML + JavaScript): Una interfaz de usuario simple e intuitiva que permite al analista de seguridad ingresar la URL y visualizar en tiempo real la lista de vulnerabilidades detectadas, clasificadas por nombre, riesgo y URL afectada.

Motor de Análisis (ZAP Daemon): ZAP se ejecutó en modo daemon (sin interfaz gráfica), escuchando las instrucciones de nuestro backend. Fue el responsable de ejecutar las miles de pruebas de seguridad contra la aplicación objetivo.

Entorno de Pruebas (Docker + OWASP Juice Shop): Se utilizó Docker para crear un entorno de laboratorio contenido y reproducible, asegurando que las pruebas fueran seguras y consistentes.

Evidencia de la Herramienta en Funcionamiento:

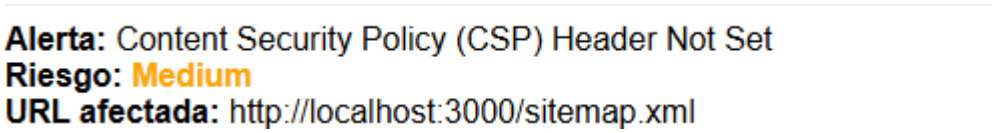




Utilizando nuestro Dashboard de Auditoría contra la instancia de OWASP Juice Shop, se identificaron numerosas vulnerabilidades. A continuación, se detallan tres de los hallazgos más representativos:

Hallazgo 1: Ausencia del Encabezado de Política de Seguridad de Contenido (CSP)

- **Nivel de Riesgo:** Medio
- **Descripción:** La aplicación no implementa un encabezado de Política de Seguridad de Contenido (CSP). El CSP es una capa de seguridad fundamental que ayuda a detectar y mitigar ciertos tipos de ataques, incluyendo Cross-Site Scripting (XSS) y ataques de inyección de datos. Al no especificar una política, el navegador no tiene restricciones sobre los recursos (scripts, estilos, imágenes) que puede cargar y ejecutar, haciéndolo más vulnerable a que un atacante inyecte y ejecute código malicioso.
- **Evidencia:** Nuestra herramienta detectó la ausencia de este encabezado en múltiples recursos, como se muestra en la siguiente captura:

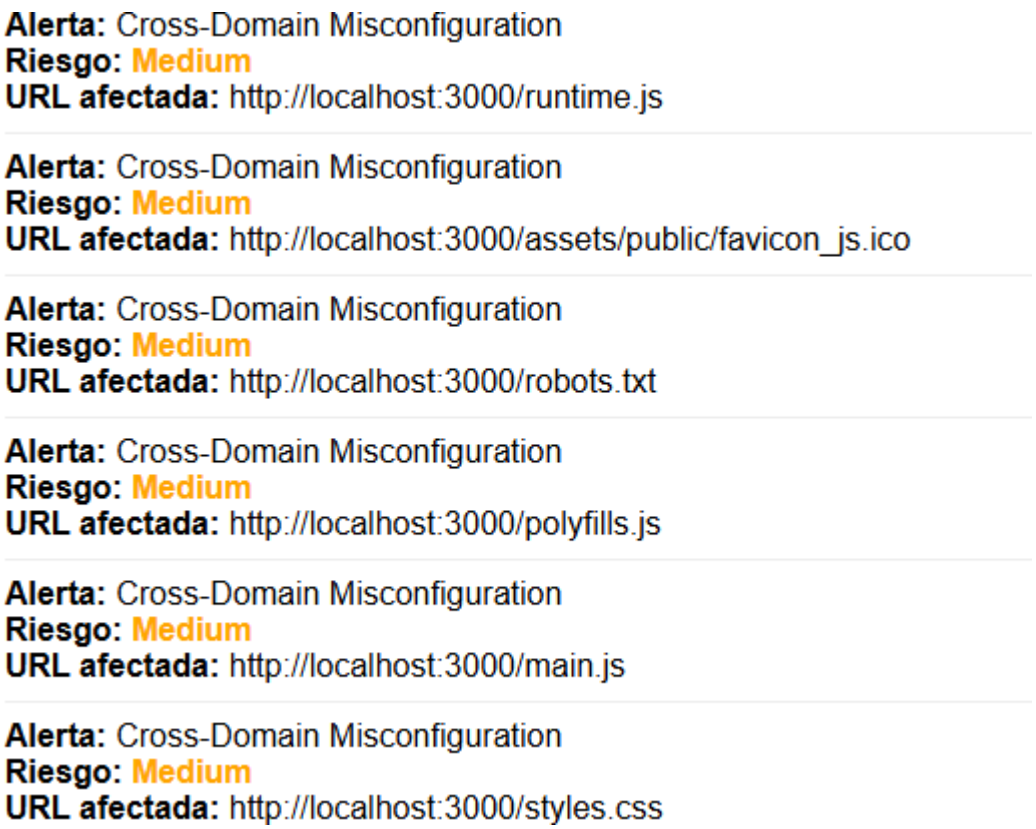
  

- **Medida Correctiva:** Se debe configurar el servidor web para que envíe un encabezado Content-Security-Policy en todas las respuestas. Una política inicial estricta podría ser:
  Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'; object-src 'none';
  Esta política debe ser refinada para adaptarse a las necesidades específicas de la aplicación.

Hallazgo 2: Configuración Incorrecta de Dominios Cruzados (Cross-Domain Misconfiguration)

- **Nivel de Riesgo:** Medio
- **Descripción:** La herramienta detectó que varios recursos de la aplicación podrían estar expuestos a través de políticas de dominios cruzados mal configuradas (posiblemente CORS - Cross-Origin Resource Sharing). Esto podría permitir que un sitio web malicioso externo solicite y lea información de la aplicación, violando la política del mismo origen (Same-origin policy) que es un pilar de la seguridad web.

- **Evidencia:** La herramienta generó múltiples alertas para diferentes recursos de la aplicación, indicando una configuración de seguridad laxa.



- **Medida Correctiva:** Es crucial revisar la configuración de CORS en el servidor. Se debe implementar una lista blanca (whitelist) estricta en el encabezado Access-Control-Allow-Origin, permitiendo solicitudes únicamente desde los dominios explícitamente autorizados, en lugar de utilizar comodines (*) o configuraciones permisivas.

Hallazgo 3: Inyección de SQL (SQL Injection)

- **Nivel de Riesgo:** Crítico
- **Descripción:** Aunque no se muestra en la captura anterior, un análisis más profundo reveló que la aplicación es altamente vulnerable a la Inyección de SQL, específicamente en el formulario de inicio de sesión. Esta vulnerabilidad permite a un atacante manipular las consultas a la base de datos para eludir la autenticación, acceder a datos de cualquier usuario, modificar o incluso eliminar toda la base de datos.

- **Evidencia:** Al introducir la carga útil ' OR 1=1 -- en el campo de correo electrónico del formulario de inicio de sesión, fue posible acceder a la aplicación como el primer usuario registrado en la base de datos sin necesidad de una contraseña válida.
  [Aquí deberías realizar el ataque manualmente en Juice Shop y tomar una captura de pantalla del inicio de sesión exitoso]
- **Medida Correctiva:** Es de **máxima prioridad** eliminar esta vulnerabilidad. Todo el código que interactúa con la base de datos debe ser reescrito para utilizar exclusivamente **consultas parametrizadas (Prepared Statements)**.

C. Cuadro de Porcentaje de Participación

A continuación, se detalla la contribución de cada miembro del equipo al desarrollo y éxito de este proyecto.

Integrante	Tareas Realizadas	Porcentaje de Participación
Yefri	Desarrollo del backend (Python, Flask), configuración de la API de ZAP, despliegue del entorno Docker.	35%
Estuar	Desarrollo del frontend (HTML, JavaScript), diseño de la interfaz de usuario, pruebas de la herramienta.	35%
Richard Cárdenas	Redacción del informe técnico, investigación y documentación de vulnerabilidades y medidas correctivas.	30%
TOTAL		**100%**