Campus Santa Fe

Ingeniería en Tecnologías Computacionales (ITC)

Semestre: 5°

Clase:

Integración de seguridad informática en redes y sistemas de software (Gpo 402)

Título:

Reporte de pruebas de seguridad - Reto

Equipo 6:

Fernando Adrián Fuentes - A01028796

Pedro Mauri Martínez - A01029143

Ricardo Alfredo Calvo - A01028889

Salvador Vaquero Becerra - A01027920

Profesores:

Carlos Enrique Vega Álvarez

Edith Carolina Arias Serna

Lizbeth Peralta Malváez **(Coordinadora)**

Osvaldo Cecilia Martínez

II. Índice

III. Objetivos

Realizar exitosamente pruebas DAST y pruebas SAST en el sistema CRM para el proyecto Sanders con implementaciones de prácticas de seguridad. Para posteriormente analizar las vulnerabilidades presentadas por las pruebas y realizar una comparación entre las pruebas DAST y SAST.

IV. Introducción

La Fundación Sanders es una fundación mexicana la cual se dedica a apoyar a grupos de personas necesitadas por medio de donaciones. El sistema desarrollado fue orientado a atender las necesidades de administración de esta fundación, como la automatización de respuestas de donativos, gestión integral de los donantes, eficiencia para toma de decisiones y transparencia de la Fundación Sanders.

El CRM se ha desarrollado utilizando React-Admin como framework para la interfaz de usuario y MongoDB para la base de datos. La gestión del sistema consta de 2 roles de usuario para diferenciar el nivel de privilegios/permisos entre administrador/empleados, e invitado/donante/usuario de la fundación. Al momento en la que un donador quiere utilizar la aplicación, sigue los pasos siguientes: Inicia sesión dentro una cuenta predeterminada de invitado que está disponible, la cual solo tiene los privilegios de crear donaciones.

- El sistema se trasladó a https.
- La información sensible (contraseñas) están encriptadas utilizando ‘bycript’.
- Se han definido adecuadamente qué roles tienen accesos a qué dashboards y a acciones del CRUD tienen permiso.

V. Pruebas DAST

Objetivos:

- Buscar vulnerabilidades dentro del funcionamiento del sistema.

Enfoque:

- Identificar vulnerabilidades de seguridad a nivel de aplicación mientras está en ejecución. Se priorizó la detección de inyecciones NoSQL y ataques XSS.

Herramienta usada:

- ZAP (OWASP Zed Attack Proxy) es una herramienta gratuita y de código abierto diseñada para realizar pruebas de seguridad en aplicaciones web. Actúa como un proxy entre el navegador y la aplicación web, interceptando y analizando el tráfico para detectar vulnerabilidades como inyecciones SQL, XSS y más.

Configuración de herramienta:

- Las pruebas se realizaron en modalidad Manual de ZAP, se utilizó el navegador Firefox, y se usaron las herramientas de Active Scan y Fuzzer.

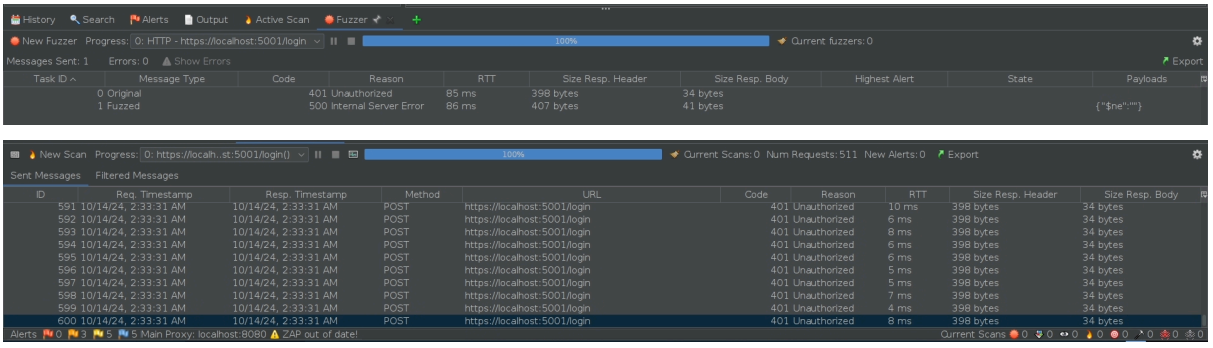
Configuración del entorno:

- La herramienta ZAP fue utilizada a través de una máquina virtual, las máquinas virtuales son entornos de software que emulan un sistema operativo completo, permitiendo ejecutar múltiples sistemas en un solo equipo físico.

Limitaciones:

- Al utilizarse una máquina virtual con recursos limitados seguramente afectaría el rendimiento del escaneo activo y del Fuzzer, lo que podría no descubrir todas las posibles vulnerabilidades.
- Las pruebas fueron realizadas en una simulación (máquina virtual), lo que puede diferir con el entorno de producción.
- Al centrarnos en vulnerabilidades conocidas como XSS e inyección NoSQL, podría dejar fuera otras amenazas potenciales.

Evidencia de ejecución de pruebas:



Hallazgos		
Tipo de vulnerabilidad	Título	Descripción
Alta	XSS	Si se encontró, en los inputs para llenar el formulario de creación.
Alta	Inyección NoSQL	No se encontró, ni utilizando Fuzzer manualmente ni automatizando el proceso de búsqueda de inyección de NoSQL

VI. Pruebas SAST

Objetivos:

- Buscar vulnerabilidades dentro de los scripts de código fuente de la aplicación.

Enfoque:

- Las pruebas SAST se centraron en revisar el código fuente del CRM buscando vulnerabilidades relacionadas con inyecciones SQL, XSS, exposición de datos sensibles y errores comunes de seguridad en bibliotecas externas.

Herramienta usada:

- Snyk es una herramienta creada para desarrolladores, la cual tiene la función de escanear softwares para detectar vulnerabilidades, y corregir problemas de seguridad dentro de sus scripts de código fuente, mediante patrones y prácticas estandarizadas.

Configuración de herramienta:

- Se ingresó a la herramienta por medio de su página web, uno de los integrantes del equipo creó una cuenta e importó el repositorio de GitHub para realizar las pruebas de manera automática.

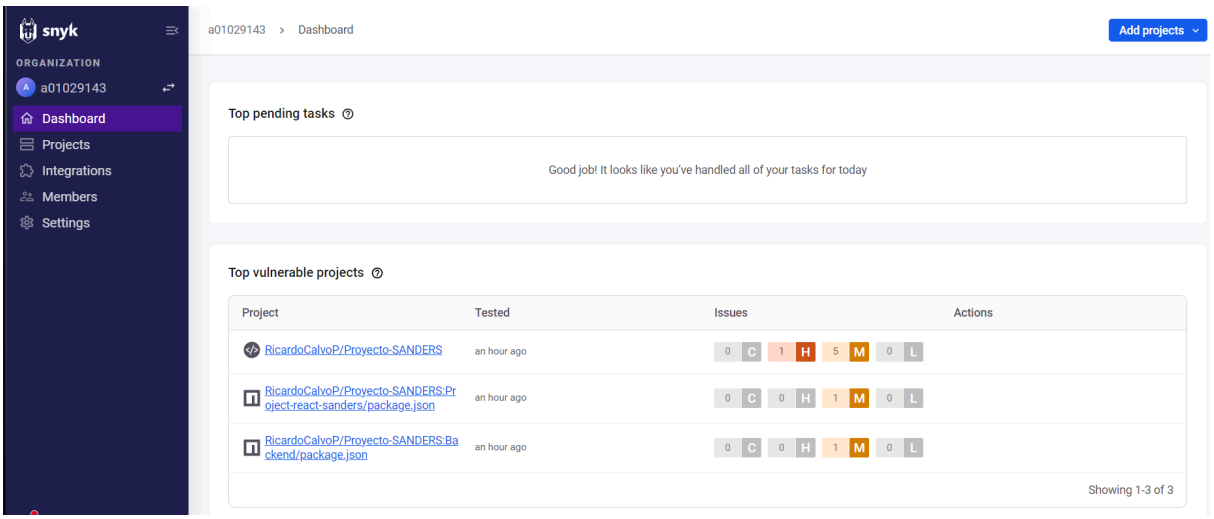
Configuración del entorno:

- Laptop Windows 11 con 16GB de RAM y 930GB de almacenamiento. Se utilizó la herramienta por medio del navegador Opera GX usando conexión de WiFi.

Limitaciones:

- El escaneo de Snyk depende de la base de datos de vulnerabilidades conocida, por lo que las amenazas no identificadas en esa base podrían no ser detectadas.
- El análisis estático no cubre vulnerabilidades que solo pueden manifestarse durante la ejecución, como problemas en la configuración del servidor o errores de lógica del funcionamiento del sistema.
- Al haber importado el repositorio de GitHub del proyecto en la herramienta Snyk, cualquier código fuera de ese repositorio no fue considerado durante las pruebas.

Evidencia de ejecución de pruebas:


snyk
ORGANIZATION
a01029143
Dashboard
Projects
Integrations
Members
Settings
a01029143 › Dashboard
Add projects
Top pending tasks
Good job! It looks like you've handled all of your tasks for today
Top vulnerable projects
Project
Tested
Issues
Actions
RicardoCalvoP/Proyecto-SANDERS
an hour ago
RicardoCalvoP/Proyecto-SANDERS:Project-react-sanders/package.json
an hour ago
RicardoCalvoP/Proyecto-SANDERS:Backend/package.json
an hour ago
Showing 1-3 of 3


H
Cross-site Scripting (XSS)
SNYK CODE | CWE-79
SCORE
767
7 const { to, name, amount } = req.body;
8
9 try {
10 let info = await sendEmail(to, name, amount);
11 res.status(200).send(`Email sent: ${info.response}`);
Unsanitized input from the HTTP request body flows into send, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).
Backend/MailSender/postMail.js
10 steps in 1 file

M Cross-Site Request Forgery (CSRF)

SNYK CODE | CWE-352

SCORE
567

```
8  import { startHttpsServer } from './Server/https.js';
9
10
11 dotenv.config();
12 const app = express();
```

CSRF protection is disabled for your ***Express app***. This allows the attackers to execute requests on a user's behalf.

Backend/**app.js**

1 step in 1 file

M Information Exposure

SNYK CODE | CWE-200

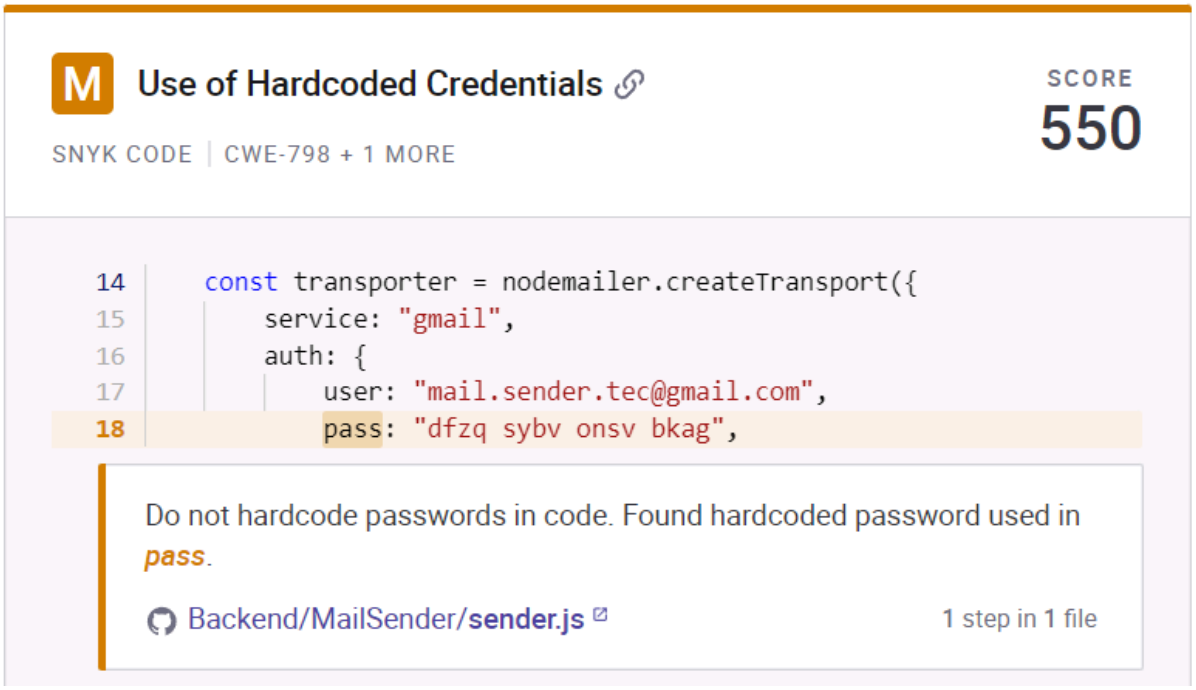
SCORE
567

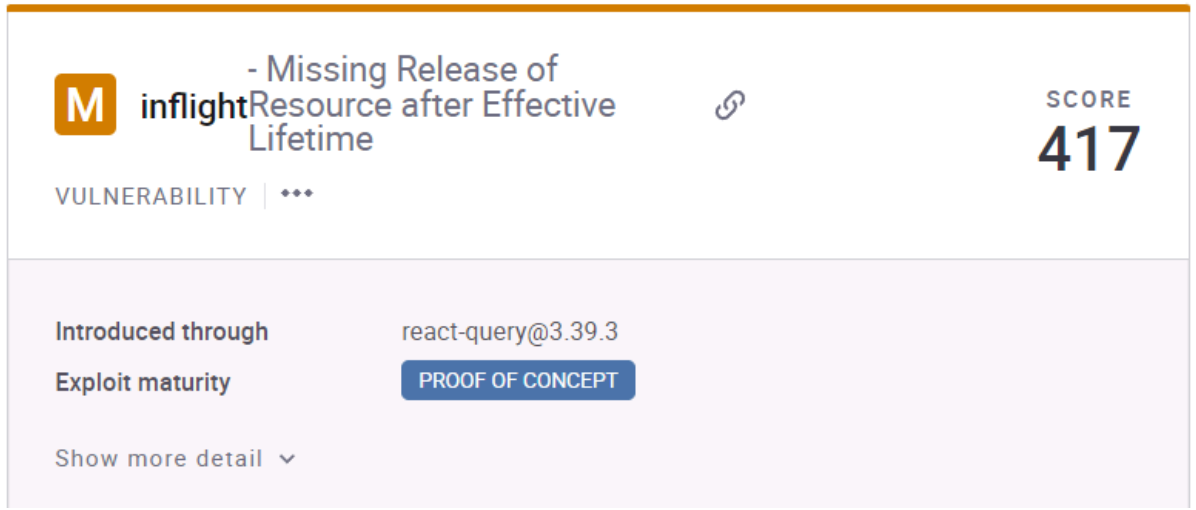
```
8  import { startHttpsServer } from './Server/https.js';
9
10
11 dotenv.config();
12 const app = express();
```

Disable X-Powered-By header for your ***Express app*** (consider using Helmet middleware), because it exposes information about the used framework to potential attackers.

Backend/**app.js**

1 step in 1 file





Hallazgos			
Tipo de vulnerabilidad		Título	Descripción
Alta	(767 / 1000)	XSS	Insertar código dentro de la página.
Media	(567 / 1000)	CSRF	Poder ejecutar solicitudes de la api.
Media	(567 / 1000)	Information Exposure	Da a conocer que se está usando express framework.
Media	(550 / 1000)	Use of Hardcoded	La contraseña del correo

	credentials	que envía la información a los donantes está a plena vista, por lo que cualquiera podría usar la contraseña e ingresar al correo.
Media (417 / 1000)	Missing Release of Resource after Effective Lifetime (en el json del frontend y en el del backend)	Un recurso que al terminar su función no se deshace del recurso, por lo que puede exponer información e incluso crashear node.

VII. Análisis comparativo entre DAST y SAST

Pruebas DAST	Pruebas SAST	Ambos
Analizar ciertos aspectos de la aplicación funcionando. Ya sea un análisis manual o automático.	Revisar el código fuente y las vulnerabilidades que puede llevar.	Son una manera de automatizar

VIII. Conclusiones

Las pruebas DAST y SAST que se realizaron nos permitieron identificar la existencia de ciertas vulnerabilidades importantes dentro de nuestro sistema para la fundación Sanders. En las pruebas DAST se encontraron algunos riesgos críticos, como la oportunidad de ataques XSS, mostrándonos la necesidad de mejorar la validación de entradas en los formularios. En las pruebas SAST se encontraron algunos problemas como el uso de credenciales hardcodeadas, cosa que puede ayudar a revelar información privada o sensible, además de riesgos relacionados con una gestión no completamente eficiente de recursos en el backend y frontend.

Revisando las 2 metodologías de prueba logramos tener una mejor idea sobre los problemas que tiene nuestro proyecto en el dashboard ya en funcionamiento y tambien en el codigo fuente, con esto podemos continuar mejorando la seguridad de nuestro CRM.

IX. Referencias bibliográficas

- Snyk. (s. f.). *Developer security | Snyk*. Recuperado de: https://snyk.io/

- Susnjara, S, & Smalley, I. (2024, 6 septiembre). Máquinas virtuales. *IBM*. Recuperado de: https://www.ibm.com/mx-es/topics/virtual-machines
- *The ZAP homepage*. (s. f.). ZAP. Recuperado de: https://www.zaproxy.org/