ESTRATEGIAS DE MEJORA QUE PERMITAN MITIGAR LAS VULNERABILIDADES DEL APLICATIVO GESTOR DOCUMENTAL ORFEO EN LA ENTIDAD AUTORIDAD

YAIR ALEJANDRO SOSA BARRETO

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BASICAS, TECNOLOGIA E INGENIERIA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D.C.
2023

ESTRATEGIAS DE MEJORA QUE PERMITAN MITIGAR LAS VULNERABILIDADES DEL APLICATIVO GESTOR DOCUMENTAL ORFEO EN LA ENTIDAD AUTORIDAD

YAIR ALEJANDRO SOSA BARRETO

Trabajo de Grado para optar por el título de

ESPECIALISTA EN SEGURIDAD INFORMÁTICA

Katerine Marceles Villalba
Director de Trabajo de Grado

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BASICAS, TECNOLOGIA E INGENIERIA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTA D.C
2023

Nota de aceptación:

_____

_____

_____

_____
Presidente del Jurado

_____
Firma Jurado

_____
Firma Jurado

Dedicatoria

Este proyecto está dedicado a mi familia quienes son la fuente de mi fortaleza, respaldo y un valioso apoyo incondicional, quienes aportan a mi vida felicidad ayudándome a dar lo mejor de mí. A mi madre quien me enseña a perseverar a pesar de las dificultades.

Agradecimientos

A los docentes de la universidad por su dedicación, tiempo, esmero y profesionalismo para dar los lineamientos y culminar este proyecto.

CONTENIDO

Pág.

LISTADO DE ILUSTRACIONES

Pág.

LISTADO DE TABLAS

Pág.

INTRODUCCIÓN

Se puede observar que en la actualidad el uso y operatividad de los sistemas informáticos es algo cada vez más común en todos los contextos de la vida, tanto a nivel personal (hogar y estudios), como a nivel laboral y profesional. Todas las organizaciones sin importar su tamaño dan uso de sistemas documentales para llevar de forma organizada su gestión, esto incluye bases de datos, formatos, registros que sirven como evidencia de las actividades realizadas al interior y algunas se envían de forma externa; dicha información cobra mayor relevancia en las entidades gubernamentales o del estado, ya que esta se convierte en sensible y por lo tanto vulnerable.

Con el fin de suplir estas necesidades informáticas en las organizaciones estatales, se creó en el año 2002 el aplicativo de Gestión Documental (ORFEO) desarrollado por la Superintendencia de Servicio Públicos Domiciliarios (SSPD); bajo licencia GNU/GPL como software libre y manejo colaborativo que tiene, ha sido implementado en la mayoría de las entidades estatales del país.

La Autoridad tiene como uno de sus objetivos tecnológicos para el año 2022, la implementación este aplicativo el cual permite de forma electrónica la elaboración o producción, gestión, el almacenamiento digital de la documentación, minimizando o eliminando su manejo de forma física o en papel. Esto permite estandarizar los procedimientos documentales en la entidad y reducir costos operativos e impactos negativos ambientales; sin embargo, el que dicha información esté de manera digital y con acceso a los funcionarios y partes externas, implica un factor de peligro público con un alto nivel de riesgo por ciberataques. Por tal motivo, se buscará implementar Hardening (endurecimiento o fortalecimiento) con acciones o actividades concretas que conduzcan al fortalecimiento a nivel de seguridad informática en los procesos, actividades o tareas específicas dentro del aplicativo ORFEO; para dicho fin se realizará un análisis bajo el marco de colaboración DevSecOps cuya finalidad es lograr la planificación, desarrollo, automatización y monitoreo bajo normas de seguridad en todas las fases del ciclo de vida de un software ya que se va a desarrollar ajustes a medida conforme a las necesidades de la Autoridad.

1. DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES DEL PROBLEMA

Las entidades públicas en Colombia deben estar comprometidas con la forma de ofrecer una mayor gestión de atención al ciudadano y mejoramiento de procesos corporativos, para ello es básico la operación de las nuevas tecnologías de era digital donde está implícito el ambiente tecnológico y la cultura de las entidades públicas[1]; esto con miras a entregar un mejor servicio con valor en idoneidad para las entidades públicas con un fundamento social y económico.

Bajo esta premisa ORFEO se presenta como una alternativa tecnológica que permite la gestión documental y de procesos, permitiendo gestionar de forma electrónica el trámite, almacenamiento, recuperación y producción de los documentos[2], evitando su manejo en papel físico, esto permite tener una madurez referente a la seguridad informática y la trazabilidad en la gestión del documento mediante su funcionalidad. Igualmente está latente la seguridad del software del gestor documental ya que es un proyecto que se va a ajustar y desarrollar a la medida de la necesidades de la Autoridad en su código fuente, librerías, compilado del código según la versión de los aplicativos utilizados y binarios que son necesarios para su función.

La expansión y masificación de la herramienta ha traspasado fronteras no solo nacionales sino también internacionales, aplicando esta solución tecnológica cómo base en muchas entidades públicas del estado colombiano entre estas están: Secretaria de Movilidad, Ministerio de Salud, IDRD, Secretaria de Gobierno, IDEAM, DNP, Ministerio de Transporte Secretaria Distrital de Seguridad, Convivencia y Justicia, Alcaldía Mayor de Bogotá, EMSA, IDU, UAESP, Función Pública, Alcaldía de Popayán, Empresa Nacional Promotora de Desarrollo Territorial. y en

[1] DEPARTAMENTO ADMINISTRATIVO DE LA FUNCIÓN PÚBLICA. [Sitio Web]. Bogotá: FUNCIONPUBLICA. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.funcionpublica.gov.co/web/eva/biblioteca-virtual/-document_library/bGsp2IjUBdeu/view_file/34268003>

[2] ORFEO LIBRE ORG. [Sitio Web]. Bogotá: ORFEOLIBRE.ORG. [Consulta: 12 de marzo 2022]. Disponible en <https://orfeolibre.org/inicio/detalles-de-orfeo-sistema-de-gestion-documental>

varios países como Ecuador, Chile, México, Venezuela, Argentina, Perú, por tanto, se ve la necesidad de un fortalecimiento a nivel de seguridad de esta tecnología emergente ya que no se cuentan con lineamientos específicos para a la seguridad del sistema de gestión documental en su código de desarrollo, dejando vacíos en la gestión de seguridad de la información debido a su criticidad y sensibilidad.

1.2 FORMULACIÓN PREGUNTA PROBLEMA

Pensar en un marco de seguridad integral como ítem fundamental de cualquier solución tecnológica, es primordial para garantizar la disponibilidad, confidencialidad e integridad de la información, buscando mitigar las vulnerabilidades e impactos que puedan presentarse a futuro, logrando la protección óptima para el funcionamiento de estas herramientas, por tanto, cualquier acción que permita aportar el fortalecimiento de dicha herramienta, debe considerarse para su aplicación y validación ya que es un proyecto que se va a ajustar y aplicar desarrollos a la medida de la necesidades de la Autoridad en su código fuente, librerías, compilado del código según la versión de los aplicativos utilizados y binarios que son necesarios para su función.

Bajo esta premisa, surge la pregunta que dará el lineamiento al desarrollo del proyecto aplicado

¿Cómo ejecutar acciones para remediar las vulnerabilidades del software a desarrollar del proyecto ORFEO aplicando el marco DevSecOps para mejorar la seguridad en las entidades públicas que gestionan este software?

Para la Autoridad Nacional de Licencia Ambientales será útil las respectivas validaciones que se puedan identificar, ya que entrega valor al servicio, garantizando calidad y seguridad a la herramienta de gestión documental.

2. JUSTIFICACIÓN

La transformación digital permite desarrollar acciones para garantizar una correcta transición en la misión del negocio y afirmar el aprovechamiento de las herramientas tecnológicas que surgen, un ejemplo de esta tecnología es el uso de la digitalización de documentos ya que esta permite facilidad en el trámite, distribución y consulta de documentos públicos[3], aplicando la migración y trasferencia apoyados del uso de recursos informáticos con documentos de gran variedad, físico, electrónico, digital, etc. que permitirá la permanencia en un tiempo considerable de la información almacenada en medios digitales. Todos estos atributos están identificados en la gestión documental ya que esta contempla una sinergia de procesos, gestiones administrativas y metodologías pertinentes a la organización, administración y distribución de los documentos producidos y recibidos por las empresas y organizaciones[4], teniendo el control y trazabilidad cuya finalidad es facilitar su manejo y preservación, volviéndose una base primordial de los procesos de toda entidad u organización.

Para la Autoridad con el objetivo de cumplir la normatividad exigida por las leyes, decretos que rigen a las entidades públicas y en aras de ser vinculante a la transformación tecnológica, decidió implementar ORFEO como su Gestor Documental. Debido a la importancia que implica el alcance de este aplicativo cómo base primordial de los procesos de la entidad, se debe certificar la disponibilidad, confidencialidad e integridad de la información, por esta razón este proyecto se justifica para validar las oportunidades de mejora y fortalecimiento de la seguridad de este.

---

[3] MINISTERIO DE TECNOLOGIAS DE LA INFORMACION Y COMUNICACIONES. [Sitio Web]. Bogotá: MINTIC. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/149186:MinTIC-publica-el-Marco-de-Transformacion-Digital-para-mejorar-la-relacion-Estado-ciudadano>

[4] ARCIHIVO GENERAL DE LA NACION. [Sitio Web]. Bogotá: AGN. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.archivogeneral.gov.co/sites/default/files/Estructura_Web/5_Consulte/Recursos/Publicacionees/ImplementacionSGDEA.pdf>

3.OBJETIVOS

3.1 OBJETIVO GENERAL

Proponer acciones que permitan mitigar las vulnerabilidades del software de gestión documental ORFEO aplicando el marco DevSecOps para mejorar la seguridad en la Autoridad.

3.2 OBJETIVOS ESPECÍFICOS

Analizar las vulnerabilidades del sistema de información de gestión documental ORFEO en la infraestructura tecnológica de la Autoridad por medio de la ejecución de pruebas de seguridad (SAST) y (DAST) en el marco de las etapas del DevSecOps.

Proponer recomendaciones para mitigar el riesgo de las vulnerabilidades encontradas dentro del sistema de gestión documental ORFEO para la Autoridad aplicando la evaluación de los ítems identificados.

Elaborar la documentación sobre el análisis y pruebas realizadas al sistema de información ORFEO, para que sean aplicadas conforme a las políticas establecidas por la Autoridad, por medio de la formalización del cumplimiento del proyecto.

4. MARCO REFERENCIAL

4.1 MARCO TEORICO

Para el análisis de la importancia que refieren a los sistemas de información enfocados específicamente a los que operan la gestión documental, se cuentan con varios tipos de postulaciones, uno de ellos es el referenciado por Erazo, Garces y Muñoz donde enfatizan la importancia de ellos relatando que:

> En el servidor de gestión documental, se maneja información clasificada en financiera, comercial, reportes al SUI (Sistema Único de Información de Servicios Públicos), correspondencia interna y externa, la cual es de vital importancia, ya que si es expuesta a terceros se ve afectada en los tres pilares de la seguridad de la información: Confidencialidad, Integridad y Disponibilidad[5].

Dejando ver que el cuidar la seguridad del sistema de información es de vital importancia para garantizar el funcionamiento seguro de este, debido a que reposa archivos principales e históricos de cualquier organización.

Para Brunet es importante cobijar la gestión y la capacidad del gestor documental de forma segura para validar que

> El desenvolvimiento de políticas públicas destinadas a la protección de los datos de los diferentes en los diferentes niveles de la gestión documental, sumados a calidad de la información que se maneje, en dependencia además con la naturaleza y el contexto de la recolección y tratamiento de estos, hacen a una indiscutible necesidad de interrelación existen.[6]

Por tanto, el objetivo de los sistemas de gestión de seguridad de la información debe estar enfocado en el otorgar protección a la integridad,

---

[5] Guerrero Erazo, H., Lasso Garces, L., & Legarda Muñoz, A. [en línea]. Pasto (Colombia): Universidad Nacional Abierta y a Distancia, 2015, [Consultado 25 de noviembre 2022]. Disponible en:https://repository.unad.edu.co/bitstream/handle/10596/3451/5203676.pdf?sequence=1&isAllowed=y/>

[6] Nahabetián Brunet, L. Protección de datos y gestión documental: Decálogo ampliado para la sociedad de la información. Revista de la Facultad de Derecho, (39), 9-9. [en línea]. Uruguay, 2015, [Consultado 25 de noviembre 2022]. Disponible en:< http://www.scielo.edu.uy/pdf/rfd/n39/n39a09.pdf>

autenticidad, fiabilidad y disponibilidad de los activos de información sea que éstos se presenten en formatos digitales o físicos.[7] Cualquier sistema de información debe contar la seguridad, monitoreo y mecanismos de control pertinentes para garantizar la protección de los diferentes activos de la entidad, bien sea de orden privada o pública.

El adoptar lineamientos de seguridad adecuados en un gestor documental garantizara beneficios que según Bautista afirma son:

> Eficacia en la gestión de los procesos administrativos como: Calidad en los procesos, seguimiento y control de las acciones realizadas, información veraz en línea, reducción de tiempo y aporta información relevante para la toma de decisiones.
>
> Agiliza los procesos para el acceso de la información, puesto que garantiza veracidad en la búsqueda de información, y minimiza los tiempos de respuesta, optimizando la productividad organizacional.
>
> Reducción de acervo documental, en los espacios físicos tales como: Archivadores, estantería, Cajas Archivísticas.
>
> Reduce los riesgos de perdida y daños documentales.
> Minimiza costos de procesos archivísticos. (almacenamiento, recuperación).[8]

Se puede percibir que pensar en la seguridad de un gestor documental es fundamental para los procesos de la organización, permitiendo tener una sinergia en los procesos y estructurar la información de forma ordenada, disponible y segura.

En el cumplimiento de las normativas de seguridad, para los sistemas de información, se debe involucrar a toda la organización, porque la interacción de este es transversal a las funciones y operación de cualquier entidad, Chávez lo referencia de esta forma:

---

[7] Ibid., p. 20.

[8] Valencia Bautista, A. D. Análisis de los riesgos de seguridad de la información del sistema de gestión documental de la Alcaldía Municipal de Ibagué. [en línea]. Ibague (Colombia): Universidad Nacional Abierta y a Distancia, 2022, [Consultado 25 de noviembre 2022]. Disponible<https://repository.unad.edu.co/bitstream/handle/10596/51504/avalenciab.pdf?sequence=1&isAllowed=y >

Se debe tener en cuenta la estructura organizacional, con los procesos que se lideran en cada área y los respectivos roles y responsabilidades a la seguridad de la información para cada una de las áreas y oficinas, aplicando la estructura de manejo y producción de información establecer políticas documentales apoyándose en las nuevas disposiciones y normas establecidas para el manejo y disposición final de la información, teniendo en cuenta los siguientes aspectos relevantes:

- Responsables de la seguridad de la información y seguridad informática
- Responsable del cumplimiento de la normatividad vigente
- Responsables y encargado del tratamiento de los datos personales.[9]

Se concluye que un sistema de información, como lo es el gestor documental es base importante para la gestión de información, no solo en el aspecto documental sino en el orden de estructuración de procesos funcionales y operativos de las organizaciones, estos fundamentos dan base importante para aplicar el resguardo en la seguridad total del sistema, desde su planeación, desarrollo de código, despliegue a producción e infraestructura tecnológica, para ubicarlo frente al usuario.

4.1.1 Amenazas a las que se exponen sistemas de información con un desarrollo no seguro. En el ciclo de desarrollo de un software, se deben tener en cuenta varias consideraciones de seguridad, ya que una ejecución de código no seguro afecta de forma directa los sistemas de información que con el interactúan, por tanto, las amenazas más recurrentes basados en el proyecto Open Web Aplication Security Proyect (OWASP) que es un proyecto de código abierto cuya finalidad es determinar y mitigar las vulnerabilidades de un software web son:

4.1.1.1 Backdoor: Punto vulnerable donde un usuario no autorizado puede acceder al sistema, por medio de un programa que se instala en el equipo de la víctima, pueden validarse como un programa maligno, cuentan con codificación propia[10].

[9] Chaves Rosero, M. A. Resguardar la información del sistema de gestión documental Docunet en la Empresa Contactar-Pasto.

[10] INCIBE, [Sitio Web]. España: INCIBE. [Consulta: 24 de octubre 2022]. Disponible en:<https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_glosario_ciberseguridad_2021.pdf>

4.1.1.2 Cross Site Request Forgery: Es la falsificación de petición en sitios cruzados, específicamente contra páginas web, donde un software malicioso obliga al site a ejecutar acciones no autorizadas a nombre de algún usuario que accede a la página, descarga la confianza que un site aloja en un usuario específico[11].

4.1.1.3 Denegación de Servicio: Es dejar un sistema, aplicación o dispositivo fuera servicio por medio de una saturación de peticiones, consumiendo recursos importantes de este[12].

4.1.1.4 Desbordamiento de Búfer: Se busca adquirir acceso remoto a un sistema por medio de un ataque que aprovecha defectos en el código fuente provocando error o fallo de sistema superando los umbrales de capacidad de este sobre escribiendo datos en los puntos de memoria[13].

4.1.1.5 Envenenamiento de DNS: Obtención del control de un servidor para gestionar peticiones que le son dirigidas para direccionarlas a otros sitios web que no son legítimos y que son puestos por el atacante, estos sites pueden contener suplantación de identidad o códigos maliciosos[14].

4.1.1.6 Inyección de Código: Proceso donde se inserta en un software especifico una cadena de instrucciones que no son parte del código original, del programa o la aplicación, generando comportamientos extraños o particulares que no son diseño del original[15].

4.1.1.7 Session Hijacking: Es el secuestro de cookies, que busca interceptar la sesión de un usuario en internet con el objetivo de acceder a su información, usando las sesiones no cifradas por protocolos como HTTP[16].

4.1.1.8 Spoofing: Es una técnica de suplantación de identidad en la red, con el uso de un programa maligno se suplanta la IP, protocolo de resolución de dirección, nombre de dominio, página web o correo electrónico[17].

---

[11] Ibid., p. 34.
[12] Ibid., p. 36.
[13] Ibid., p. 36.
[14] Ibid., p. 40.
[15] Ibid., p. 53.
[16] Ibid., p. 70.
[17] Ibid., p. 73.

4.1.1.9 Watering Hole: Ataque donde se infecta una página web de legitima con alta regularidad de tránsito de visitantes los cuales quedan infectados al momento del acceso[18].

4.1.1.10 XSS: Cuando un site web tiene contenidos dinámicos y dependen de la interacción con el usuario, es posible insertar un formulario o un programa malicioso, buscado su ejecución para acciones como cambio de configuraciones, secuestro de cuentas, o monitoreo del tráfico de la red[19].

4.1.2 Diferencias, Semejanzas Entre Los Enfoques De Devops, Devsecops y Técnicas de Testeo para Sistemas de Información.

Linda Rosencrance, en su artículo para ComputerWeekly, plantea algunas diferencias y semejanzas entre los enfoques de DevOps y DevSecOps así:

> Son similares en algunos aspectos, incluido el uso de la automatización y los procesos continuos para dictaminar ciclos colaborativos de florecimiento. Sin embargo, DevOps prioriza la velocidad de entrega, mientras que DevSecOps desplaza la seguridad hacia la izquierda, lo que significa mover la seguridad al punto más temprano posible en el proceso de desarrollo[20].

Dando la base del porque tiene más beneficios realizar la implementación de ORFEO con el enfoque DevSecOps.

De acuerdo con lo que IONOS Digital Guide y su autor refieren que:

> Desde hace algunos años, la seguridad viene ganando cada vez más importancia en el ámbito del desarrollo de software. En especial, cuando se trata de procesos cortos de desarrollo, que tienen que producirse cada vez con más rapidez entre las versiones, el cumplimiento de los estándares de seguridad es todo un desafío. En este contexto, si se deja

[18] Ibid., p. 78.
[19] Ibid., p. 79.
[20] ROSENCRANCE, Linda. SecOps o DevSecOps. *ComputerWeekly* [en línea]. 2021, octubre, [Consultado 12 de marzo 2022]. Disponible en: <https://www.computerweekly.com/es/definicion/SecOps-o-DevSecOps?_gl=1*1olh05s*_ga*MjEwNzAwNzgzMC4xNjUzNzQ5NTIx*_ga_TQKE4GS5P9*MTY1Mzc0OTUyMS4xLjEuMTY1Mzc0OTYxNS4w&_ga=2.38138851.1378944387.1653749521-107007830.1653749521>

> la seguridad para el final, tras la fase de desarrollo en sí, puede que no se logre alcanzar tales estándares[21].

Por lo que para la ejecución en la organización y debido a que la implementación de ORFEO está en su etapa inicial, se decide adoptar el DevSecOps que integra y ofrece un alto nivel de seguridad y ciclos cortos de lanzamiento de productos. De ahí:

> Que DevSecOps se implemente a través de una guía adecuada y efectiva depende de cómo se adapten los equipos y comisión del ente a la alteración que supone. Sin una organización empresarial abierta e interconectada que facilite la aviso entre equipos y departamentos, el principio de DevSecOps no puede funcionar[22].

Dentro de las metodologías de evaluación o testeos de los sistemas de información según J Gutierrez existen varios tipos de pruebas entre ellas:

> Pruebas Unitarias: Las que prueban el diseño, comportamiento y funcionalidad de todos los componentes ya codificado.
>
> Pruebas de Integración: Validación del cumplimiento de funcionalidad de componentes e interfaces y sus interacciones entre sí.
>
> Pruebas de Sistema: Comprobación de validación a fondo de funcionalidad, integración y ejecución con un ambiente similar al productivo.
>
> Pruebas de Implantación: Validación del sistema total en el entorno productivo.
>
> Pruebas de Aceptación: Verificación del sistema en el cumplimiento de todos los requisitos y va con el Visto Bueno del usuario definitivo.
>
> Pruebas de Regresión: Comprobación de cambios realizados a lo componentes, donde no se generan fallos adicionales en otros componentes no modificados[23].

---

[21] DIGITAL GUIDE IONOS. [Sitio Web]. España: IONOS. [Consultado 13 de marzo 2022]. Disponible en: <https://www.ionos.es/digitalguide/servidores/seguridad/que-es-devsecops>

[22] RED HAT. [Sitio Web]. Estados Unidos: RED HAT. [Consulta: 13 de marzo 2022]. Disponible en: <https://www.redhat.com/es/topics/devops/what-is-devsecops>

[23] Gutierrez, J. J., Escalona, M. J., Mejías, M., & Torres, J. (2004). Aplicando técnicas de testing en sistemas para la difusión Patrimonial. In Actas del V Congreso Nacional de Turismo y Tecnologías de la Información y las comunicaciones (TURITEC'2004) (pp. 237-252). [Consulta: 13

Por esta razón se evidencia la importancia de evaluar el código fuente, los aplicativos y sistemas de información que son la base de la gestión y operación de la Autoridad, bajo un marco de seguridad como DevSecOps, se hace de carácter transcendental que el despacho de la dirección con la gestión y soporte de la coordinación de TI, igualmente se debe informar los beneficios en la nueva integración del marco de seguridad involucrando a todos los grupos y colaboradores en la toma de decisiones que conlleven cambios.

4.2 MARCO CONCEPTUAL

Para dar inicio con la temática del proyecto, se deben contemplar conceptos base, tanto en la gestión de vulnerabilidades como los procesos, componentes y dispositivos que permiten la interoperabilidad del sistema de información a analizar, para este caso (ORFEO), esto permite contextualizar de forma directa con la operación y función del proyecto, entre ellos están:

4.2.1 Seguridad Informática: Como lo revela el estudio realizado por Alba Estrada comenta que:

> Con base en la norma ISO/IEC 27001 se define la seguridad informática como aquellas características y condiciones de sistemas de procesamientos de datos y su almacenamiento que permite garantizar la confidencialidad, integridad y disponibilidad por lo cual, aporta los pasos para la implementación de un sistema de arreglo inofensivo, pero solamente hace referencia a la parte de seguridad sin tener en cuenta el desarrollo y de manera menos importante la operatividad del usuario, en este aspecto[24].

de marzo 2022]. Disponible en: https://www.researchgate.net/profile/Mj-Escalona/publication/228545848_Aplicando_tecnicas_de_testing_en_sistemas_para_la_difusion_Patrimonial/links/00b495303d59041d75000000/Aplicando-tecnicas-de-testing-en-sistemas-para-la-difusion-Patrimonial.pdf

[24] ESTRADA, ALBA, MARTIN. Fundamentos para implementar y certificar un sistema de gestión de seguridad informática bajo la norma ISO/IEC 27001. Serie científica de la Universidad de las Ciencias Informáticas. 2012

4.2.2 Apache: Es un software con servicios y plataforma web gratuito Open Source que se puede acoplar con diversidad de software para servidores HTTP para sistemas operativos modernos, incluyendo UNIX y Windows, con el objetivo de proporcionar un servidor seguro, eficiente y extensible[25].

4.2.3 AST: Application Security Testing, consiste en la ejecución de las diferentes técnicas de prueba de seguridad donde su objetivo es buscar vulnerabilidades de seguridad en las aplicaciones[26].

4.2.4 Black Box: Es una asignación de pruebas donde el tester de penetración se ubica en el rol del atacante sin conocimiento interno del sistema objetivo[27].

4.2.5 DAST: Dynamic Application Security testing, son pruebas de seguridad de aplicación de forma dinámica o conocidas como caja negra en la cual se pueden encontrar vulnerabilidades de seguridad en la ejecución actica del código normalmente para aplicaciones web[28].

4.2.6 DevOps: Es una nueva forma de trabajo colaborativa, una metodología que asocia el desarrollo del software y la operación[29].

4.2.7 DevSecOps: Es un marco de integración o colaboración del proceso desarrollo, la gestión seguridad informática y la operación con la integración del desarrollo de código fuente y despliegues[30].

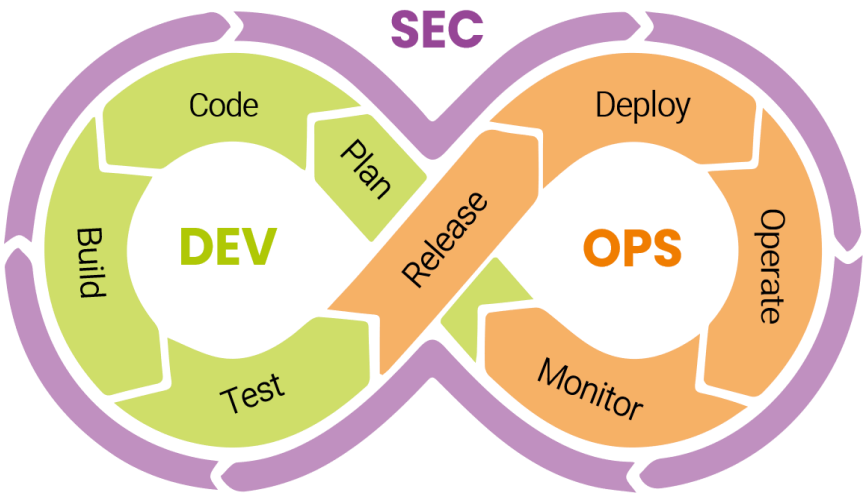
En este sentido el marco DevSecOps permitirá llevar a cabo la ejecución y logro de los objetivos planteados.

El marco DevSecOps se identifica con el símbolo del infinito en el cual se evidencian todas las interacciones y aspectos para tener en cuenta dentro

[25] APACHE HTTP SERVER PROYECT. [Sitio web]. New Orleands: APACHECON NORTH AMERICA. [Consulta: 25 de octubre 2022]. Disponible en <https://httpd.apache.org/>
[26] DEVSECOPS LATINO AMERICA. Una guía práctica sobre la nueva era del testing de seguridad. Argentina: DevSecOps Argentina, 2020. E-book. [En línea]. (Recuperado en 25 de octubre de 2022). Disponible en <https://docs.google.com/viewer?url=https://devsecops-latam.org/contenidos/ebook-devsecops-calms.pdf&embedded=true>
[27] Ibid., p. 7.
[28] Ibid., p. 39.
[29] Ibid., p. 4.
[30] Ibid., p. 4.

de esta práctica de desarrollo seguro, iniciando desde la planeación, codificación, construcción, pruebas, empaquetado, despliegues, operación y el monitoreo del software, como se puede ver en la siguiente ilustración 1:

Ilustración 1. Marco DevSecOps



Fuente 1. KIREYGROUP.COM, DevSecOps - El efecto Darwin. Milano

Esta metodología busca interiorizar la cultura de seguridad dentro de los estándares y procesos de desarrollo, en la gestión diaria con la inclusión de prácticas asociadas a seguridad, como el análisis de código seguro, estando estático o dinámico, en conjunto con las dependencias o al integrar liberarías.

> Este enfoque de seguridad como se puede ver en la ilustración 1, ésta permite un modelamiento de amenazas desde la funcionabilidad del aplicativo identificando posibles amenazas hasta el impacto que pueda generar[31].
>
> Se debe tener como referencia que las áreas involucradas (Desarrollo, seguridad y operación) deben estar en la misma sinergia para tener en conjunto un centro y una visión en común y es la búsqueda del aseguramiento en todas las fases o ciclo de vida del proyecto.

[31] Ibid., p. 12.

> Se debe enfocar en el análisis de la codificación segura validando los datos de entrada, observar la estructura del código en búsqueda de vulnerabilidades apoyándose en herramientas como Static Application Security Testing (SAST) para el código en reposo y las Dynamic Application Security Testing (DAST) cuando el código se opere, igualmente las pruebas de explotación de vulnerabilidades PenTesting, sin olvidar la gestión de monitoreo en búsqueda de eventos de seguridad en la infraestructura, para que en sus fases finales se apliquen constantes auditorias, parches, escaneo en el proceso operativo del aplicativo y finalmente entregar un valor en la configuración y traslado de datos seguros del proyecto[32].

Se debe entender que este marco involucra el trabajo en equipo, donde el desarrollo se beneficia a tener un software seguro y de calidad, la seguridad reduce costos y esfuerzos y aumenta el cumplimiento y por último las operaciones pueden evitar vulnerabilidades de las aplicaciones en producción aplicando los controles necesarios. La finalidad es lograr integrar el modelo de seguridad desde las etapas iniciales del proyecto y no en la etapa final como los modelos tradicionales de análisis para este caso ORFEO.

4.2.8 Gestión documental: Esto se puede definir como un sistema abierto que tiene interacciones con otras disciplinas y tiene como principios la asignación de documentos a cada usuario, el uso de los documentos y salvaguardarlos en el tiempo [33], para Sarada lo referencia

> Como toda información que es generada por un sistema documental se representa mediante documentos que son manejados por el mismo sistema como objetos de información digital que cuentan con características aparte del documento como los metadatos y a información de identificación digital de cada documento[34].

4.2.9 Orfeo:

> Sistema de gestión documental desarrollado primeramente por la Superintendencia de Servicios Públicos Domiciliarios (SSPD) en Colombia, como software libre bajo licencia GNU/GPL cuya finalidad es el concepto

[32] Ibid., p. 37.
[33] RANGANATHAN, S. R.: The Five Laws of Library Science. Bangalore: Sarada Ranganathan Endowment for Library Science, 1993, en Vickery.B. C., y Vickery, A. Bowker-Saur.1992, p. 260.
[34] RANGANATHAN, S. R.: The Five Laws of Librarv Science Op.cit, p.18

de la creación colectiva y libre. Una herramienta puede instalarse en cualquier sistema Operativo (GNU/Linux, Unix, Windows), con diferentes bases de datos (Postgrest SQL, Oracle, MySqL y Ms-SQL), además maneja múltiples tipos de Formatos (ODT, XML, DOC), logrando así obtener independencia de plataforma tecnológica y reducción de costos en la implantación[35].

4.2.10 PHP: Es un Procesado de Hiper Texto, es un lenguaje de código abierto cuyo diseño está enfocado al desarrollo de interacción web y se puede acoplar en HTML, opera sintaxis a C, Perl y Java[36].

4.2.11 SAST: Pruebas de seguridad de aplicabilidad estática o también llamada prueba de caja blanca donde permite al personal de desarrollo visualizar vulnerabilidades en el código de desarrollo estando éste en un estado de reposo[37].

4.2.12 Sistemas de información: Para toda organización es indispensable apoyar su gestión en la tecnología, sistemas de información, tecnologías emergentes que le permiten mejorar sus procesos, desde la óptica de los sistemas de información son procesos identificables basados en una estructura que realiza alguna acción, que se desarrolla, evoluciona y que cambia con el ambiente para obtener un resultado, por tanto se puede identificar que la gestión de información es la integración ya nuevo marco de trabajo ofrecidas por los sistemas de información.[38]

Estos sistemas permiten una interacción que mantienen los flujos de información con entes externos y otros al interior, igualmente pueden ser transversales, ascendentes o descendentes de tipo formal o informal que en todas estas transiciones es vinculante el usuario que gestiona el sistema y que usa el resultado de este. Por consiguiente, la suma de los

[35] ORFEO LIBRE ORG. [Sitio Web]. Bogotá: ORFEOLIBRE.ORG. [Consulta: 12 de marzo 2022]. Disponible en <https://orfeolibre.org/inicio/detalles-de-orfeo-sistema-de-gestion-documental>
[36]PHP. [Sitio web]. [Consulta: 25 de octubre 2022]. Disponible en <https://www.php.net/manual/es/intro-whatis.php/>
[37] DEVSECOPS LATINO AMERICA. Una guía práctica sobre la nueva era del testing de seguridad. Argentina: DevSecOps Argentina, 2020. E-book. [En línea]. (Recuperado en 25 de octubre de 2022). Disponible en <https://docs.google.com/viewer?url=https://devsecops-latam.org/contenidos/ebook-devsecops-calms.pdf&embedded=true>
[38] GÓMEZ, Laureano Felipe. Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir. Códices [en línea]. Bogotá (Colombia): Universidad de la Salle, 30 de junio de 2007, vol. 3, nro. 01 [Consultado 15 de marzo 2022]. ISSN 1794-9815. Disponible en: <https://ciencia.lasalle.edu.co/co/vol3/iss1/2/>

> conjuntos de diferentes recursos bien sean estos económicos, técnicos o humanos y la forma como interactúan y relacionan entorno a suplir una necesidad de información, para la toma de decisiones.[39]

4.2.13 Software Libre: El software libre tiene una connotación de gestión transparente, susceptible a modificaciones, con la finalidad de originar código haciéndolo libre y sí se incorpora con otro software, esta derivación debe ser considerada también libre. Richard Stallman lo define como

> Aquel que tiene la libertad de ejecutar el programa con el propósito específico, debe tener la libertad para modificar el programa y ajustarlo a las necesidades particulares, igualmente la libertad de redistribución de copias de forma gratuita o una remuneración y la libertad de distribuir sus versiones modificadas del programa aprovechando las mejoras introducidas o desarrolladas por la persona.[40]

4.2.14 Vulnerabilidades: Debilidad o fallo en un sistema, programa o código que esta inseguro pudiendo otorgar que un agresor pueda deteriorar él mismo.[41]

4.2.15 White box: Es la asignación de pruebas donde los testers tienen acceso completo al código fuente a la documentación y arquitectura de la aplicación o el objetivo a analizar.[42]

4.2.16 Seguridad en la información: Es un conjunto de procesos que buscan gestionar la accesibilidad, aseguramiento, confidencialidad, integridad y disponibilidad de la información, con el objetivo de minimizar los riesgos de la seguridad de la información.[43]

---

[39] GÓMEZ, Laureano Felipe. Interoperabilidad en los Sistemas de Información Documental (SID): Op.cit, p.25

[40] STALLMAN, Richard. Free software, free society: Selected essays of Richard M. Stallman. Lulu. com, 2002.

[41] DEVSECOPS LATINO AMERICA. Una guía práctica sobre la nueva era del testing de seguridad. Argentina: DevSecOps Argentina, 2020. E-book. [En línea]. (Recuperado en 25 de octubre de 2022). Disponible en <https://docs.google.com/viewer?url=https://devsecops-latam.org/contenidos/ebook-devsecops-calms.pdf&embedded=true>

[42] Ibid., p. 7.

[43] Ibid., p. 70.

4.3 MARCO LEGAL

Existen varias normatividades vigentes que aplican para el manejo y operación del software de gestión documental que involucran la temática de seguridad de la información, el tratamiento de datos y la administración de la información. Es importante aclarar que se enlistan de acuerdo con la jerarquización de la pirámide documental y por el año de expedición.

Ley 527 de 1999 Expedida el 18 de agosto por el congreso de la república y por medio de la cual se define y reglamenta el acceso y uso de los mensajes de datos, del comercio electrónico y de las firmas digitales, y se establecen las entidades de certificación.[44] Nos da el lineamiento referente a la gestión de las firmas digitales en la documentación electrónica, ya que los documentos deben lleva firmas para garantizar la veracidad de este.

En el proceso de gestión, normalización, tratamiento de archivos y los principios en la gestión documental, la Ley 594 De 2000 Expedida el 14 de julio por el congreso de la república y por medio de la cual se dicta la Ley General de Archivos.[45] Es la base para la estructuración del gestor documental que da el lineamiento para el manejo de los archivos físicos y digitales.

La ley 1273 de 2009 Expedida el 5 de enero por el congreso de la república y por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado denominado 'de la protección de la información y de los datos' y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones.[46] Es uno de los pilares fundamentales para buscar la seguridad, confidencialidad de los datos y la información operados por el gestor documental.

---

[44] COLOMBIA. CONGRESO DE LA REPUBLICA. Ley 527. (18, agosto, 1999). Por lo cual se define y reglamenta el uso de datos y comercio electrónico. Bogotá: Congreso de la República, 1999.
[45] COLOMBIA. CONGRESO DE LA REPUBLICA. Ley 594. (14, julio, 2000). Por lo cual se define la ley general de archivo. Bogotá: Congreso de la República, 2000.
[46] COLOMBIA. CONGRESO DE LA REPUBLICA. Ley 1273. (5, enero, 2009). Por lo cual se crea un bien jurídico tutelado denominado protección de la información y datos. Bogotá: Congreso de la República, 2009.

Para garantizar la disponibilidad de la información contenida en el gestor la ley 1712 De 2014 Expedida el día 6 de marzo por el congreso de la república y por medio de la cual se crea la Ley de Transparencia y del Derecho de Acceso a la Información Pública Nacional y se dictan otras disposiciones.[47] Para la consulta de los datos por parte de cualquier solicitante, pero en el marco de confidencialidad de los datos que opera el gestor documental.

El decreto 19 de 2012 Expedido el 10 de enero por el presidente de la república y por el cual se dictan normas para suprimir o reformar regulaciones, procedimientos y trámites innecesarios existentes en la Administración Pública.[48] Busca que el sistema de información facilite los procesos y tramites del solicitante frente a la gestión de consulta y disponibilidad de los documentos del sistema de información.

El acuerdo 07 De 1994 Expedido el 29 de junio por el Archivo General de la Nación en donde se adopta el Reglamento General de Archivos, como norma reguladora del quehacer archivístico.[49] Rige la labor del tratamiento archivístico que se debe ejecutar en el sistema de información o gestor documental.

Acuerdo 060 De 2001 Expedido el 30 de octubre por el Archivo General de la Nación y por medio del cual se establecen los lineamientos y procedimientos que permiten a las unidades de correspondencia, cumplir con los programas de gestión documental, para la recepción, distribución, seguimiento, conservación y consulta de los documentos[50]. Da el lineamiento para el ciclo de vida de los documentos que tratara el sistema de información.

[47] COLOMBIA. CONGRESO DE LA REPUBLICA. Ley 1712. (6, marzo, 2014). Por lo cual se define la ley de trasparencia y acceso a información pública. Bogotá: Congreso de la República, 2014.
[48] COLOMBIA. PRESIDENCIA DE LA REPUBLICA. Decreto 19. (10, enero, 2012). Por lo cual se define y reglamenta ley de anti tramites. Bogotá: Presidencia de la República, 2012.
[49] COLOMBIA. ARCHIVO GENERAL DE LA NACION. Acuerdo 07. (29, junio, 1994). Por lo cual se define el reglamento de archivos y quehacer archivístico. Bogotá: Archivo General de la Nación, 1994.
[50] COLOMBIA. ARCHIVO GENERAL DE LA NACION. Acuerdo 060. (30, octubre, 2001). Por lo cual se define los programas de gestión documental. Bogotá: Archivo General de la Nación, 2001.

4.4 MARCO DE ANTECEDENTES

En función del estudio de las diferentes posturas frente al análisis de vulnerabilidades en el software de gestión documental, se puede observar que se aplican marcos o metodologías enfocadas específicamente en las aplicaciones web, esto porque el servicio prestado es consumido por ese canal, según Diana Caucali

> Es importante identificar y describir los requisitos necesarios con los que debe contar un software libre que garantice la seguridad informática. Por ello el modelo basado en OWASP permite diversificar y analizar los diferentes aspectos para definir los aquellos mecanismos de protección de la información de un sistema que almacena el documento electrónico y digital de forma segura[51].

También menciona que se debe validar de forma estructurada los aplicativos de software libre para que garanticen la confidencialidad, integridad y disponibilidad de la información, por medio de la ejecución de pruebas que permitan identificar fallos de seguridad, buscando diferentes mecanismos de protección de la información.

Otra ponencia es la de Diego Osorio donde enfatiza que:

> Se hace necesario abordar la seguridad como un aspecto de gran importancia durante el proceso de desarrollo y se propone una metodología de análisis, desarrollo y pruebas en busca de corregir vulnerabilidades a tiempo durante todo el desarrollo del software haciendo referencia a la Guía de Pruebas de OWASP, aplicando estándares, métricas y técnicas, con el fin de revisar el código dependiendo el contexto en que se apliquen[52].

Describe que dentro del ciclo de vida del desarrollo de software se debe implementar planes de mediciones para evaluar aspectos de vital importancia como la seguridad, estructura del desarrollo con funciones seguras, validación de riesgos y la identificación de vulnerabilidades.

---

[51] CAUCALI, Diana. Análisis y definición de requisitos de seguridad informática fundamentado en OWASP para el cumplimiento en los aplicativos basados en software libre de gestión documental. [En línea].2020, [Consultado 24 de octubre de 2022]. Disponible en < https://repository.unad.edu.co/jspui/bitstream/10596/38709/1/dmcaucalib.pdf>

[52] OSORIO, Diego. Plan y pruebas de seguridad de un sistema de gestión documental tomando como referencia la guía de pruebas de OWASP. [en línea]. 2020, junio, [Consultado 24 de octubre de 2022]. Disponible en < https://repository.udistrital.edu.co/bitstream/handle/11349/25722/OsorioGutierrezDiegoAndres2020.pdf?sequence=1&isAllowed=y>

En la búsqueda de la operación segura del software, se debe validar las funciones y controles de seguridad de este, para que sean mitigados aplicando un modelamiento de amenazas y análisis de código, Girón y Torres manifiestan que:

> La revisión de código fuente es el proceso de comprobar manualmente el código fuente de una aplicación web en busca de incidencias de seguridad. Muchas vulnerabilidades de seguridad serias no pueden ser detectadas con comprobar software cerrado de terceras partes, como sistemas operativos. Cuando se realizan pruebas en aplicaciones web (especialmente cuando han sido desarrolladas internamente), el código fuente debería ser puesto a disposición para comprobarlo[53].

Queda claro que en el ciclo de vida de los sistemas de información se debe tener una perspectiva de mejora continua que permita estar en una constante optimización, por medio del seguimiento, monitoreo y análisis de aseguramiento del sistema, esto conforme a las necesidades de la organización[54].

El análisis, ejecución de pruebas y los ajustes que se realicen al software que soporta un sistema de información, se convierten en las herramientas de primera línea para buscar la seguridad minina con el objetivo de mitigar el riesgo.

---

[53] GIRON, L y TORRES H. Detección y generación de recomendaciones para el cierre de las vulnerabilidades relacionadas para el cierre de las vulnerabilidades relacionadas con el top 10 OWASP identificadas en la aplicación web de historias clínicas en la institución prestadora de servicios de salud especializada en audiología audicom IPS. [en línea].2015, [Consultado 24 de octubre de 2022]. Disponible en: < http://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/2861/Trabajo%20de%20grad.pdf?sequence=1&isAllowed=y>

[54] Tramullas, Jesus. Los sistemas de información: Una reflexión sobre información, sistema y documentación. Revista general de información y documentación [en línea]. Universidad Computence (Madrid): Universidad de Zaragoza, 1997, vol. 7, N° 1, p. 207-229. [Consulta: 24 de octubre 2022]. Disponible en:< http://eprints.rclis.org/23751/1/11876-11957-1-PB.PDF>

5. ENFOQUE METODOLOGICO

Este proyecto se desarrolló bajo un enfoque de proyecto aplicado a través de una metodología de tipo experimental y descriptiva, entendiendo éstas como

> El proceso mediante el cual se manipula una variable experimental, en condiciones de riguroso control, para descubrir de qué modo y por qué causa se produce una situación o acontecimiento particular. La experimentación consiste en modificar, bajo cuidadoso control, las condiciones de un hecho y en observar e interpretar los cambios que ocurren en este último. Para lograr la interpretación de algo que se encuentra representado a través de datos, específicos sobre las propiedades de personas, grupos, comunidades o cualquier otro fenómeno sometido a análisis. Es el descubrimiento de los hechos seguido de la interpretación correcta del significado o importancia que se describe. [55]

En este entendido el desarrollo del enfoque se ejecutará por faces o etapas las cuales comprenden:

Fase I. Fase de Recepción de Código Fuente:

Recepción del código Fuente: Por convenio con el Departamento Entidad, quienes suministraran la entrega a título gratuito dicho código, instructivos y manuales del Gestor Documental ORFEO, para ajustar a la medida de la necesidad de la Autoridad.

Fase II. Fase de Alojamiento de Código Fuente:

Carga del código Fuente en los servidores de ambiente de pruebas para gestionar el análisis y desarrollo a la medida.

[55] CAMACHO DE BÁEZ, Briceida. Metodología. [en línea]. 2008 [Consultado el 25 de octubre de 2022]. Disponible en <repositorio.uptc.edu.co >

Fase III. Fase de Evaluación de Herramienta para Análisis:

Revisión Código Fuente: Validación manual de binarios y fuentes del código a los lenguajes de programación.

Evaluación de Herramienta de Análisis: Validar las diferentes herramientas de análisis de vulnerabilidades de código fuente, por medio de criterios basados en OWASP.

Fase VI. Fase de Ejecución de Pruebas:

Fase de Preparación Ambiente Para Ejecución De Pruebas SAST: Preparación de Ambiente software, servidores, puertos y demás para pruebas Dinámicas del código fuente.

Fase de Ejecución de Pruebas SAST: Lanzamiento de la herramienta de análisis DAST sobre el código fuente del proyecto ORFEO.

Fase de Preparación Ambiente Para Ejecución De Pruebas DAST: Preparación de Ambiente software, servidores, puertos y demás para pruebas Dinámicas del código fuente.

Fase de Ejecución de Pruebas: Lanzamiento de la herramienta de análisis DAST sobre el código fuente del proyecto ORFEO.

En las diferentes fases de ejecución del proyecto se articulará con la metodología de desarrollo seguro DevSecops.

6. DESARROLLO DE LOS OBJETIVOS

6.1 DESARROLLO DEL OBJETIVO 1.

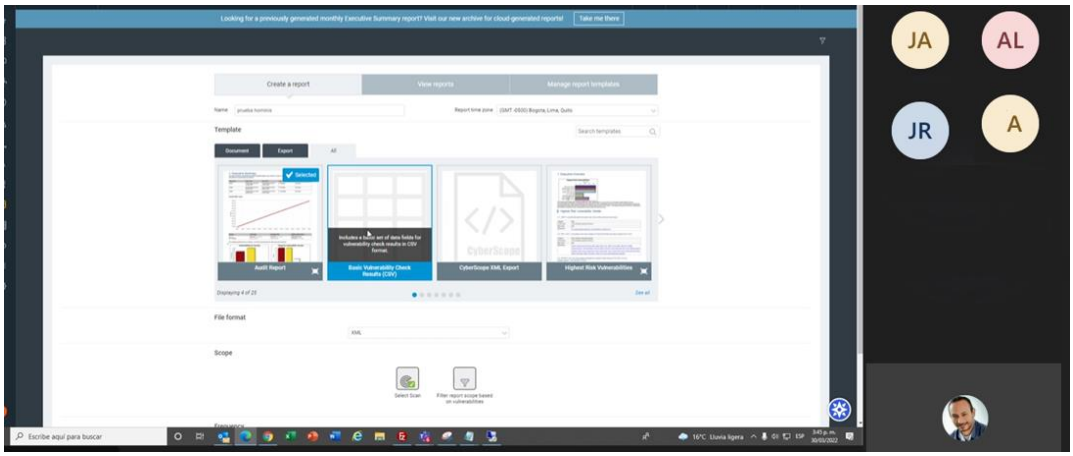
Analizar las vulnerabilidades del sistema de información de gestión documental Orfeo en la infraestructura tecnológica de la Autoridad por medio de la ejecución de pruebas de seguridad (SAST) y (DAST) en el marco de las etapas del DevSecOps.

6.1.1 Análisis de vulnerabilidades Orfeo.

Para la ejecución del respectivo análisis, se planearon diversas actividades que permitieron cumplir paso a paso con el planteamiento de este enmarcado en las etapas del DevSecOps.

6.1.1.1 Socialización Del Proyecto. Como primera actividad se gestaron los respectivos permisos y trámites internos para llevar a cabo el proyecto aplicado en la entidad. Se programó una reunión con el personal de seguridad informática, infraestructura y sistemas de información, para notificar y coordinar la operación del proyecto, definiendo el alcance y los objetivos para la actividad, como se muestra en la ilustración 2.

Ilustración 2. Socialización Proyecto



Fuente 2. Elaboración propia, captura de pantalla.

Como conclusión de la reunión se estableció que en el alcance del proyecto se debe contemplar la inclusión del análisis y proyección de la infraestructura con tratamiento de alta disponibilidad, esto, con miras a mitigar la indisponibilidad y denegación de servicio para los usuarios con la información crítica.

6.1.1.2 Topología Alta Disponibilidad Orfeo. Para dar cumplimento a la solicitud expuesta por el personal de infraestructura se plantea una topología de red según los recursos y proyección de la solución, entendiendo la alta disponibilidad como la implementación que puede garantizar un nivel de disponibilidad y continuidad de la operación en un trascurso de tiempo esperado en el desempeño y atención a las múltiples peticiones de los usuarios en el uso del servicio web[56].

Esta validación arrojó que la carga operacional de la solución de gestor documental ORFEO se identifica en las peticiones web y el servicio de combinador que es quien identifica marcaciones específicas en una plantilla de formato .docx y asigna firmas, número de radicado, fecha, entre otros.

En la topología se identifica la infraestructura de alta disponibilidad la cual consta de los siguientes recursos:

Balanceador de carga Front-end: Este cumplirá la gestión de asignación de carga operativa a los dos servidores Web (front-end) de todas las peticiones de los usuarios, funcionará con el algoritmo Rond Robin.

Servidores web Front-end: Estos dos servidores cumplen la gestión de las solicitudes asignadas por el balanceador consumiendo los servicios requeridos para la operación de la radicación de los documentos, bien sean físicos o electrónicos.

Servidor Web Services: Este gestiona o expone los servicios trasversales para que lo recursos solicitados sean direccionados a los combinadores, bodega o web.

[56] SINISTERRA, María Mercedes; HENAO DIAZ, Tania Marcela y LÓPEZ RUIZ, Erik Giancarlo. Clúster de balanceo de carga y alta disponibilidad para servicios web y mail. Informador técnico, [En línea]. Universidad de la Rioja, 2012. [Consultado 15 de mayo 2022]. Disponible en <http://Dialnet-ClusterDeBalanceoDeCargaYAltaDisponibilidadParaSer-4364562.pdfno 76, p. 93-102>

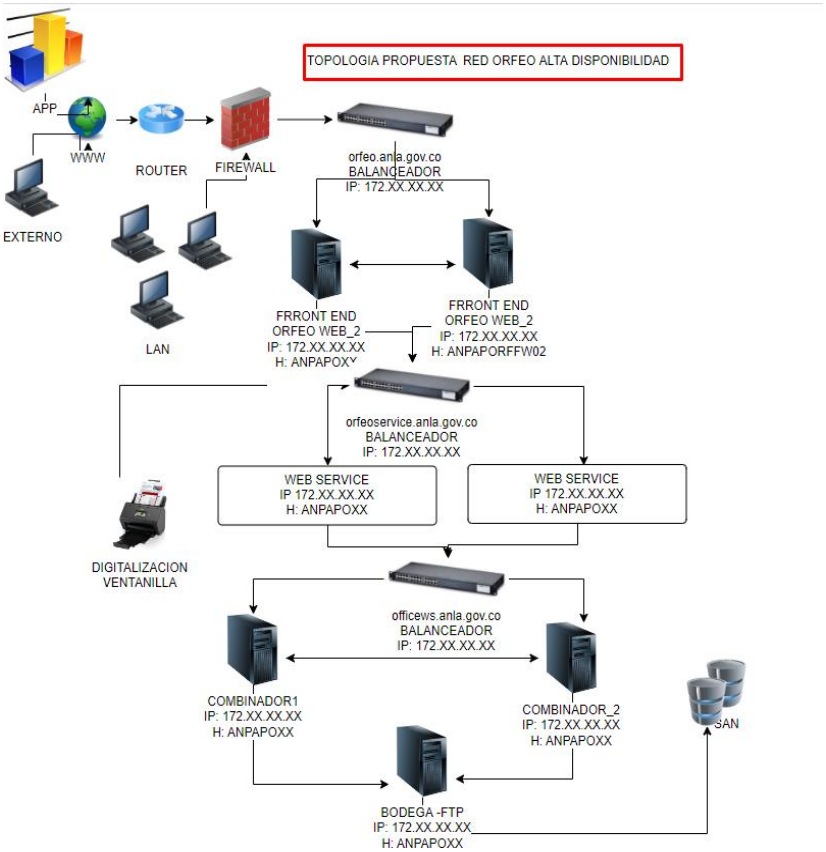
Balanceador de Carga Back-end: Estos dos servidores cumplen la gestión de las solicitudes asignadas por los servidores web direccionados por los servicios transversales y asignan la combinación de documentos bajo el algoritmo Round Robin.

Servidores Combinadores: Estos operan la transformación del documento en cualquier formato a .pdf, registrado los datos, metadatos y binarios necesarios para la radicación.

Servidor Bodega- FTP: Opera el almacenamiento de los expedientes según la TRD registrada y es donde se alojan los documentos radicados a los expedientes específicos. También cumple la función de carga de archivos con peso significativo por el servicio FTP usando el puerto seguro 990.

A continuación, se muestra la topología propuesta en la siguiente ilustración:

Ilustración 3. Topología Propuesta



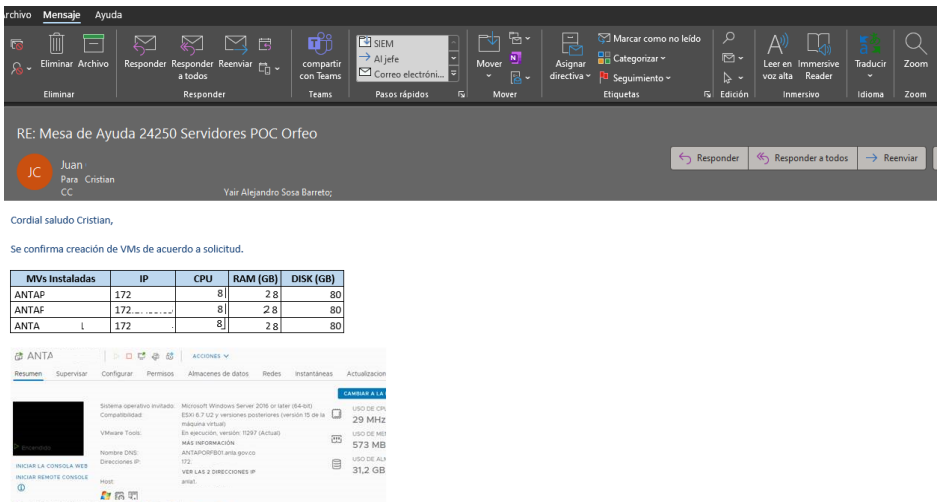
Fuente 3. Elaboración propia.

Esta propuesta se definió buscando solucionar la disponibilidad constante en la operación del aplicativo, ya que, si algún recurso tecnológico diseñado en la topología falla, se tendrá la seguridad que entrará en acción la línea alterna de backup y contar con recurrencia del recurso. Permitiendo garantizar el servicio 7 x 24.

6.1.1.3 Asignación De Infraestructura Orfeo. Como segunda actividad se reciben los servidores por parte del departamento de infraestructura, con sistemas operativos Windows server 2019, igualmente con las validaciones de accesos, permisos y direccionamiento IP del servidor establecido para la implementación del proyecto de ORFEO.

Esta actividad se convalidó con el personal de seguridad de la información bajo las políticas establecidas por la entidad y con las directrices del personal de infraestructura tecnológica. A estos servidores se carga el código fuente del aplicativo de gestión documental para la operación, como se muestra en la siguiente ilustración 4.

Ilustración 4. Correo Infraestructura



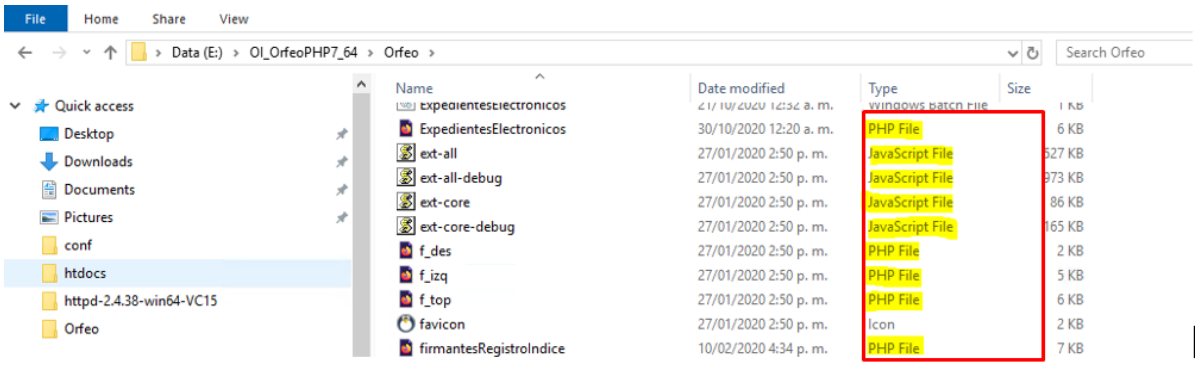
Fuente 4. Elaboración propia, captura de pantalla.

El código fuente, que por convenio interadministrativo se logró asociar con el Departamento Entidad, quienes suministraran la entrega a título gratuito dicho código, instructivos y manuales del Gestor Documental ORFEO. Este código reposará en el servidor asignado para tal fin y se ejecutaran las pruebas y análisis definidos.

6.1.2 Evaluación de herramientas para el análisis

6.1.2.1 Revisión Del Código Fuente De Orfeo Teniendo el código en los repositorios de la entidad se da inicio al revisión manual, observando los archivos, paquetes y carpetas donde está contenido, validando las extensiones de estos, en este proceso es de vital importancia aclarar que el software de gestión documental entregado a la entidad es un código espagueti, el cual es de comprensión compleja, donde no se tiene una estructura ordenada en saltos de código y el cual ha tenido modificaciones, personalizaciones, ajustes a medida, muchas veces con mala estructuración, repeticiones innecesarias, deficiente paso de parámetros, escasa comprensión del paradigma orientado a objetos57; por tanto, presenta el acoplamiento de diversos lenguajes de programación entre ellos la raíz de Orfeo que es PHP compilado su binario con la versión 7.2.16, versión de apache 2.4.38, también se evidencia que se ejecutan líneas en java y .net. Esta temática es de gran impacto, ya que el correcto análisis del lenguaje en el que fue desarrollado o modificado permita la lectura de la herramienta y la evaluación de forma y efectiva el código fuente al momento de implementar las respectivas pruebas, como se muestra en la ilustración siguiente:

Ilustración 5. Análisis Código Fuente



Fuente 5.Elaboración propia, captura de pantalla.

[57] COMPAÑ ROSIQUE, Patricia, *et al*. Explicando el bajo nivel de programación de los estudiantes: (Ejemplar dedicado a: Investigación en Docencia Universitaria de la Informática) [En línea]. Universidad de Alicante, 2018. [Consultado del 15 de mayo 2022]. Disponible en <https://dialnet.unirioja.es/servlet/articulo?codigo=6264614>

6.1.2.2 Selección De Herramienta De Análisis Luego de la revisión manual del código fuente en el que está estructurada la solución de gestión documental ORFEO, se evidenció que cuenta con lenguajes PHP, Java, .Net y por tanto se limitan las opciones para la selección de herramientas con las que se puedan ejecutar las pruebas estáticas de código SAST, en el mercado se encuentran gran variedad de herramientas para ciertos lenguajes de programación o lecturas de varios lenguajes pero no abarcan todos en los que se desarrolló el sistema de información y que sean de código abierto un selecto grupo son de pago.

Se realizó una caracterización donde se encontró una gama de herramientas que cuentan con la versatilidad y funcionalidad que se ajustan a la necesidad de la entidad y con apoyo de las sugerencias de *Open Web Aplication Security Proyect* (OWASP) que es un proyecto de código abierto cuya finalidad es determinar y mitigar la inseguridad del software, presenta en su página web un análisis de herramientas en su página Source Code Analysis Tools en el enlace *https://owasp.org/www-community/Source_Code_Analysis_Tools*, donde se listan herramientas de análisis SAST, basados en varios criterios de selección:

1. Soporte de lenguaje de programación: La herramienta debe soportar varios lenguajes de programación para el análisis de versatilidad de los binarios.

2. Hallazgos de vulnerabilidades basados en top ten de OWASP: La gestión de vulnerabilidades de la herramienta seleccionada debe contener o consultar la DB de OWASP actualizada para estar alineada con las últimas tendencias de vulnerabilidades.

3. Tasa de Falsos Positivos: Esta debe permitir la gestión configuración o parametrización de falsos positivos en las reglas de análisis del código fuente.

4. Habilidad de incluir en herramientas de Integración/Implementación Continua: Debe ser una herramienta que permita escalabilidad e integración con otras aplicaciones para lograr la automatización de procesos.

5. Resultados de Análisis de Código Estático: Debe generar los resultados del análisis de forma entendible y con gestión de mitigación de los hallazgos.

En cuanto a las herramientas se validaron las sugeridas por (OWASP), las cuales menciona y entrega información específica para realizar la gestión con DevSecOps, entre ellas están:

Tabla 1. Herramientas de Escaneo

Nombre	Propietario	Licencia	Plataforma	Descripción
Bandit		Open Source		Bandit es un escáner de vulnerabilidades de código fuente completo para Python
Brekeman		Open Source		Brekeman es un escáner de vulnerabilidades específicamente diseñado para Ruby en aplicaciones Rails
Codesake Dawn		Open Source		Codesake Dawn es un escáner de vulnerabilidades específicamente diseñado para Ruby
FindBugs		Open Source		Escáner de vulnerabilidades de código fuente para Java.

Continuación Tabla 1.

Nombre	Propietario	Licencia	Plataforma	Descripción
FindSecBu		Open Source		Complemento de seguridad para SpotBugs que mejora capacidad de SpotBugs para hallar vulnerabilidades de seguridad en código Java.
Flawfinder		Open Source		Escáner para código desarrollado en C y C++
Graudit		Open Source	Linux	Escanea múltiples idiomas en busca de diversas vulnerabilidades de seguridad. Básicamente, es un Grep de código con mejoras de seguridad.
LGTM		Open Source		Valida las vulnerabilidades de leguajes como C, C++, Java y Python.
Progpilot		Open Source		Herramienta de análisis estático para PHP detecta vulnerabilidades como XSS e Inyección SQL.

Continuación Tabla 1.

Nombre	**Propietario**	**Licencia**	**Plataforma**	**Descripción**
PreFast	Microsoft	Open Source		Herramienta de análisis estático que identifica defectos en programas C/C++. Última actualización en 2006.
Puma Scan	Puma Security	Comercial		Analizador estático de código fuente en C# de .NET que se ejecuta como una extensión de Visual Studio IDE y extensión de Azure DevOps y un ejecutable de línea de comandos (CLI).
.NET *Security Guard*		Open Source		Escáner para código desarrollado .NET.
phpcs-security-audit		Open Source		Reglas de PHP_*CodeSniffer* para encontrar fallos o debilidades relacionadas en PHP y sus populares CMS o *frameworks*. Cuenta con reglas para el núcleo de para Drupal 7.

Continuación Tabla 1.

Nombre	**Propietario**	**Licencia**	**Plataforma**	**Descripción**
RIPS		Open Source		Escáner para código desarrollado PHP.
SonarQube		Open Source		Escanea código fuente en 15 lenguajes en busca de errores, vulnerabilidades y problemas de código. SonarQube tiene complementos para IDES como Eclipse, Visual Studio e IntelliJ.
VisualCode Grepeper		Open Source	Windows	Valida vulnerabilidades de varios lenguajes de programación entre ellos C/C++, C\#, VB, PHP, Java, PL/SQL[58].

Para la selección de la herramienta se escogió el criterio de cobertura de análisis a los lenguajes de programación, es decir se inclina la decisión por la herramienta que más lenguajes de programación pueda analizar, esto, debido al código fuente de ORFEO cuenta con su base de lenguaje con tres fuentes PHP, .net y java. Por tanto, la herramienta deber versátil en el análisis, se pondera con valor de 1 a cada lenguaje que la herramienta pueda analizar y 0 la que no cuenta con la cobertura, el resultado de la sumatoria seria máximo 7 y minino 1, en un listado de lenguajes entre ellos: Java, .NET, C, C++, Python, Golang, Ruby y PHP.

[58] OWASP. Source Code Analysis Tools [En Linea].[Consulta: 15 de mayo 2023]. Disponible en < https://owasp.org/www-community/Source_Code_Analysis_Tools>

Se realiza la evaluación y se evidencia que la ponderación más alta es para la herramienta SonarQube que registra la sumatoria máxima de 7 en la cobertura del análisis de los lenguajes de programación donde están incluidos los lenguajes desarrollados para el gestor documental ORFEO, por tanto, la herramienta a escoger es SonarQube, como se muestra en la siguiente ilustración.

Para la validación de la herramienta se establece la equivalencia de los símbolos

X = 0
√ = 1

Ilustración 6. Comparativo Herramientas SAST

Scanners	Java	.NET	C, C++	Python	Golang	Ruby/Rails	PHP	TOTAL
Bandit	X	X	X	√	X	X	X	1
Brakeman	X	X	X	X	X	√	X	1
Codesake Dawn	X	X	X	X	X	√	X	1
FindBugs	√	X	X	X	X	X	X	1
FindSecBugs	√	X	X	X	X	X	X	1
Flawfinder Flawfinder	X	X	√	X	X	X	X	1
Graudit	√	√	X	√	X	√	√	5
LGTM	√	X	√	√	X	X	X	3
Progpilot	X	X	X	X	X	X	√	1
PreFast (Microsoft)	X	X	√	X	X	X	X	1
Puma Scan	X	√	X	X	X	X	X	1
.NET Security Guard	X	√	X	X	X	X	X	1
RIPS	X	X	X	X	X	X	√	1
phpcs-security-audit	X	X	X	X	X	X	√	1
SonarQube	√	√	√	√	√	√	√	7
VisualCodeGrepper (VCG) -	X	X	√	X	X	X	√	2

Fuente 6.CLARANET, Herramientas open source para adoptar DevSecOps.

SonarQube es una herramienta o escáner estático con el cual se evalúa la confiabilidad, seguridad y mantenibilidad de los proyectos de desarrollo, con análisis de calidad y bajos falsos positivos. SonarQube tiene un desarrollo funcional como analizador de código, herramienta de

reportes, modulo con la detección de defectos en el código y la restauración de los cambios realizados en el código[59].

Otra herramienta en la cual se apoyará la gestión del proyecto es Scan Engine, este es un scanner de vulnerabilidades, desarrollado por la compañía Rapid 7 firma de seguridad fundada por el creador de Metasploit, su tecnología es On-Premis. Esta herramienta según la Página Web RAPID7 Scan Engine es un Software de gestión de vulnerabilidades, supervisa las exposiciones en tiempo real y se adapta a las nuevas amenazas con datos actualizados, lo que garantiza que siempre pueda actuar en el momento del impacto, por medio de sus cualidades sobre pasa a otros Scanner de vulnerabilidades, en especial su característica de Visión en Tiempo Real del Riesgo. Adicionalmente cuenta con análisis DAST[60].

Con la cual se buscará tratar las pruebas dinámicas, debido a que la entidad cuenta con la adquisición propia de Rapid 7 y licenciada de la herramienta, pero no se contaba con el lineamiento de análisis de seguridad en el código de las aplicaciones. Por tanto, esta será la segunda herramienta en la cual se realizarán las pruebas DAST.

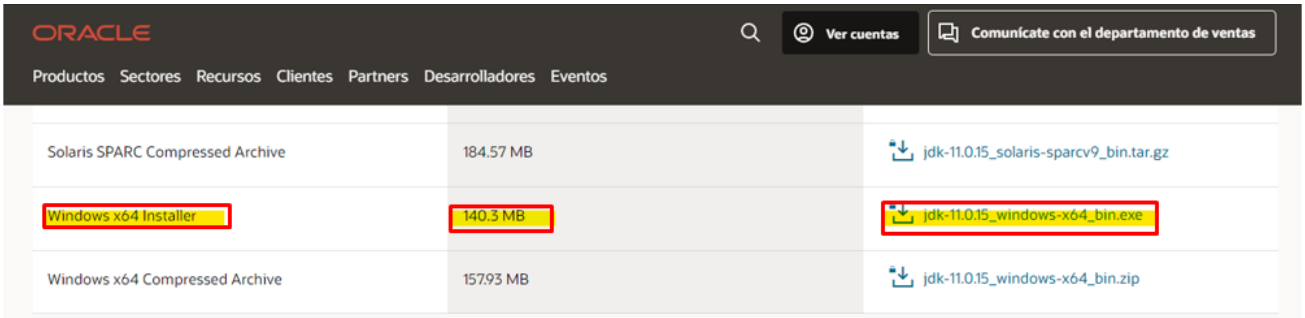
6.1.3 Ambiente Para Ejecución De Pruebas SAST. Para aplicar el análisis del código con la herramienta SonarQube, se deben realizar parametrizaciones, instalaciones y configuraciones con el propósito de afinar las herramientas que interactúan entre ellas y garantizar su correcta ejecución, esto está sujeto al sistema operativo donde reposa el código fuente.

6.1.3.1 Instalación De Software SAST. Para la ejecución de SonarQube se deben instalar varias herramientas funcionales que permiten la validación de las pruebas, entre ellas la versión de Java JDK 11.0.15 que se puede obtener de la página de Oracle e instalar de forma normal como se muestra en la siguiente ilustración 7. Para este ejercicio se instala la versión de Windows a 64 bits, ya que es el sistema operativo con el cual se ejecutará el análisis del código.

[59] OSPINA DELGADO, Juan Pablo. Análisis de seguridad y calidad de aplicaciones (Sonarqube) [En línea]. Tesis Maestría. Universidad Obterta de Cataluyna, 2015. [Consultado 15 mayo 2022]. Disponible en < http://openaccess.uoc.edu/webapps/o2/handle/10609/43263>

[60] RAPID7. Escáner de vulnerabilidades Nexpose. [en línea]. [Consulta: 15 de mayo 2022]. Disponible en <https://www.rapid7.com/products/nexpose>

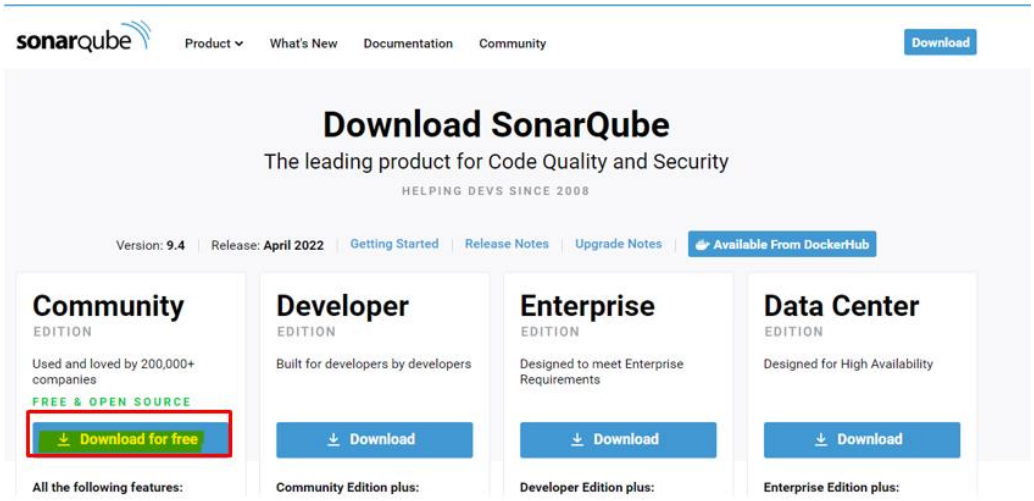
Ilustración 7. JDK 11



Fuente 7. ORACLE, Java Downloads

Luego se procede a instalar el SonarQube versión 9.4.0.5442 igualmente para la versión de Windows, como se observa en la ilustración 8, el cual instalará un servidor web local que permitirá visualizar los resultados del análisis.

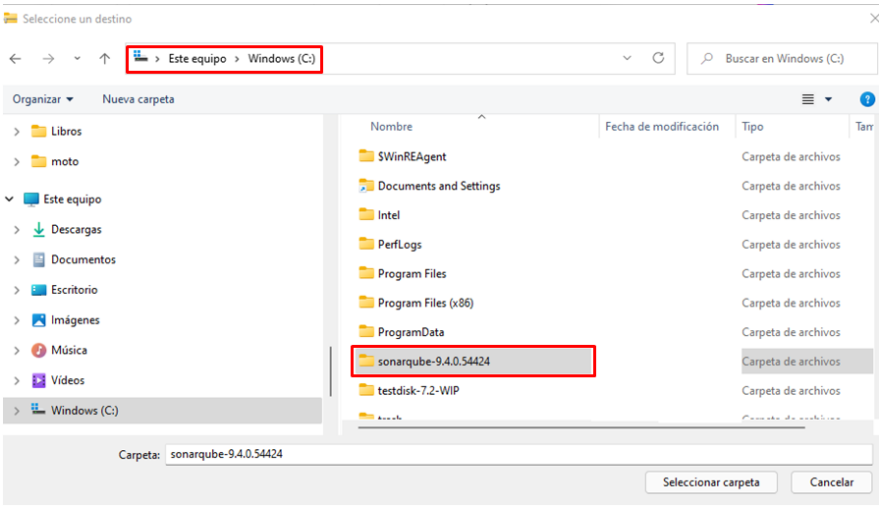
Ilustración 8. Descarga SonarQube



Fuente 8. SONARQUBE, Download SonarQube

Se procede con la descarga del archivo .zip se debe descomprimir en la partición C de Windows con el nombre y versión de la herramienta, como se evidencia en la ilustración 9.

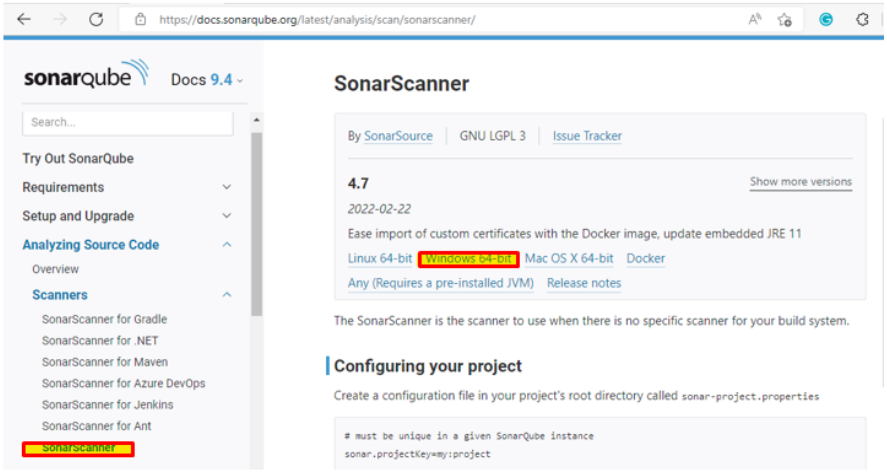
Ilustración 9. Instalación SonarQube



Fuente 9.Elaboración propia, captura de pantalla

Seguidamente se instala la función de escáner para que valide y cargue al servidor web local el análisis respectivo del código. La versión del software de escáner es la 4.7.0 descargado de la página documental del SonarQube, como se aprecia en la ilustración 10.

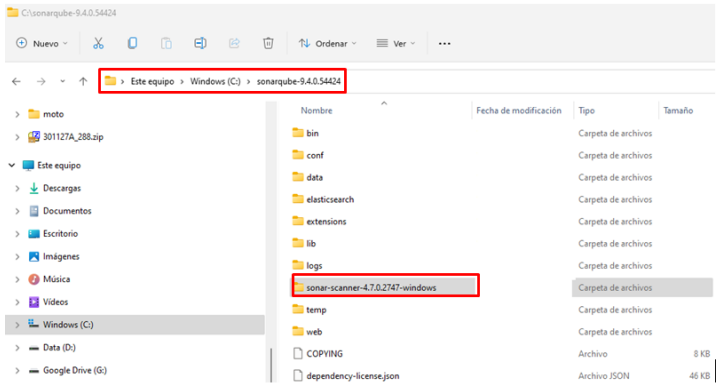
Ilustración 10. Sonar Scanner



Fuente 10. SONARQUBE, Download SonarQube. [en línea]. [Consultado el 15 de mayo de 2022] Disponible en:<www.sonarqube.org/download>

Se descomprime el software en el repositorio creado anteriormente C:\sonarqube-9.4.0.54424\sonar-scanner-4.7.0.2747-windows, puesto que se debe configurar las variables de entorno para la ejecución de las herramientas, como se observa en la ilustración 11.

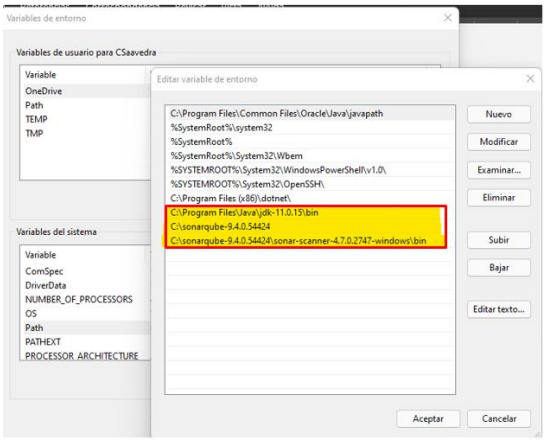
Ilustración 11. Instalación Sonar Scanner



Fuente 11.Elaboración propia, captura de pantalla

Ahora se debe configurar las variables de entorno en el sistema operativo para que la ejecución de análisis de SAST sea exitosa, se agrega la ruta de la carpeta al PATH del sistema operativo para que la consulte, como se muestra en la ilustración siguiente.

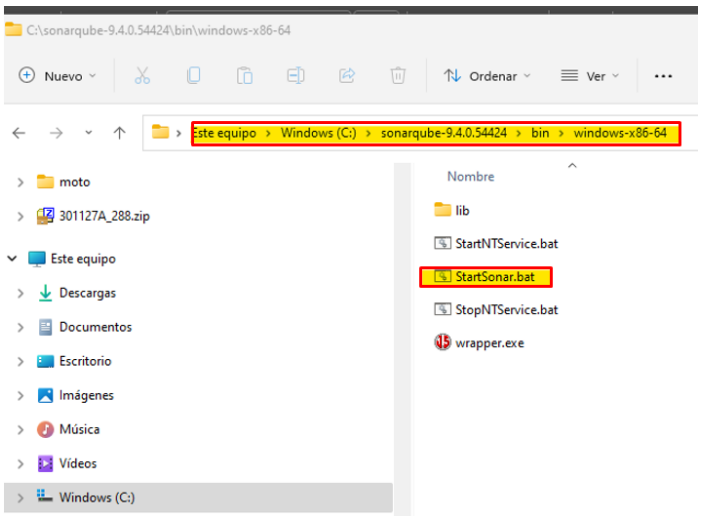
Ilustración 12. Variables de Entorno



Fuente 12.Elaboración propia, captura de pantalla

6.1.4 EJECUCIÓN DE ANÁLISIS SAST SONARQUBE. Continuando con el proceso se debe ejecutar el archivo StarSonar.bat extraído en la ruta C:\sonarqube-9.4.0.54424\bin\windows-x86-64 para lanzar el servicio web, como se valida en la ilustración 13.

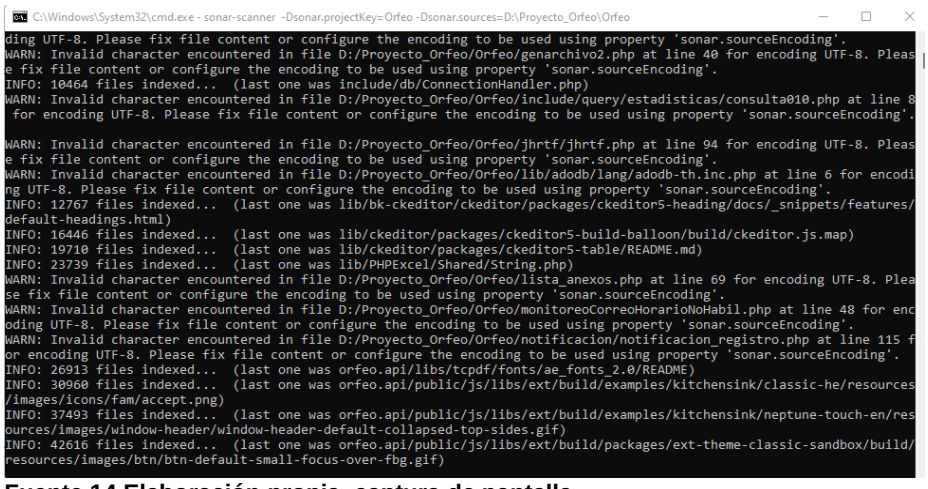
Ilustración 13. Bat Servicio Web



Fuente 13.Elaboración propia, captura de pantalla

Se ejecuta el lanzador en java e instala el servidor web de SonarQube, como se evidencia en la ilusración 14.

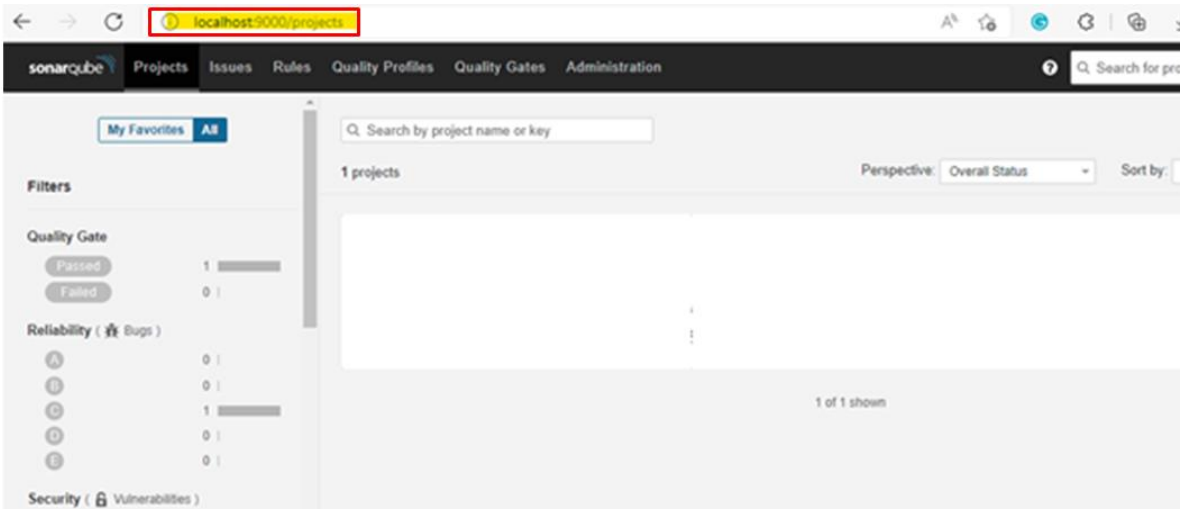
Ilustración 14. Lanzar SonarQube Web



Fuente 14.Elaboración propia, captura de pantalla

Ahora se valida el acceso al servicio con el localhost:9000, como se ve en la ilustración 15.

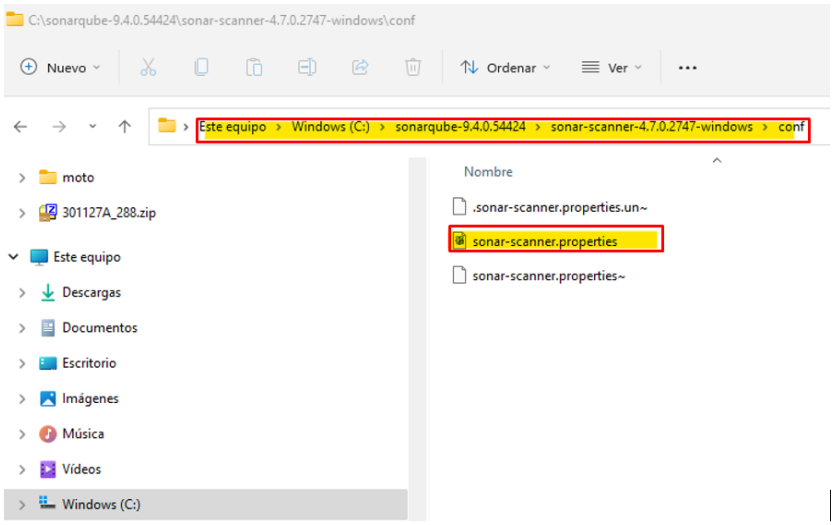
Ilustración 15. Servicio Web



Fuente 15.Elaboración propia, captura de pantalla

Se debe configurar el Sonar Escáner para la ejecución de las pruebas en el código en reposo de ORFEO, para ello se ingresa al repositorio C:\sonarqube-9.4.0.54424\sonar-scanner-4.7.0.2747-windows\conf\sonar-scanner.properties, cómo se referencia en la ilustración 16.

Ilustración 16. Sonar Scanner Properties



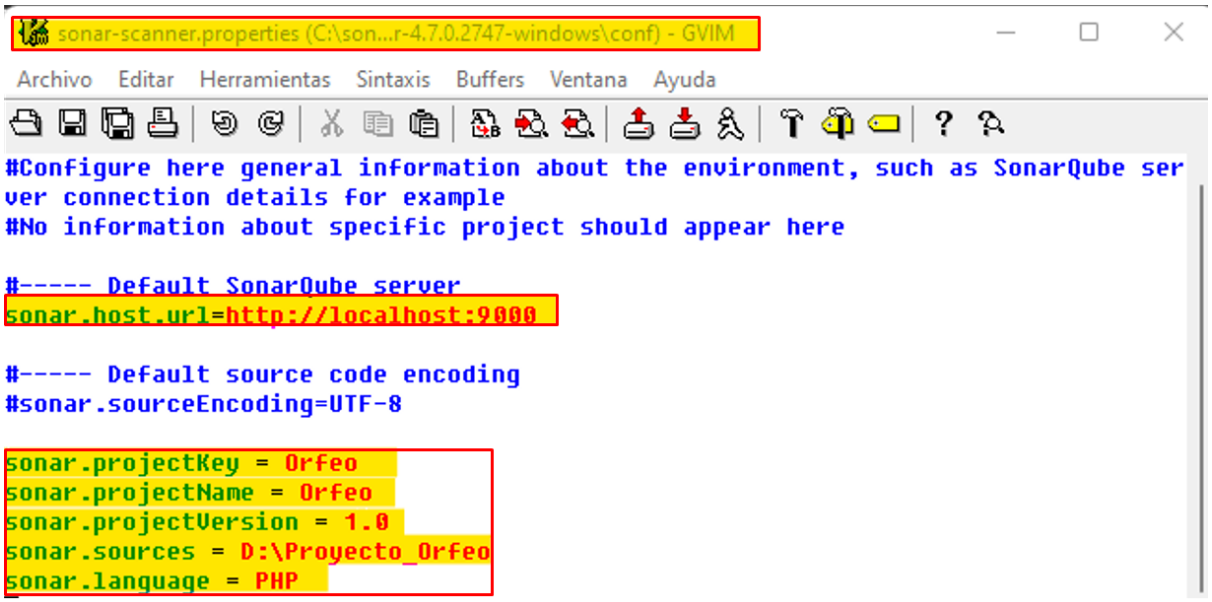
Fuente 16.Elaboración propia, captura de pantalla

Luego se debe parametrizar el archivo sonar-scanner.properties con la siguiente configuración:

sonar.host.url=http://localhost:9000 (Dirección de acceso al servicio Web)
sonar.projectKey = Orfeo (Donde hace referencia a la calve única del proyecto a analizar)
sonar.projectName = Orfeo (Refleja el nombre del Proyecto a gestionar)
sonar.projectVersion = 1.0 (Versionamiento del Código Fuente)
sonar.sources = D:\Proyecto_Orfeo (Repositorio del Código a analizar)
sonar.language = PHP (lenguaje de programación del Proyecto)

Se puede apreciar en la ilustración 17.

Ilustración 17. Parametrización Scanner.properties

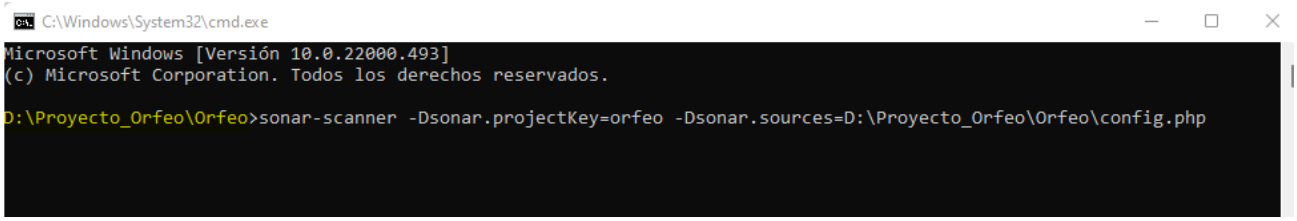


Fuente 17.Elaboración propia, captura de pantalla

Ahora se debe acceder al repositorio donde está alojado el código fuente por cid y ejecutar la línea de comandos:

sonar-scanner -Dsonar.projectKey=orfeo (Donde esta es la misma llave de configuración en el archivo.propierties)
-Dsonar.sources=D:\Proyecto_Orfeo\Orfeo\ (Es la ubicación del repositorio del código fuente) como se observa en la ilustración 18.

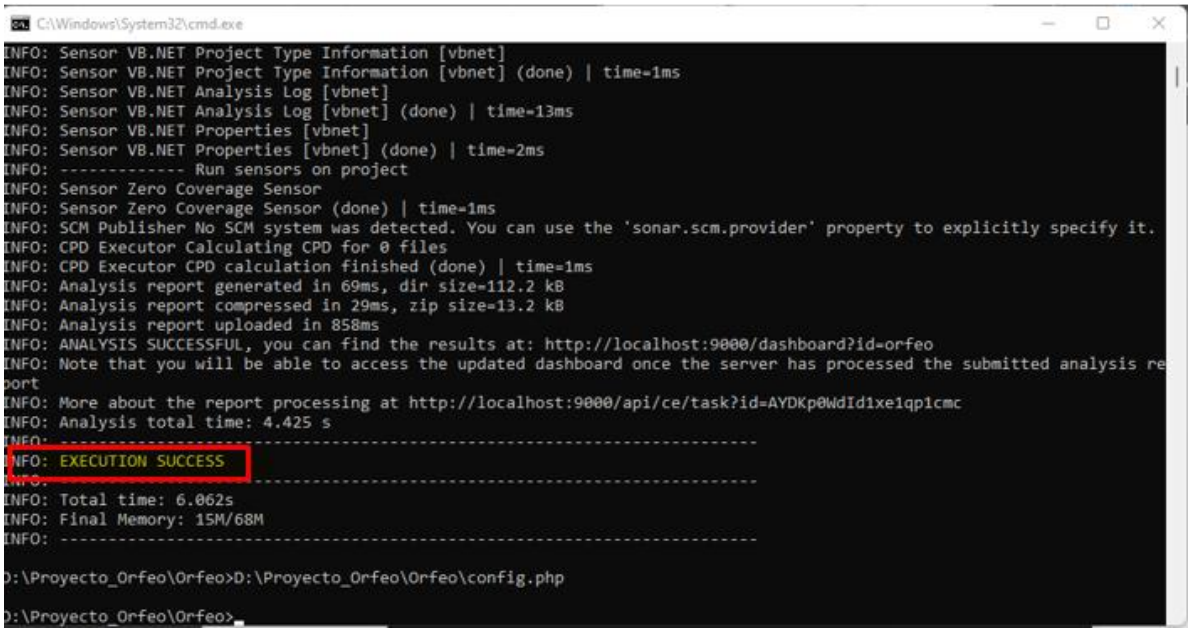
Ilustración 18. Lanzamiento de Escáner



Fuente 18.Elaboración propia, captura de pantalla

Se observa la ejecución correcta de la prueba SAST y enviándola al servidor web, como se aprecia en la ilustración 19.

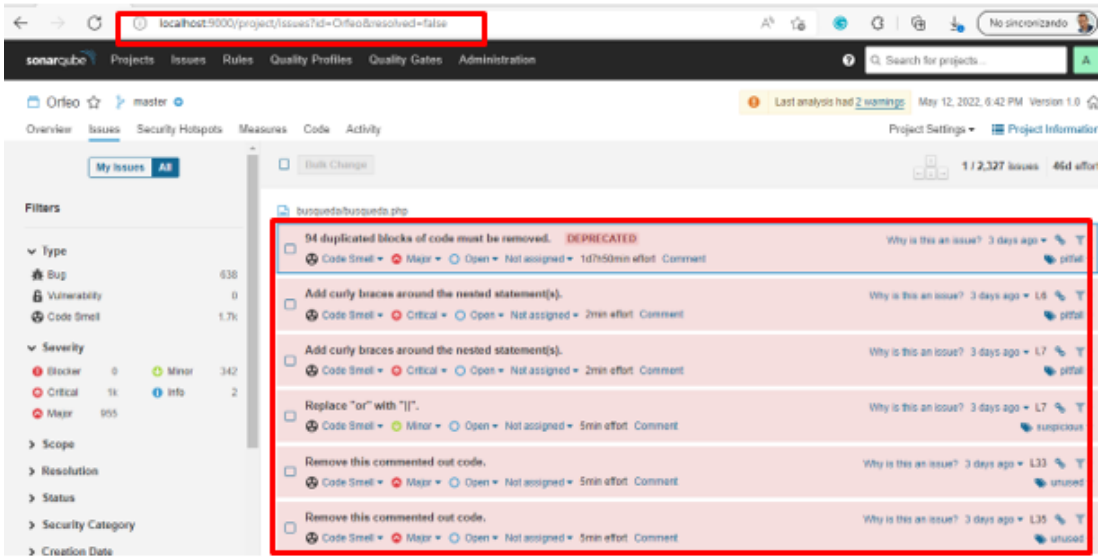
Ilustración 19. Ejecución Escáner Sonar



Fuente 19.Elaboración propia, captura de pantalla

Ahora se remite a la URL http://localhost:9000/projects y se visualizará el respectivo análisis del código, como se observa en la ilustración 20.

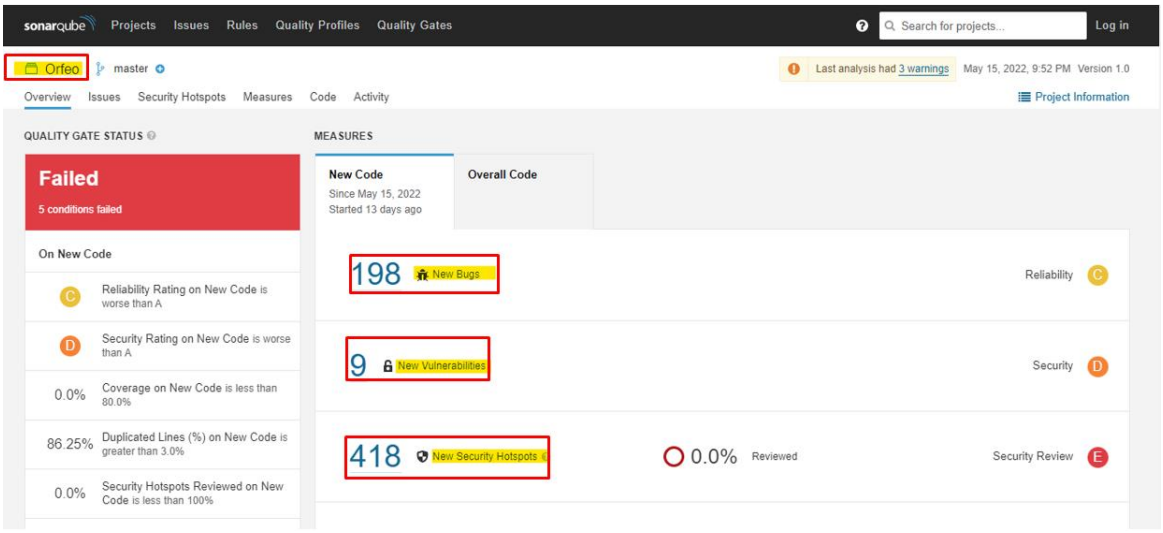
Ilustración 20. Servidor Web



Fuente 20.Elaboración propia, captura de pantalla

Los resultados del análisis arrojan el siguiente esquema, que se aprecia en la ilustración 21.

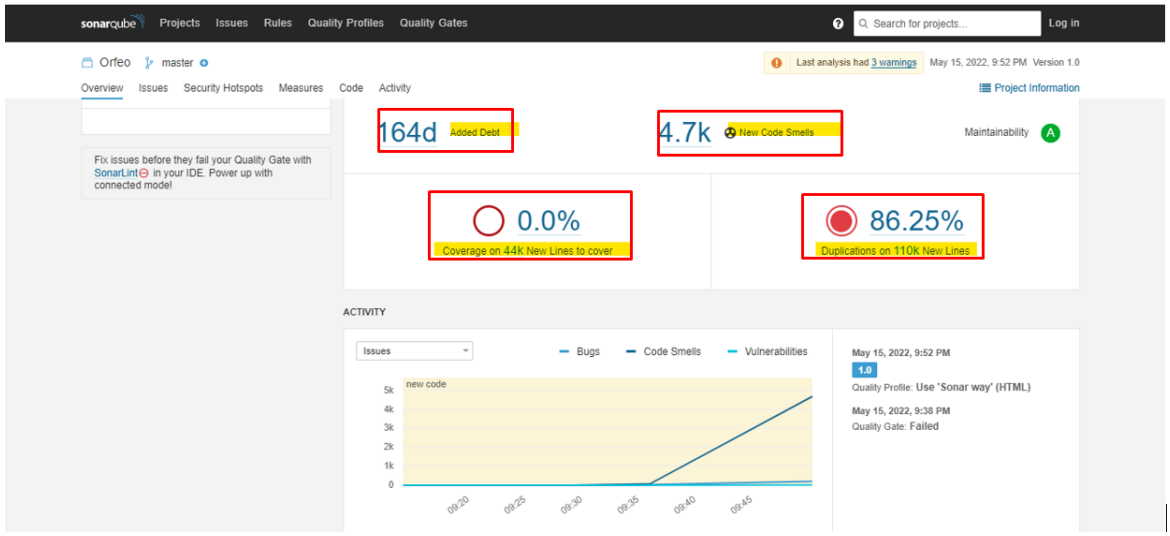
Ilustración 21. Resultados de Análisis



Fuente 21.Elaboración propia, captura de pantalla

Se observan los gráficos de los resultados de ejecución del análisis dejando ver los siguientes porcentajes como se aprecia en la ilustración 22.

Ilustración 22. Resultados de Análisis 2



Fuente 22.Elaboración propia, captura de pantalla

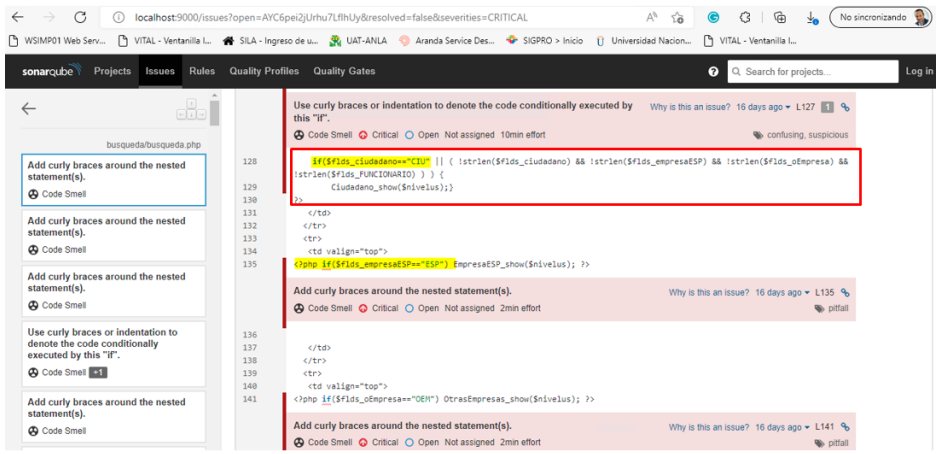
Se evidencia 198 Bugs, 9 vulnerabilidades y 418 puntos de acceso de seguridad, 164 deudas técnicas o falta de mantenimiento del código, errores en las líneas de código y presenta un 86.25 % de duplicidad en el código.

6.1.5 Listado de Vulnerabilidades Encontradas SAST. La herramienta SonarQube realiza las validaciones del código estático, por reglas contenidas ya establecidas que ayudan a identificar determinadas funciones asociadas a vulnerabilidades de código, cabe aclarar que SonarQube integra funciones de análisis con los parámetros de OWASP y MITRE, garantizando que sea un análisis funcional y actualizado, aunque se puede realizar una parametrización más personalidad de las reglas de análisis, para este efecto se dejara por defecto.

Para la compresión del análisis realizado por la herramienta SonarQube, a continuación, se enunciarán las vulnerabilidades encontradas, donde se relaciona su descripción general y la correlación de estas en el código fuente, las cuales tenemos:

6.1.5.1 Add curly braces around the nested statement(s) Como se evidencia en la ilustración 23, la mala práctica de no cerrar las sentencias o ciclos if con las llaves o corchetes {}, esto permite que si se ingresa una nueva declaración el código generar error y se romperá de forma inesperada.

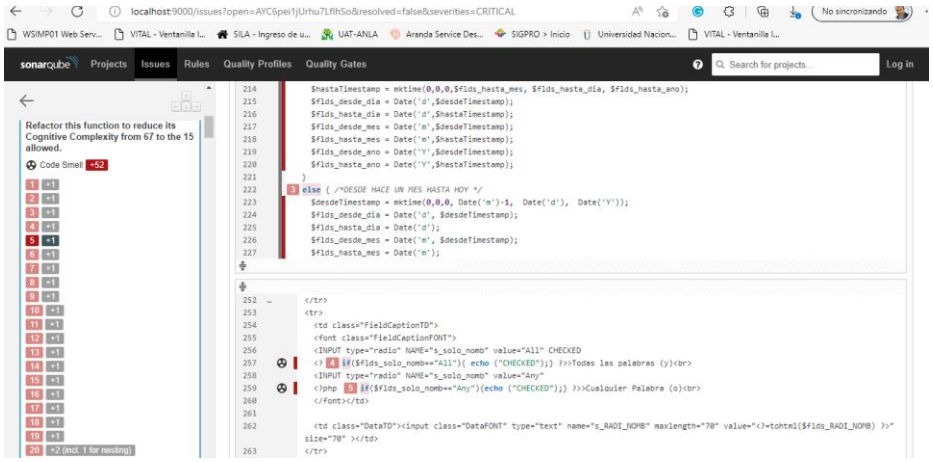
Ilustración 23. Add curly braces



Fuente 23.Elaboración propia, captura de pantalla

6.1.5.2 Refactor this function to reduce its Cognitive Complexity Se evidencia que esta función se puede optimizar y reducir la cantidad de líneas de código debido a esto ejerce más procesamiento y consumo de recursos para recorrer el flujo del código, como se observa en la ilustración 24.

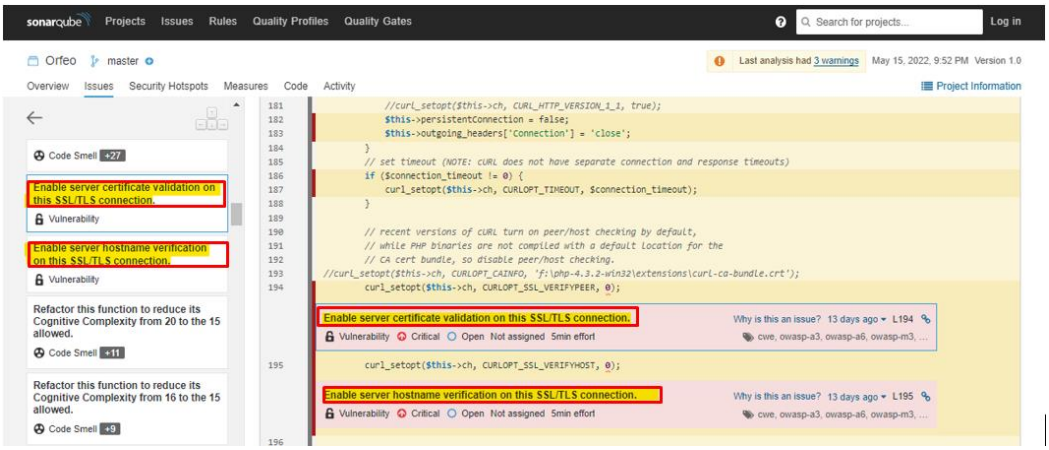
Ilustración 24. Refactor Cognitive Complexity



Fuente 24.Elaboración propia, captura de pantalla

6.1.5.3 Enable server certificate validation on this SSL/TLS. El análisis permitió ver vulnerabilidades y específicamente que se debe habilitar la validación de del certificado de conexión SSL/TLS del servidor, ya que de no estarlo los datos enviados entre el servidor y el cliente pueden ser manipulados o interceptados, como se valida en la ilustración 25.

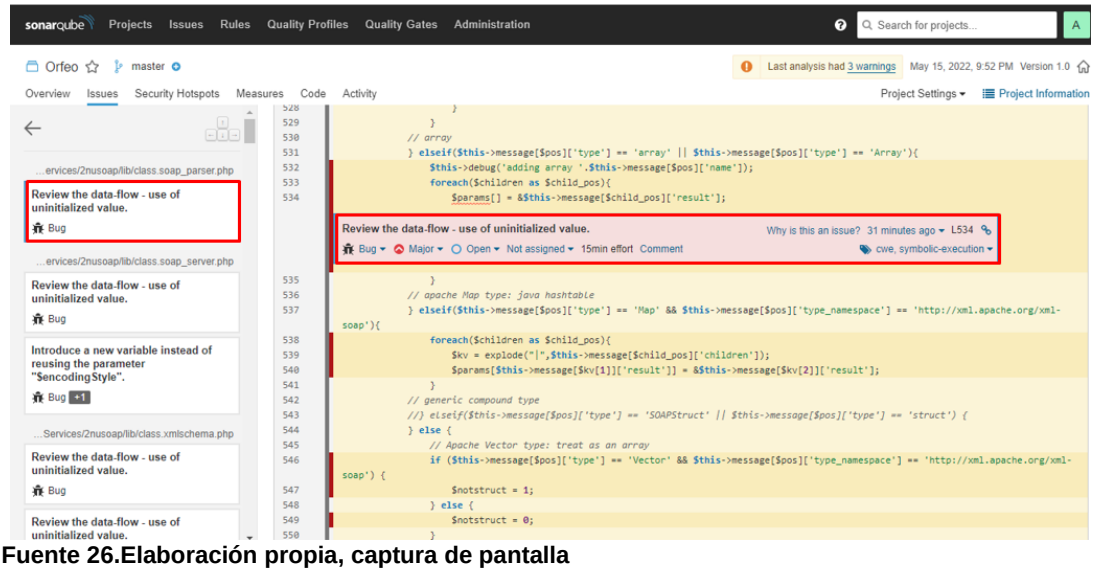
Ilustración 25. Server Certificate SSL/TLS



Fuente 25.Elaboración propia, captura de pantalla

6.1.5.4 Bugs. Como se refleja en la ilustración 26 se visualizan los bugs un ejemplo de ello es el de flujo de datos, donde el valor no está inicializado.

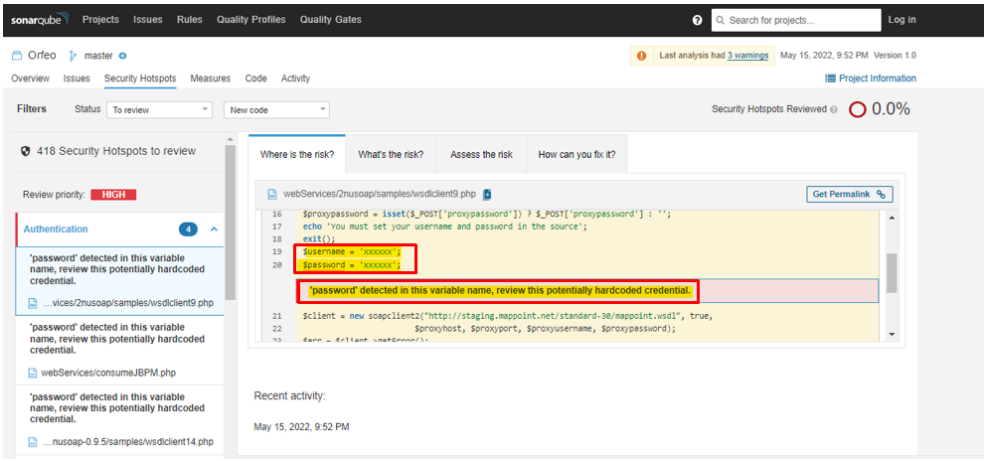
Ilustración 26. Bugs



Fuente 26.Elaboración propia, captura de pantalla

6.1.5.5 Password detected in this variable. Se refleja también vacíos de seguridad como variables nombradas como *password*, permitiendo en un posible ataque al capturarla, como se ve en la ilustración 27.

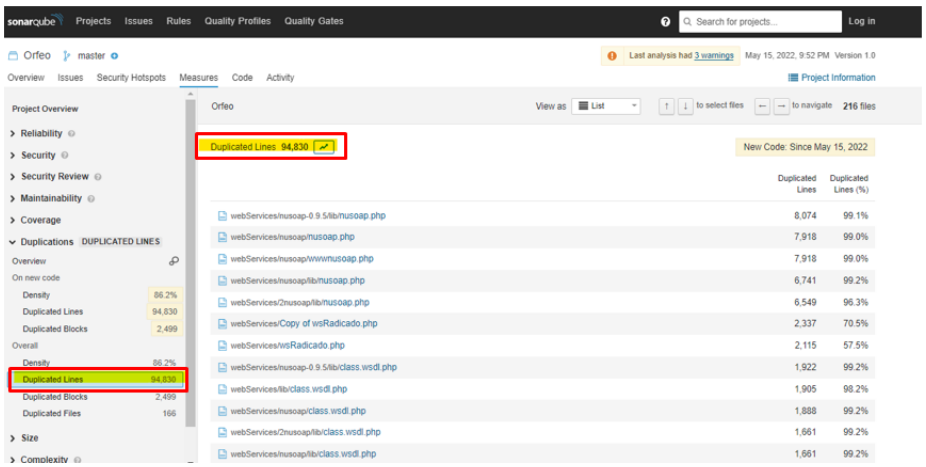
Ilustración 27. Password Detected in this Variable



Fuente 27. Elaboración propia, captura de pantalla

6.1.5.6 Duplicated Lines. Las líneas de código duplicadas o repetidas son una vulnerabilidad crítica por que genera un amento innecesario de código, ejecuta instrucciones con funciones que no son requeridas y refleja que el desarrollo no cuenta con un diseño ajustado ni apropiado, como se aprecia en la ilustración 28.

Ilustración 28. Duplicated Lines

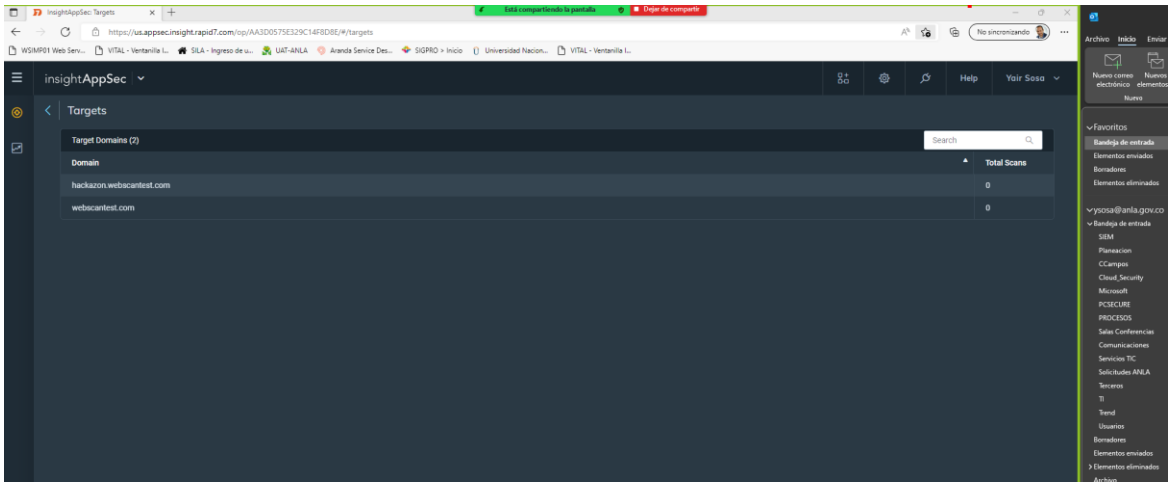


Fuente 28. Elaboración propia, captura de pantalla

6.1.6 Ambiente para Ejecución de Pruebas DAST. Para aplicar el análisis del código con la herramienta Scan Engine, se deben realizar parametrizaciones, instalaciones y configuraciones con el propósito de afinar las herramientas que interactúan entre ellas y garantizar su correcta ejecución, esto está sujeto al sistema operativo donde reposa el código fuente.

6.1.6.1 Instalación Software DAST. Para la ejecución de las pruebas (DAST) del código fuente, a continuación, se presenta la serie de pasos que son requeridos para el proceso de instalación y configuración del software, al igual que en las SAST se debe ajustar el ambiente, para ello se ingresa a la plataforma de rapid7 con la herramienta Scan Engine, descargando el agente que se instala en el equipo, para que se puedan consultar los resultados del análisis por medio de un servidor web, como se evidencia en la ilustración 29.

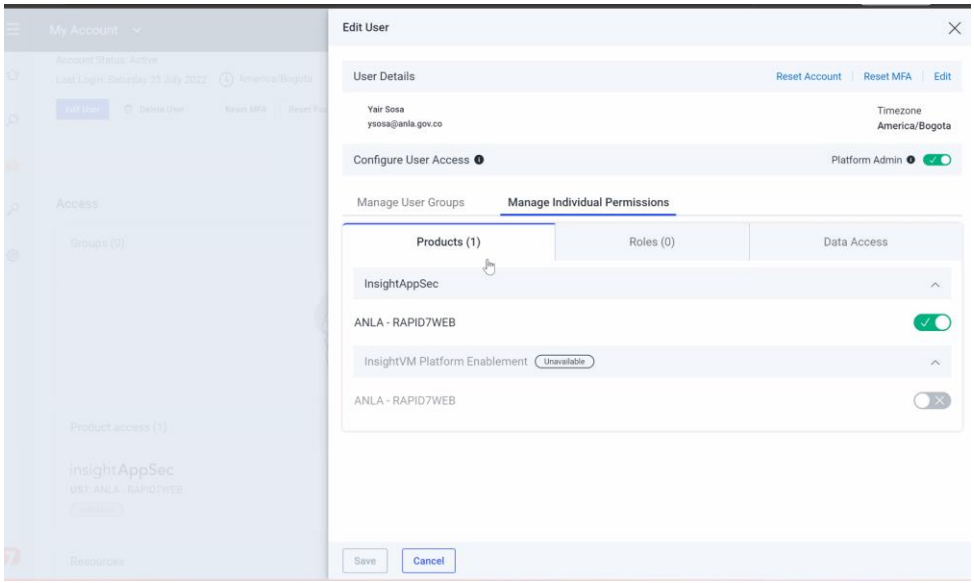
Ilustración 29. Acceso Descarga Agente



Fuente 29. Elaboración propia, captura de pantalla

Ahora se procede a realizar la configuración, validación y accesos de permisos adecuados para el usuario que administrara la plataforma web, como se observa en la ilustración 30.

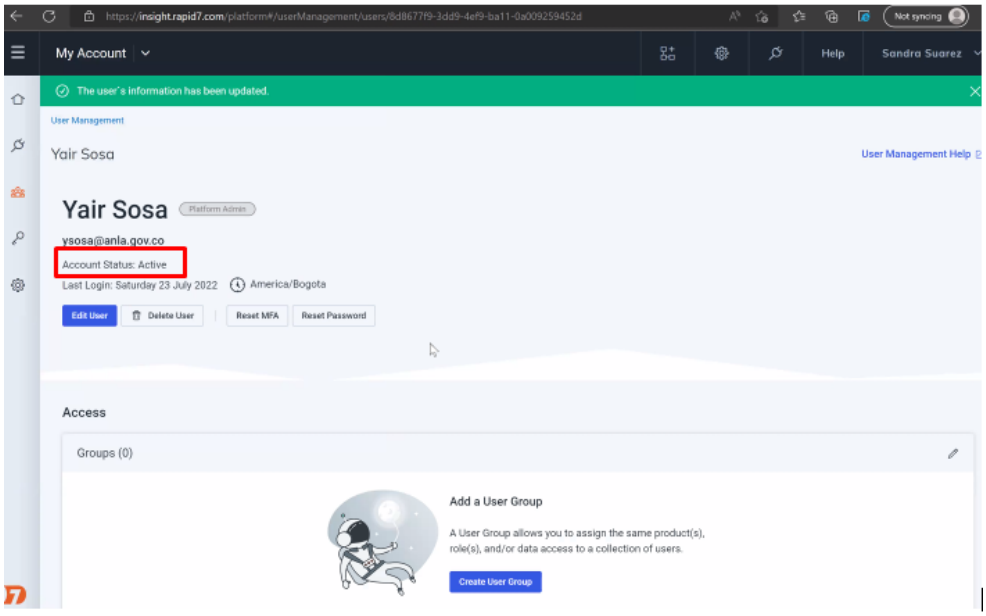
Ilustración 30. Permisos de Administración Web



Fuente 30. Elaboración propia, captura de pantalla

Contando con los accesos de usuario y contraseña se encuentren activos para la configuración, se valida el estatus estando activo, como se evidencia en la ilustración 31.

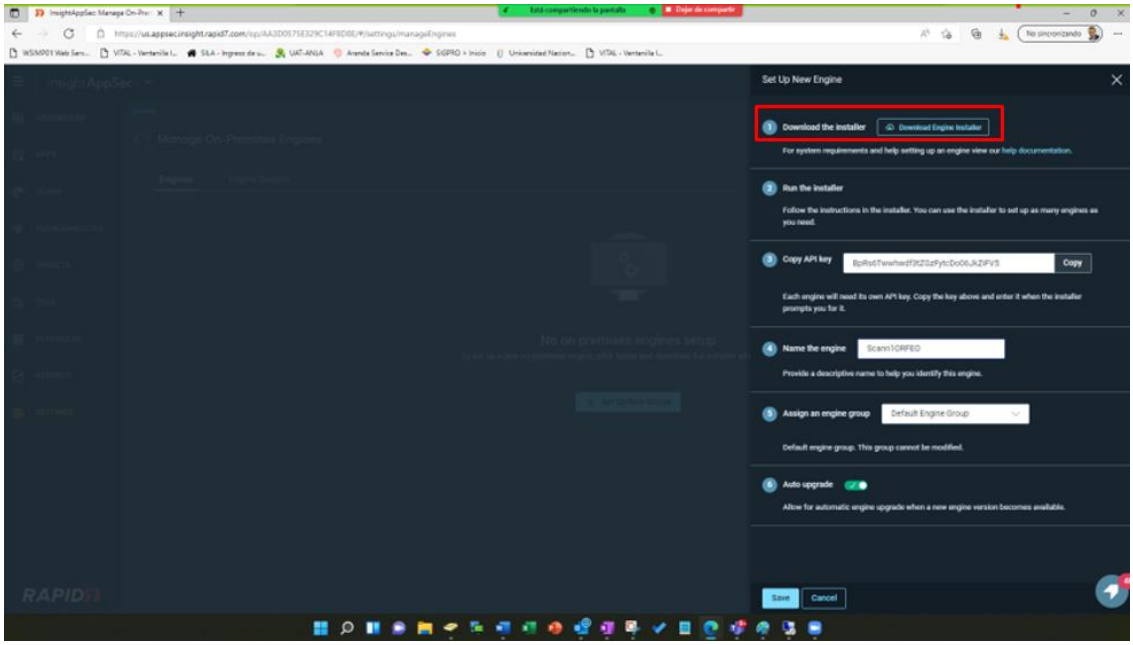
Ilustración 31. Activación de Cuenta



Fuente 31. Elaboración propia, captura de pantalla

Ahora como se refleja en la ilustración 32, se configura la descarga del nuevo motor de análisis de la herramienta, la cual se instala en el equipo de ejecutará el análisis.

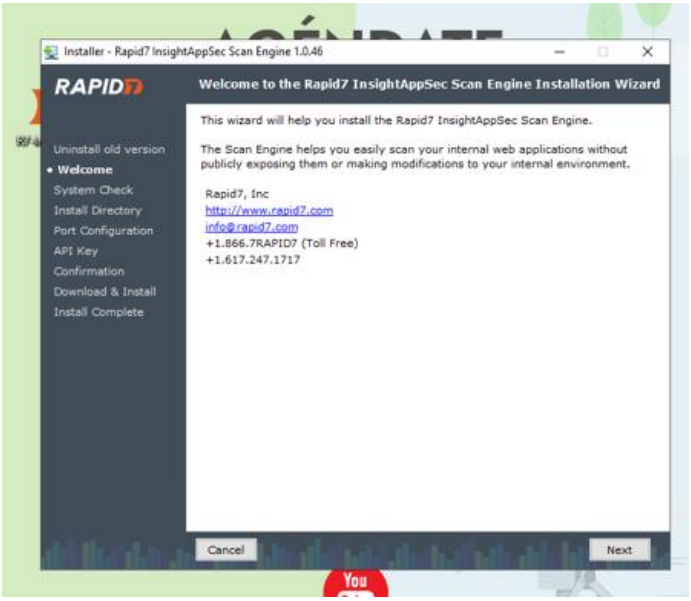
Ilustración 32. Descarga del Motor de Análisis



Fuente 32. Elaboración propia, captura de pantalla

Ya teniendo los accesos requeridos en la plataforma para la descarga del software, se procede a ejecutar el instalador en la máquina designada que ejecutara el escaneo de las vulnerabilidades, cabe aclarar que la versión del sistema operativo en donde se instala la herramienta es un Windows server 2019 se instala de forma normal con el asistente, según se evidencia en la ilustración 33.

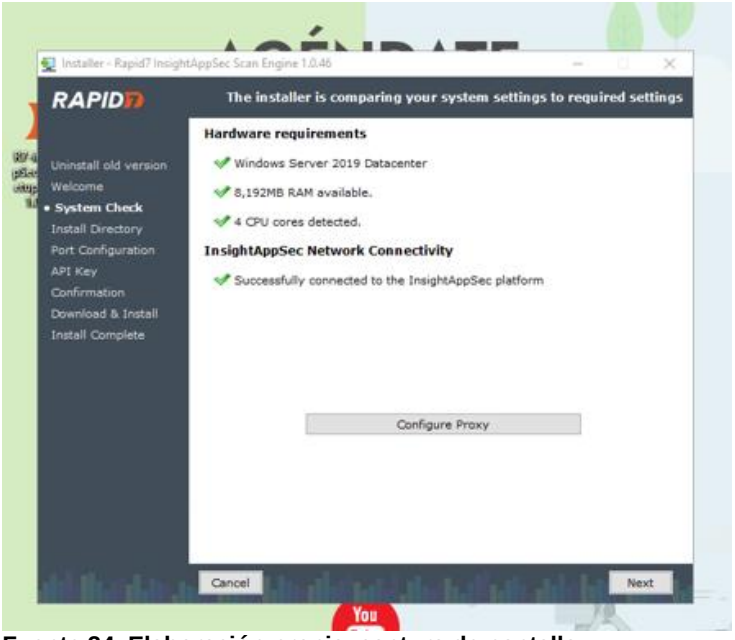
Ilustración 33. Instalación del Motor de Escaneo



Fuente 33. Elaboración propia, captura de pantalla

Este motor de escaneo necesita unas características de equipo particular para poder lograr el análisis, entre ellos un procesador de 4 Cores, RAM superior a 8GB, estas validaciones las realiza el aplicativo., como se ve en la ilustración 34.

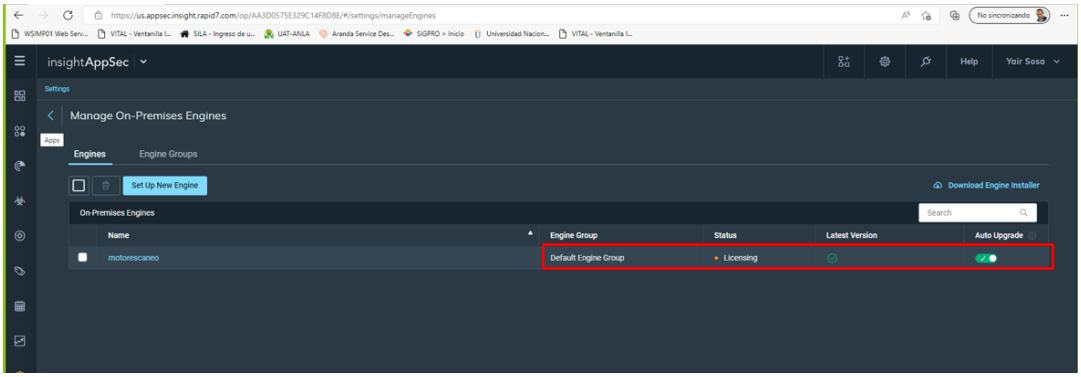
Ilustración 34. Validación del Servidor



Fuente 34. Elaboración propia, captura de pantalla

Para poder ejecutar las pruebas se ajusta en la plataforma web la validación de la licencia para la ejecución de las pruebas DAST del código, igualmente se configura la URL del proyecto orfeopruebas@aaaa para lanzar el análisis, como se ve en la ilustración 35.

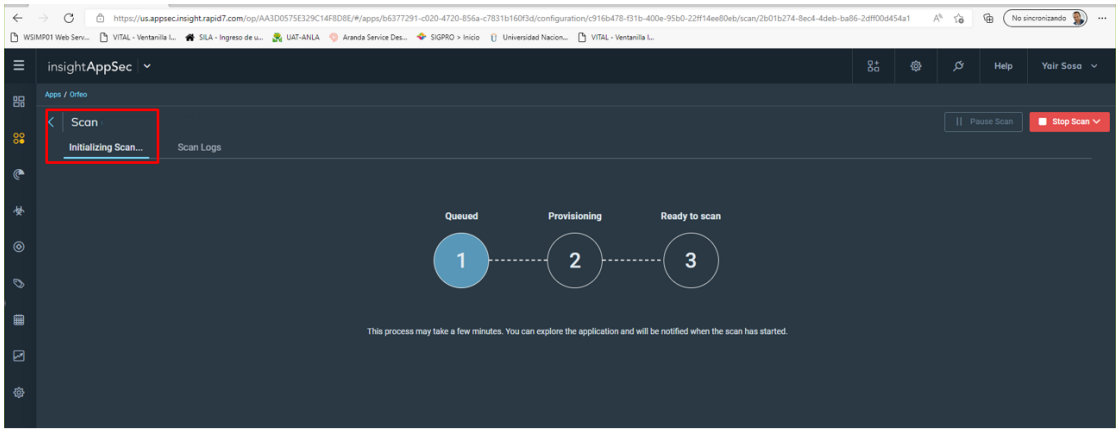
Ilustración 35. Validación de URL



Fuente 35. Elaboración propia, captura de pantalla

6.1.7 Ejecución De Análisis DAST Scan Engine Rapid 7. Como se evidencia en la ilustración 36, se lazan el motor de escáner sobre la URL del proyecto donde se tiene alojado el aplicativo ORFEO en la entidad, que para este ejercicio es orfeopruebas@aaaa Servicio instalado en uno de los servidores entregado por el grupo de infraestructura tecnológica.

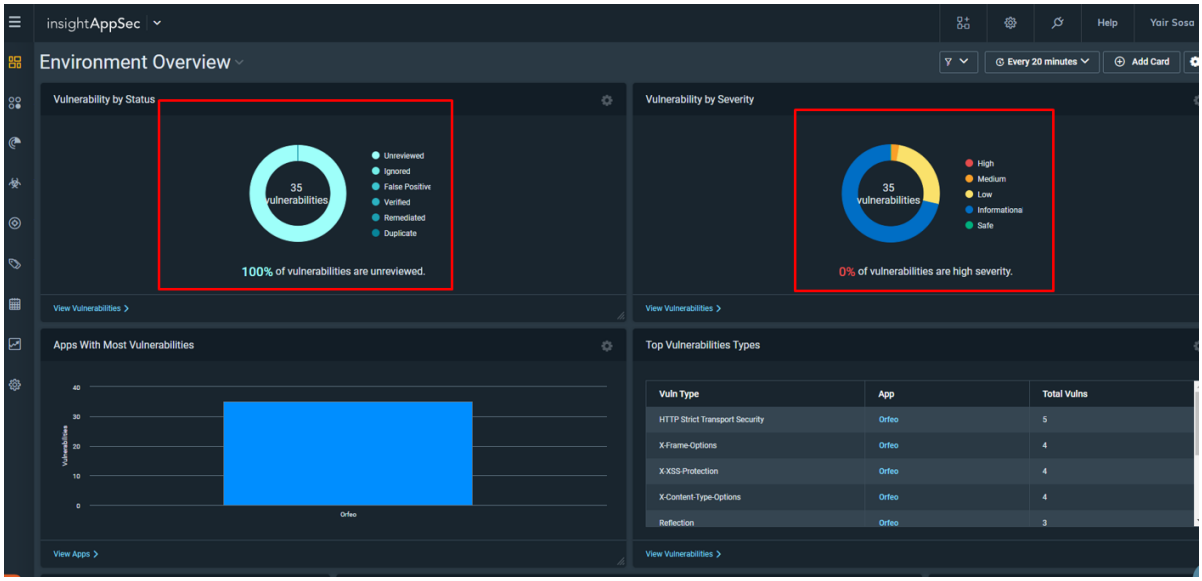
Ilustración 36. Lanzamiento de Escaneo DAST



Fuente 36. Elaboración propia, captura de pantalla

Luego del análisis se evidencia que el aplicativo con las pruebas automáticas dinámicas al aplicativo ORFEO refleja 35 vulnerabilidades, entre ellas fijación de sesión, datos expuestos, falsificación de sitio, etc, como se puede validar en la ilustración 37.

Ilustración 37. Análisis Vulnerabilidades DAST



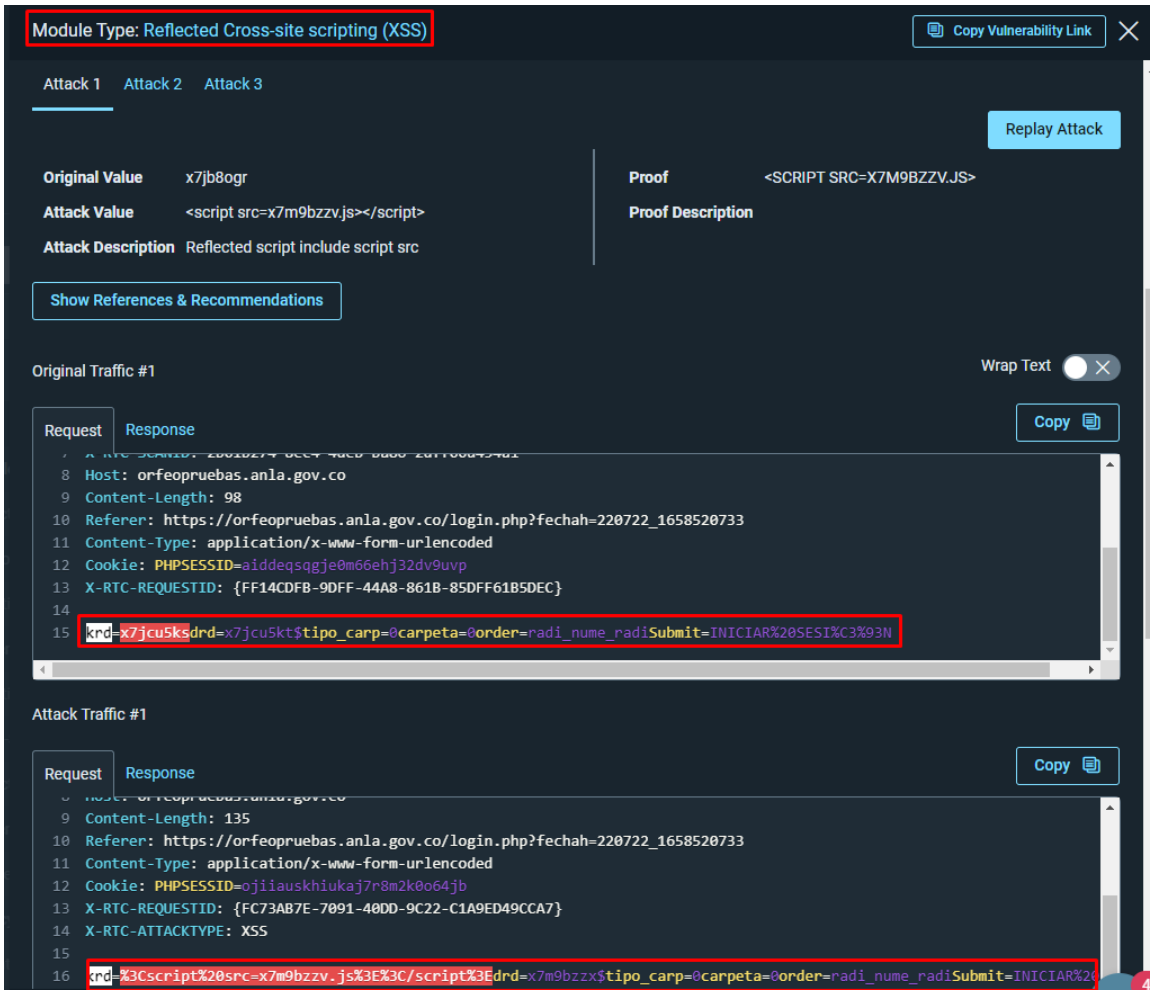
Fuente 37. Elaboración propia, captura de pantalla

6.1.8 Listado Vulnerabilidades Encontradas DAST. Esta herramienta realiza las validaciones del código dinámico, por reglas contenidas ya establecidas que ayudan a identificar determinadas funciones asociadas a vulnerabilidades de código, cabe aclarar que Nexpose Scan Engine de Rapid 7 integra funciones de análisis con los parámetros de OWASP, MITRE, COMC garantizando que sea un análisis funcional y actualizado, aunque se puede realizar una parametrización más personalidad de las reglas de análisis, para este efecto se dejara por defecto. Se aprecia en la ilustración 38, el listado de las vulnerabilidades del código del proyecto ORFEO en el aplicativo web de la herramienta.

Para tener una visión general acerca del análisis realizado por la herramienta Engine de Rapid 7, a continuación, se enunciarán las vulnerabilidades encontradas con su descripción general y la relación de estas en la afectación del código fuente, las cuales se tienen:

6.1.8.1 Cross-site-Scripting (XSS). Una vulnerabelidad no persistente donde se usa datos del usuario no validos e involucran consulta de tipo dinámico, sin ninguna codificación de parametros de conformidad al texto. Para este ejemplo la variable krd es un parametro de insercion de script malicioso y por tanto una vulnerablidad activa directo con el front. Como se ve en la ilustración 38.

Ilustración 38. Cross-Site Scripting (XSS)

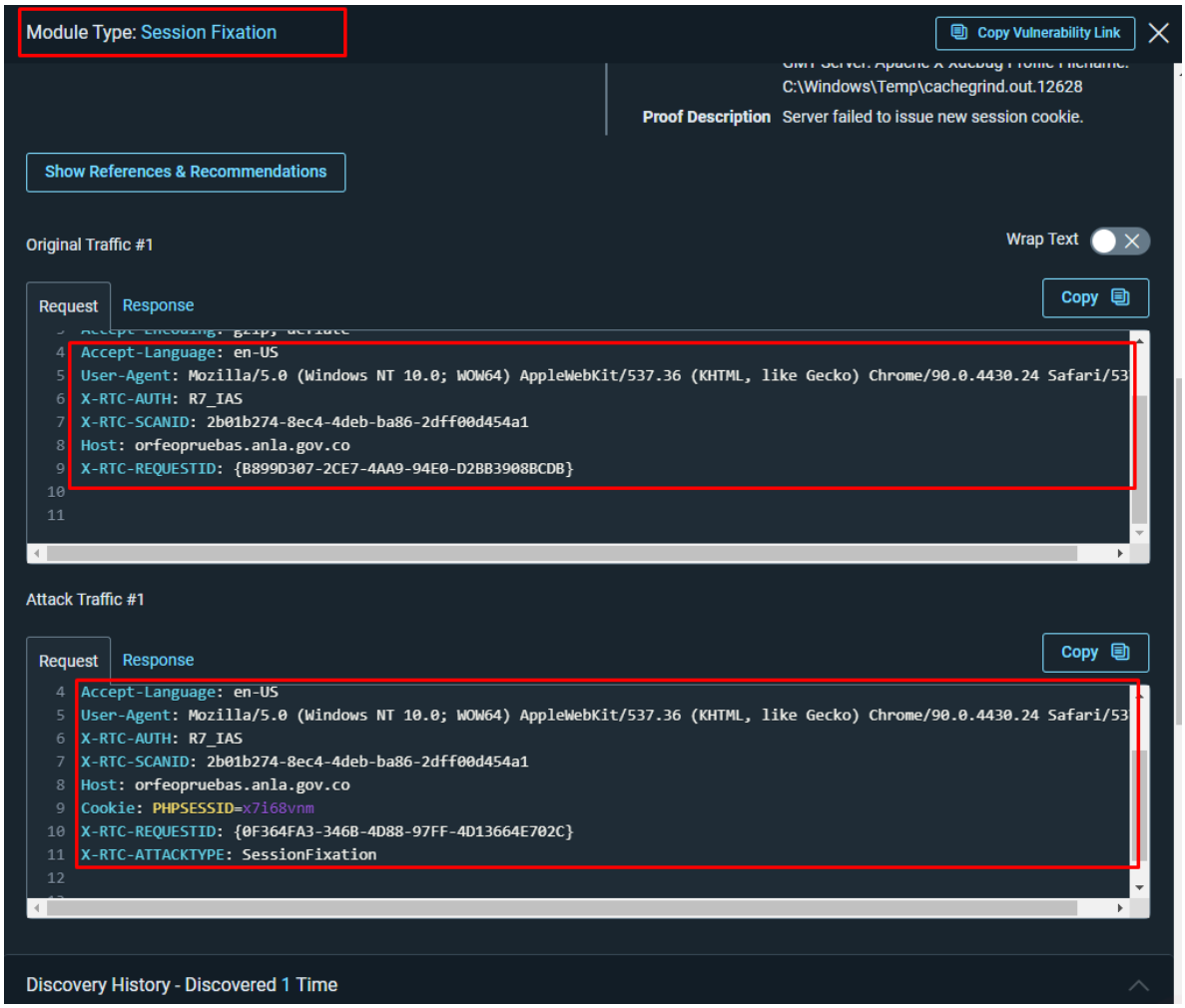


Fuente 38. Elaboración propia, captura de pantalla

6.1.8.2 Session Fixation. Vulnerabilidad que permite a un atacante retener una sesión de algún usuario que sea válida, es decir por medio de una cookie de ID de sesión se captura dicha sesión por la URL o envió de

método POST, se puede observar que en la cookie se ejecutó la PHPSESSID=x7i68vnm para capturar la sesión con el ID, como se refleja en la ilustración 39.

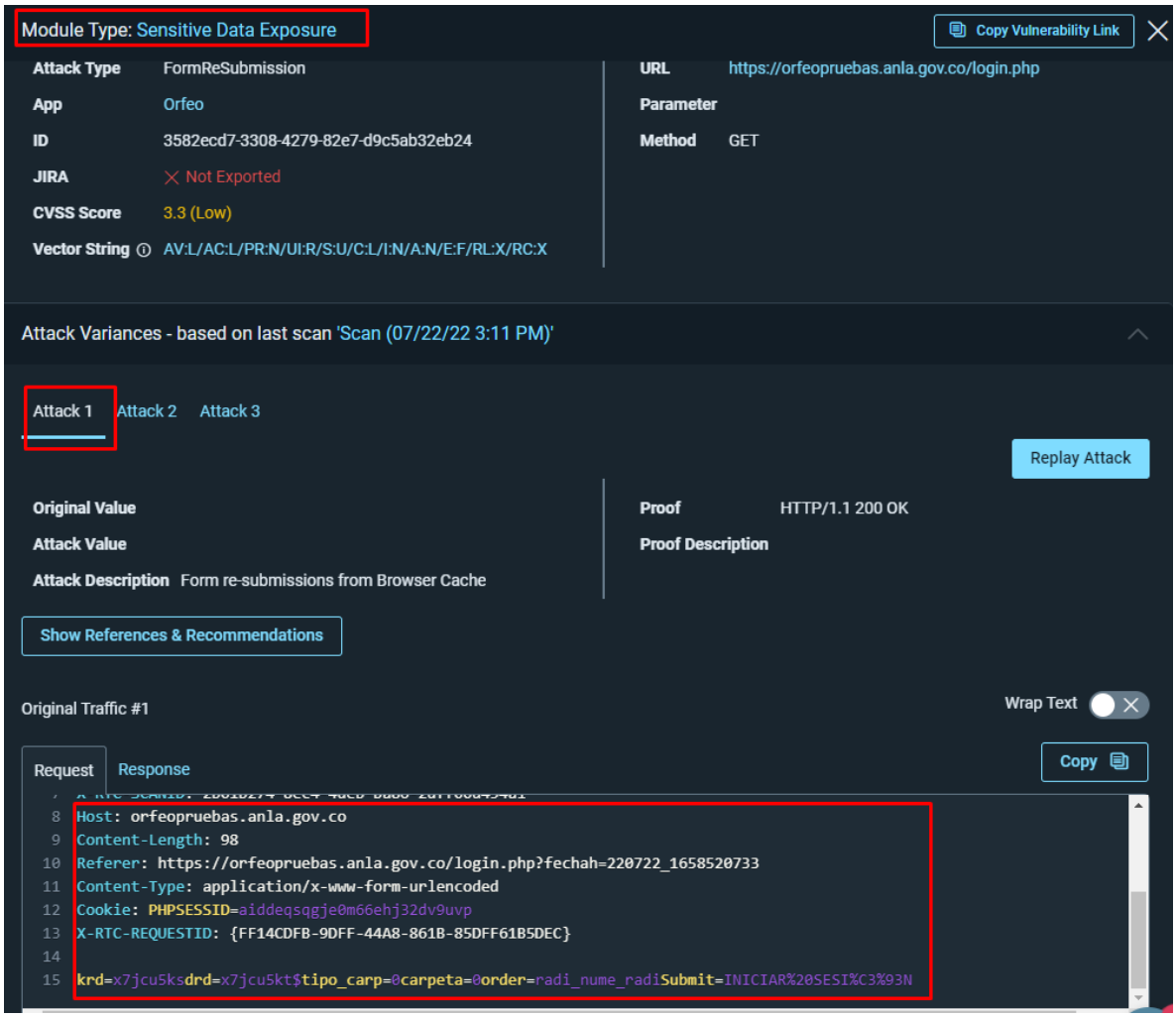
Ilustración 39. Session Fixation



Fuente 39.Elaboración propia, captura de pantalla

6.1.8.3 Sensitive Data Exposure. En esta vulnerabilidad se trasmiten datos confidenciales o sensitivos, de forma clara en texto, por tanto, puede ser visible y rastreable por algún atacante, esto por medio del envío de una solicitud con método POST o GET usando un formulario web, se evidencia falta de algún tipo de cifrado para los datos confidenciales. Como se aprecia en la ilustración 40.

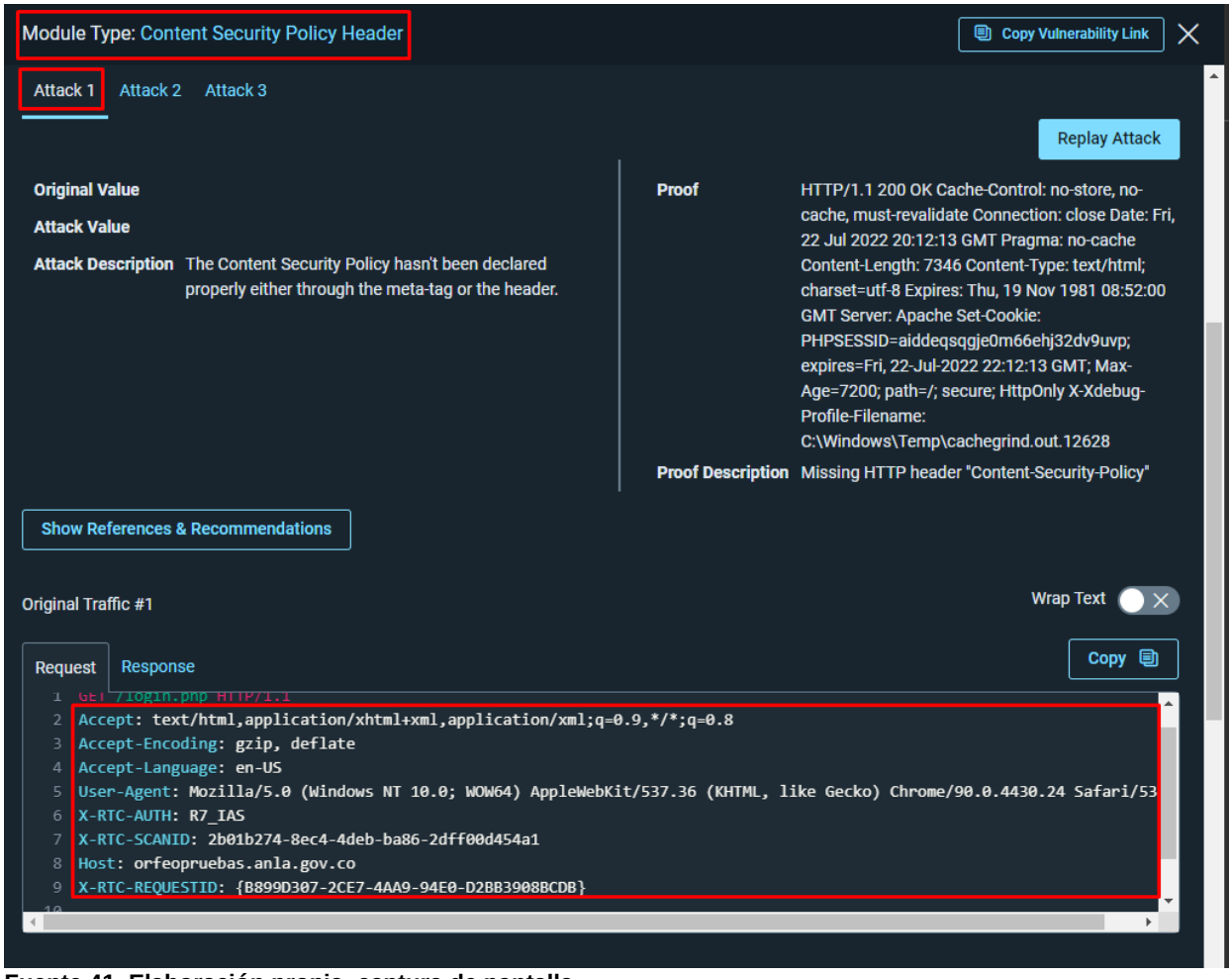
Ilustración 40. Data Exposure



Fuente 40. Elaboración propia, captura de pantalla

6.1.8.4 Content Security Policy Header. No se encuentra configurada o declarada de forma correcta, las políticas de seguridad en el desarrollo o configuración de la codificación del software, por tanto, puede convertirse en una debilidad de seguridad permitiendo explorar el contenido del navegador recibido del servidor web y generando confianza en ese contenido se puede ejecutar scripts maliciosos. Como se evidencia en la ilustración 41.

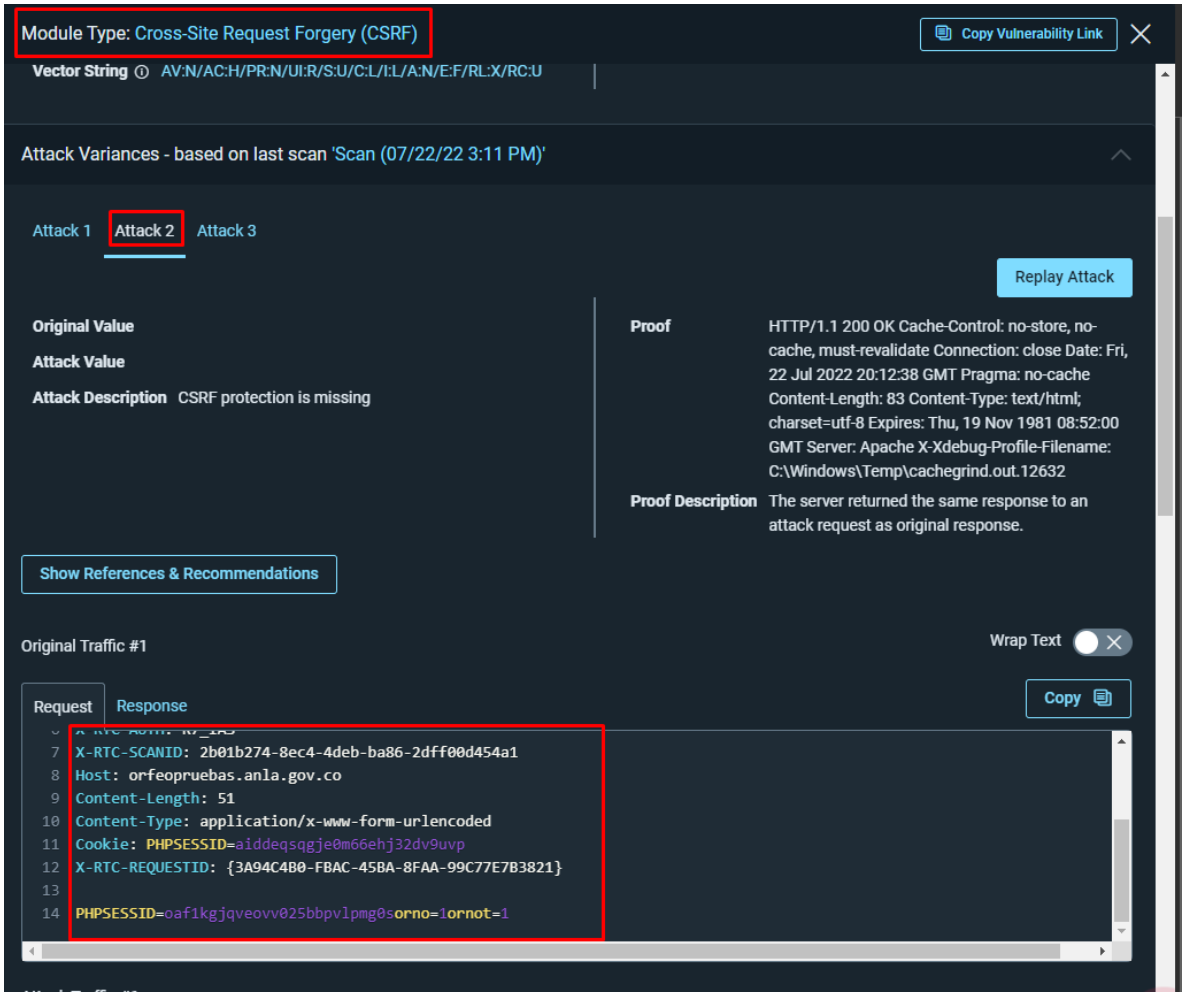
Ilustración 41. Policy Header



Fuente 41. Elaboración propia, captura de pantalla

6.1.8.5 Cross-Site Request Forgery (CSRF). Esta vulnerabilidad de falsificación de solicitudes entre sitios es cuando en el servidor web se define para recibir solicitudes de cliente, pero sin ningún tipo de mecanismo de verificación de envío intencional, dejando falta de verificación de la autenticidad de los datos enviados dejando vía libre para que un atacante ejecute una solicitud al servidor web y este la tome como autentica. Se aprecia que el servidor responde a un ataque como una solicitud original. Como se valida en la ilustración 42.

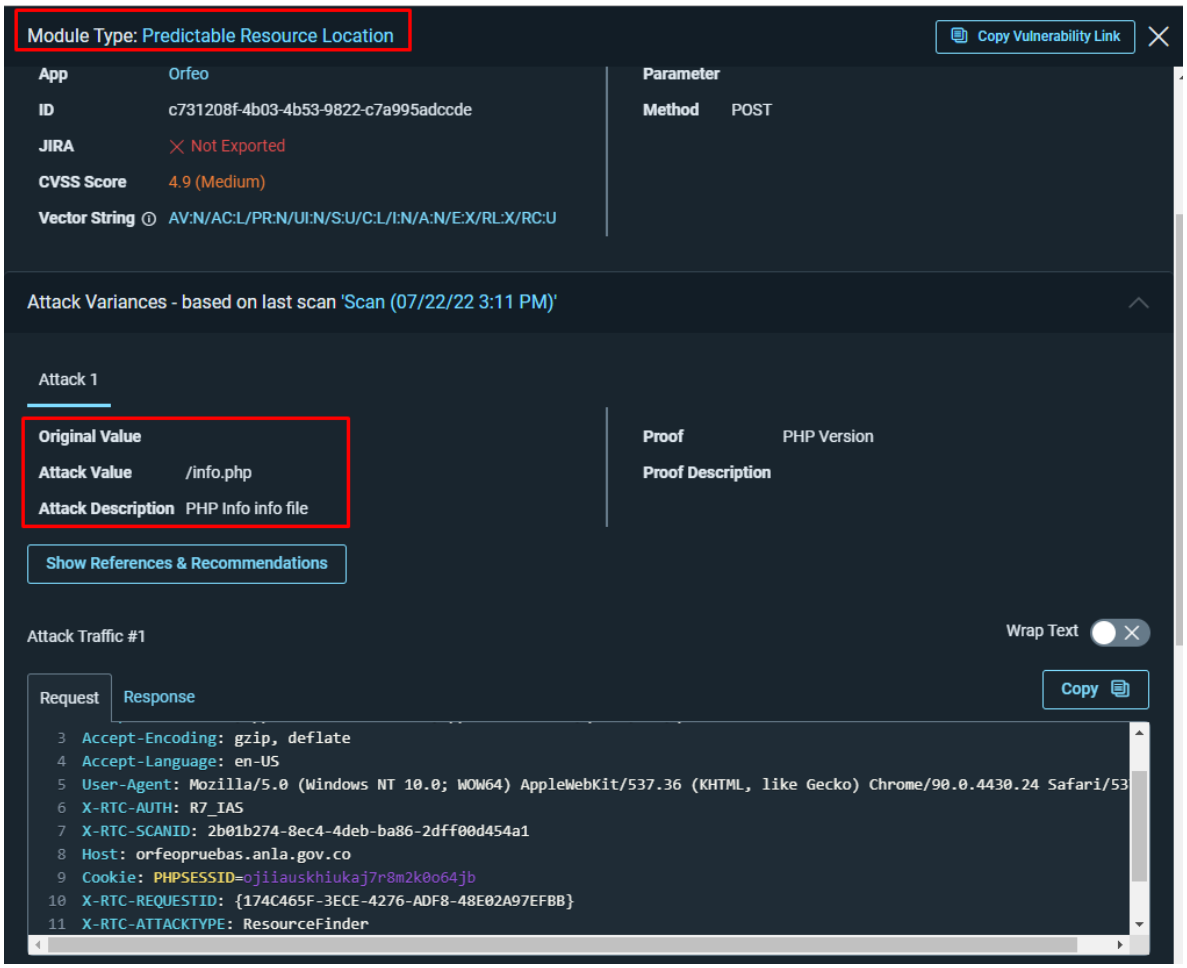
Ilustración 42. CSRF



Fuente 42. Elaboración propia, captura de pantalla

6.1.8.6 Predictable Resource Location. Esta vulnerabilidad permite ver las rutas de los diferentes directorios de configuración e información de las herramientas instaladas en el servidor, permitiendo revelar información de interfaces; para este caso en particular se observa la información del software PHP, su versión y complementos instalados en el mismo, esto permite un campo de acción para el atacante para recopilar información crucial. Como se valida en la ilustración 43.

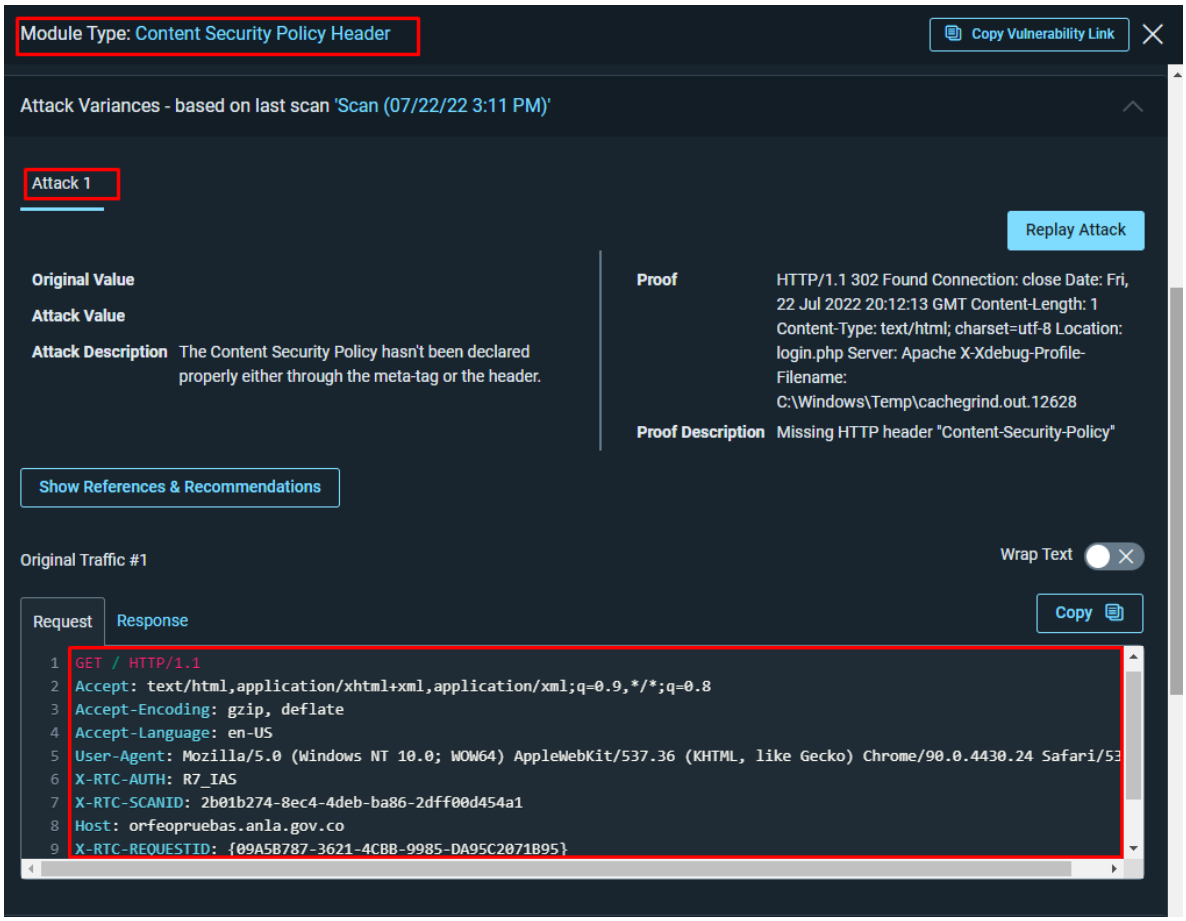
Ilustración 43. Resource Location



Fuente 43. Elaboración propia, captura de pantalla

6.1.8.7 Content Security Policy Header. Como se valida en la ilustración 44, no se cuenta con la configuración de seguridad adecuada en la verificación del control de acceso bien sea por métodos HTTP, GET o POST, el atacante puede aprovechar esta vulnerabilidad para usar funciones de privilegios para eliminar registros ejecutando scripts maliciosos por el navegador.

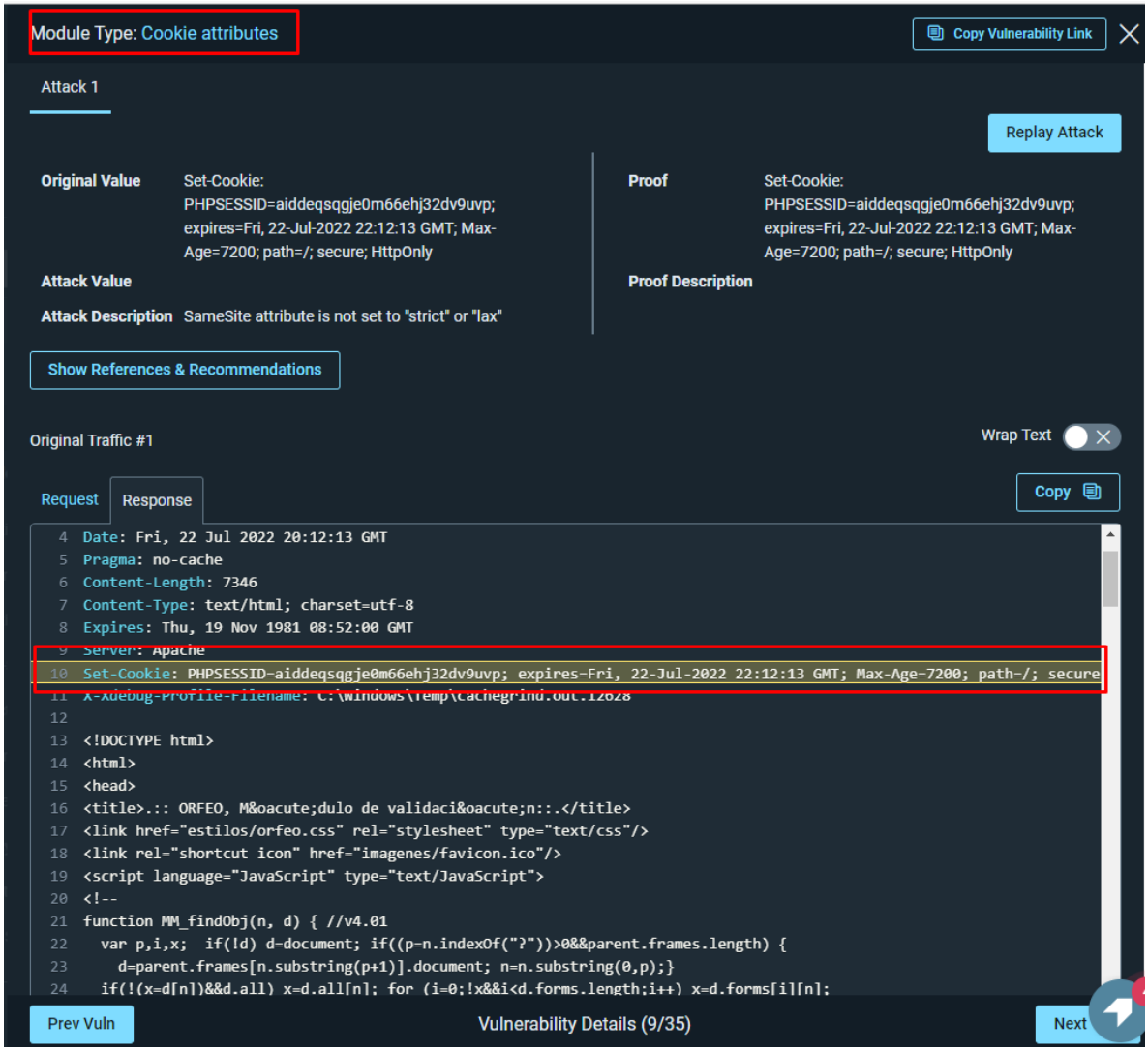
Ilustración 44. Policy Header



Fuente 44. Elaboración propia, captura de pantalla

6.1.8.8 Cookie Attributes. Esta vulnerabilidad del atributo de forma segura para las cookies confidenciales con las sesiones HTTPS no se tiene ajustado de la forma adecuada, por tanto, se pueden enviar esas cookies en texto sin un formato, para este efecto el set-cookie capturo la sesión. Como se observa en la ilustración 46.

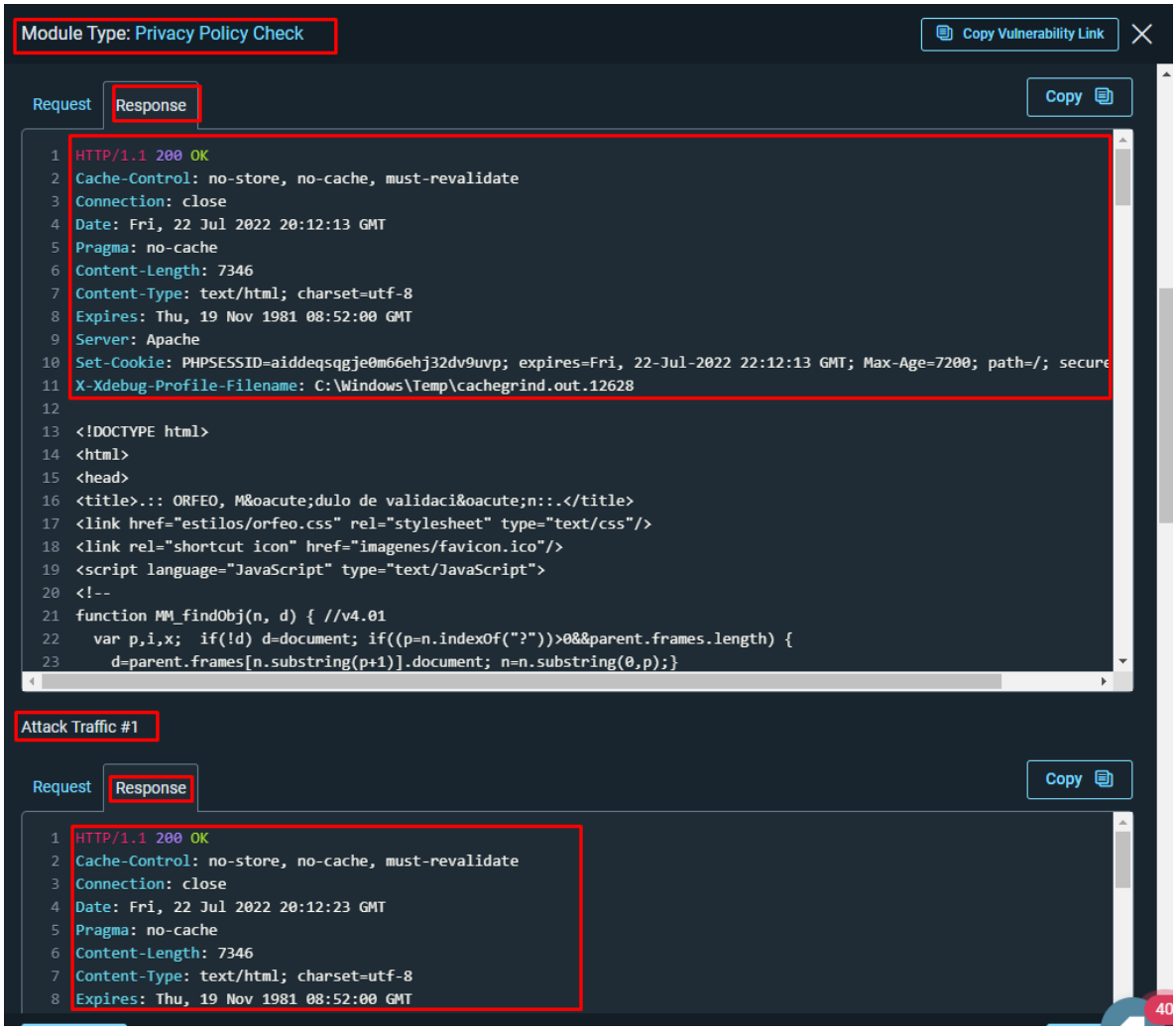
Ilustración 45. Cookies



Fuente 45. Elaboración propia, captura de pantalla

6.1.8.9 Privacy Policy Check. No se cuenta con el establecimiento de una política de privacidad del manejo de datos, este desarrollo opera identificadores de cookies para la sesiones y recopilación de data personal que pueden dejar el rastro y si se asocian con otros identificadores combinándose con el análisis del tráfico de información de servidores, se pueden crear perfiles de identificación de usuarios. Como registra en la ilustración 46.

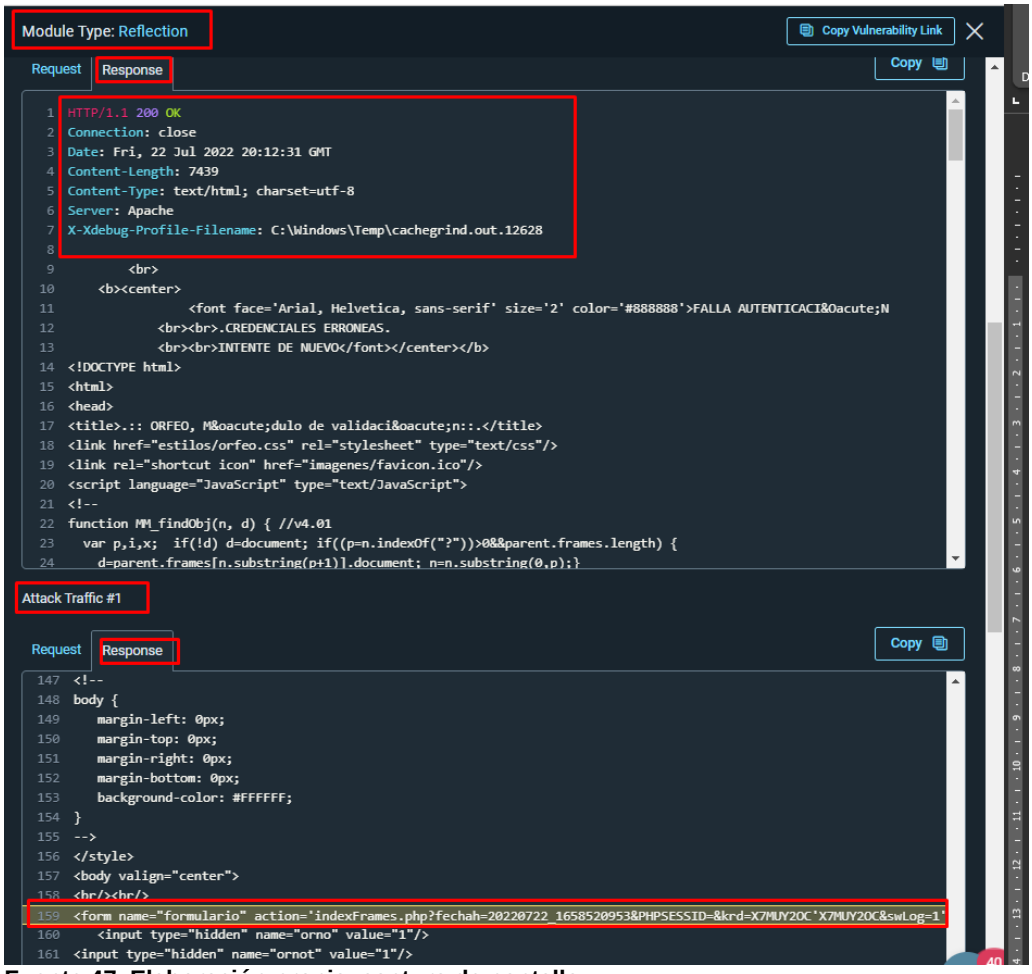
Ilustración 46. Policy Check



Fuente 46. Elaboración propia, captura de pantalla

6.1.8.10 Reflection. Esta vulnerabilidad consiste en modificar valores que el código web en el Front de ORFEO usa para pasar variables por medio de las cookies, un atacante podría capturar la identidad de un usuario por medio de inserción de scripts y redirigiéndolo a una web clonada, para sustraer los datos, es una vulnerabilidad asociada a Cross-site Scripting. Como se ve en la ilustración 48.

Ilustración 48. Reflection



Fuente 47. Elaboración propia, captura de pantalla

Se puede concluir que en las dinámicas de análisis de código se cuentan con herramientas funcionales que son versátiles y ayudan a los equipos involucrados del soporte, desarrollo y seguridad para plantear puntos de partida en la planeación de tareas y ajustes necesarios para desplegar software menos vulnerable, eficiente y de buena calidad.

6.2 DESARROLLO DEL OBJETIVO 2.

Elaborar la documentación sobre el análisis y pruebas realizadas al sistema de información ORFEO, para que sean aplicadas conforme a las políticas establecidas por la por medio de la formalización del cumplimiento del proyecto.

En el presente capitulo se formaliza la documentación técnica de los eventos encontrados por las pruebas realizadas de la siguiente forma:

6.2.1 Software Aplicado Para Las Pruebas.

Static Application Security Testing (SAST): Realizadas con el Software Libre SonarQube versión 9.4, instalado local para la ejecución del análisis del código fuente.

S.O: Windows 11
Procesador: Xeon 2.0 Ghz
Memoria: 64 RAM
HD:1 TB

Dynamic Application Security Testing (DAST): Realizadas con el aplicativo Nexpose Scan Engine 1.0.48 de Rapid 7, instalado el agente en un servidor de la red corporativa suministrado por el personal de Infraestructura Tecnológica para la ejecución del análisis del código fuente.

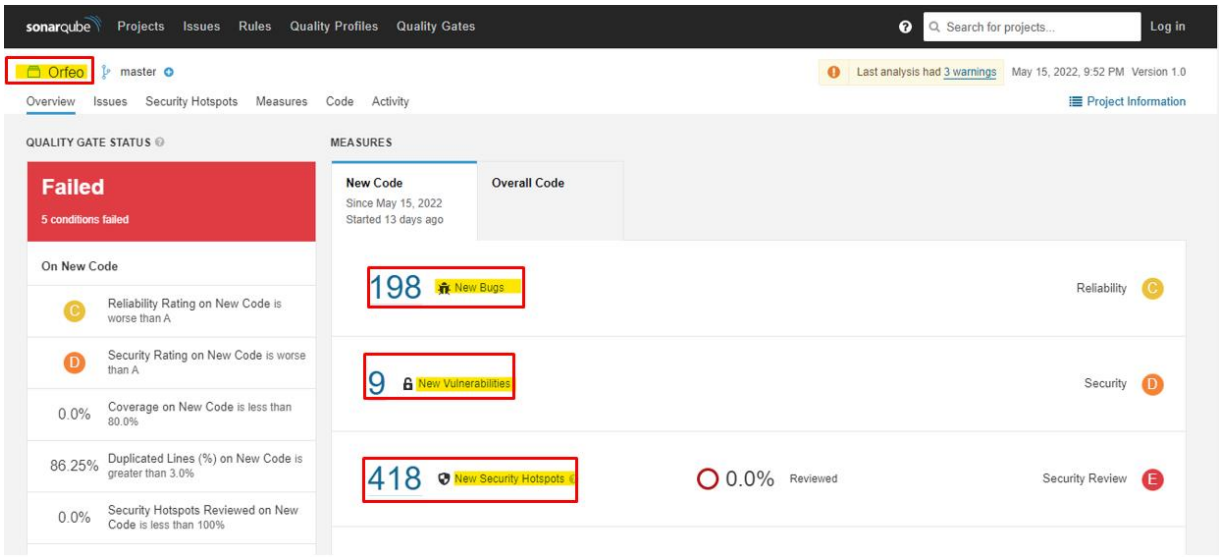
S.O: Windows Server 2019
Procesador: Xeon 2.4 Ghz
Memoria: 12 RAM
HD: 200 GB

6.2.2 Resultados

Static Application Security Testing (SAST)

Con el uso del Software Libre SonarQube versión 9.4, se evidencia 198 Bugs, 9 vulnerabilidades y 418 puntos de acceso de seguridad, 164 deudas técnicas o falta de mantenimiento del código, errores en las líneas de código y presenta un 86.25 % de duplicidad en el código. Como se evidencia en la ilustración 48.

Ilustración 48. Resultados de Análisis Finales

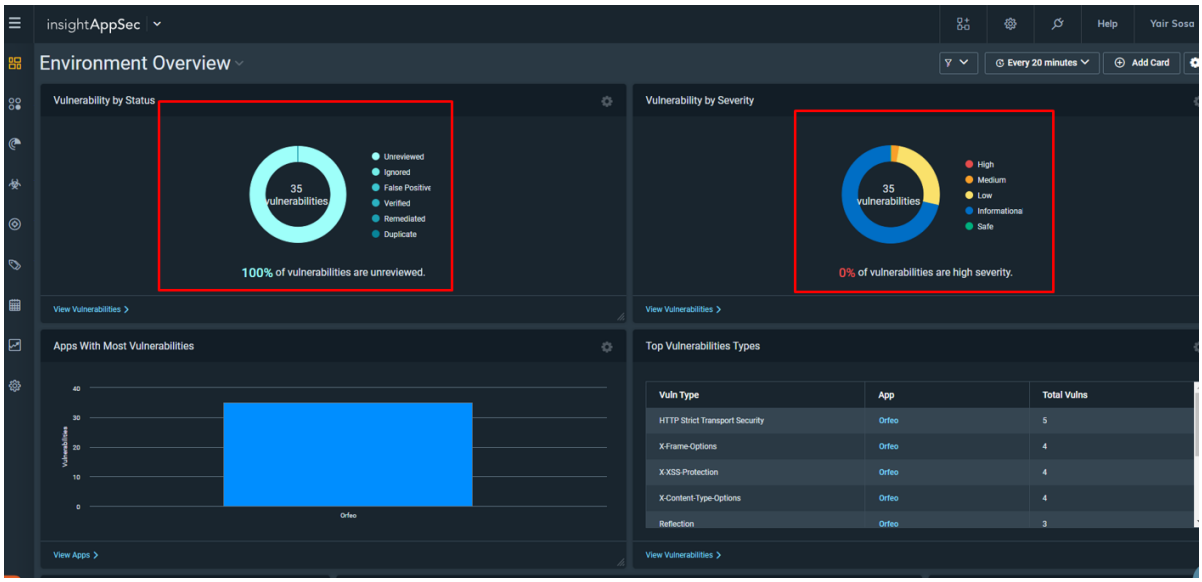


Fuente 48. Elaboración propia, captura de pantalla

Dynamic Application Security Testing (DAST)

Con el uso del software propietario, con licencia perteneciente a la entidad Nexpose Scan Engine 1.0.48 se evidencia que el aplicativo con las pruebas automáticas dinámicas al aplicativo ORFEO refleja 35 vulnerabilidades, entre ellas fijación de sesión, datos expuestos, falsificación de sitio, etc. Como se observa en la ilustración 49.

Ilustración49. Análisis Vulnerabilidades DAST Finales



Fuente 49. Elaboración propia, captura de pantalla

A continuación, en la siguiente tabla se anuncian las recomendaciones según la asociación de cada vulnerabilidad, es de aclarar que se presentan estas recomendaciones al personal de administración técnico y funcional del aplicativo, igualmente al personal de seguridad para que de forma grupal se trabaje en la mitigación de las vulnerabilidades encontradas.

Tabla 2. Vulnerabilidades, Ataques Y Mitigaciones

Vulnerabilidad	Tipo	Vector
CWE-1006	Malas prácticas de codificación	*Add curly braces around the nested statement* **Ataque:** Prácticas de codificación de consideración inseguras y con posibilidades que sea explotada una vulnerabilidad, indica que el producto no se ha desarrollado cuidadosamente en aspectos seguros.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-1006	Malas prácticas de codificación	*Add curly braces around the nested statement* Deja espacio para que un atacante realice inserción de código aprovechando que las llaves {} no se cierran, esta inserción podría llevar al usuario a una página maliciosa o URL no valida y capturar datas sensibles. Genera errores al momento de ejecución del código. **Mitigación:** Aplicar las políticas de buenas prácticas, no dejando líneas de código abiertas, cerrando las sentencias de ejecución. Adoptar conjuntos de estándares de codificación para el equipo de trabajo.
CVE-2015-4024	Vulnerabilidad complejidad algorítmica (PHP)	*Refactor this function to reduce its Cognitive Complexity* **Ataque:** Permite a los atacantes de forma remota provocar denegación de servicio por consumo de recursos del equipo, bien sea procesador o memoria usando formularios para el crecimiento inadecuado del flujo lineal del código e incrementar las estructuras que rompen el flujo.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CVE-2015-4024	Vulnerabilidad complejidad algorítmica (PHP)	*Refactor this function to reduce its Cognitive Complexity* **Mitigación:** Actualización de cambio de PHP 5. Gestionar las mejores prácticas de escritura, desarrollo y manteamiento del código reduciendo la complejidad por los métodos a nivel de clases y aplicaciones.
CWE-295	Validación de Certificado Incorrecta	*Enable server certificate validation on this SSL/TLS* **Ataque:** Permite que cualquier atacante suplante el servidor web o la entidad de confianza para conectar a un host malicioso. **Mitigación:** Implementar certificados válidos para la encriptación de datos.
CWE:327	Algoritmo Criptográfico en Riesgo	*Enable server certificate validation on this SSL/TLS* **Ataque:** El atacante puede descifrar el algoritmo y comprometer datos confidenciales y data sensible. **Mitigación:** Utilizar algoritmos o mecanismos de cifrado simétrico o asimétricos con hashes robustos SHA2, SHA3, SHA-256.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
OWASP A02:2021	Fallas Criptográficas	*Enable server certificate validation on this SSL/TLS* **Ataque:** Se puede ejecutar el monitoreo de trafico de red, degradando las conexiones HTTPS interceptando solicitudes y secuestrando la sesión. **Mitigación:** Cifrar la data con protocolos seguros como TLS.
CWE-457	Uso de Variable no inicializada	BUGS **Ataque:** Se puede controlar o leer los contenidos de la memoria de pila generando implicaciones de seguridad pre inicializando la variable para inyectar código. **Mitigación:** Asignación de valores a las variables, gestionar bibliotecas de cadenas seguras y abstracción de contenedores.
CWE-561	Código Muerto	BUGS **Ataque:** Invocación del código inactivo puede generar bucles infinitos, generar cadenas constantes en los nombres de un campo relevantes. **Mitigación:** Eliminar, depurar el código inactivo y ejecutar escaneos con herramientas especializadas.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-628	Invocación de funciones con argumentos incorrectos	BUGS **Ataque:** Invocar funciones o rutinas no específicas, generando comportamiento incorrecto suministrando al atacante recursos del sistema. **Mitigación:** Ejecutar el compilador para validar la falla, observando el código que no se despliegue en el control.
CVE-2021-42536	Exposición de Recursos e Información Confidencial	*Password detected in this variable* **Ataque:** Información vulnerable a la divulgación del nombre de usuario y la contraseña del par al permitir que todos los usuarios accedan para leer variables globales, pueden escalar privilegios. **Mitigación:** Cambiar la variable en el código, sin registro de visibilidad o texto claro. Usar cifrado de datos para encriptar las contraseñas.
CWE-1041	Uso Código Redundante	*Duplicated Lines* **Ataque:** Invocar funciones o rutinas no específicas, generando comportamiento incorrecto en el sistema de información.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-1041	Uso Código Redundante	*Duplicated Lines* Elevando el consumo de recursos del servidor, generando tiempos de espera en respuesta debido al procesamiento de las líneas de código redundantes, haciendo la mantenibilidad del código compleja, con tendencia a errores y no ejecución de cambios realizados. **Mitigación**: Mantener las buenas prácticas en medidas de calidad de desarrollo en el código. Depurar las líneas de código duplicadas que no ejecuten procedimientos o manejo de variables de datos. Implementar funcionalidades frecuentes en una sola, para hacer un llamado desde la base del código de ser necesario y requerido.
CWE-259 CWE-321 CWE-798	*Hardcores*	*HardCoded* (Código Quemado) **Ataque:** el atacante puede tener acceso a los valores estipulados o preestablecidos y explotar fallas de seguridad cambiando las credenciales o el *HardCoded*.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-259 CWE-321 CWE-798	*Hardcores*	*HardCoded* (Código Quemado) **Mitigación:** Usar clases constantes para el alanceando de los valores. Implementar métodos de inicio de sesión para que el usuario ingrese una contraseña segura.
OWASP A07:2021 OWASP A03:2021	Identificación y fallos de Autenticación Inyección	*Cross-site-Scripting (XSS)* **Ataque:** La variable krd opera un parámetro de scripts inserción malicioso vulnerabilidad activa directo con el Front del aplicativo. Dejando que el atacante ejecute HTML y JavaScript en el navegador de la víctima, cuando se engaña a un usuario para que haga clic en el enlace malicioso, el código inyectado se dirige al sitio web vulnerable. **Mitigación:** Cumplir con el filtrado estricto de las codificaciones de caracteres HTML. Filtrar toda la información enviada al servidor a través de formularios POST/GET y parámetros de consulta de URL con un énfasis particular en el filtrado de caracteres específicos de HTML.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-384 OWASP A07:2021 OWASP A02:2021	Fijación de Sesión Identificación y fallos de Autenticación Fallos Criptográficos	*Session Fixation* **Ataque:** Un atacante secuestra una sesión de usuario válida del valor de la cookie de ID de sesión para que pueda continuar usando el mismo valor de cookie. El atacante puede determinar el nombre y el valor de la ID de la sesión, donde el servidor acepta cualquier peticion del atacante, este visita el sitio y rastrea en la respuesta llevando a la victima a una web maliciosa. **Mitigación:** Metodo para cambiar la ID de sesión cuando los usuarios inician sesión. No aceptar ID de sesión en los parámetros GET o POST. Usar cookies HTTP como para codificar la información de la sesión. Regenerar el SID por en los sistemas que lo admiten, usando identificadores de sesión SSL/TLS.
CWE-319 OWASP A02:2021 OWASP A03:2021	Transmisión de Informacion Sensible por Texto Claro	*Sensitive Data Exposure* **Ataque:** El atacante puede indicir a un usuario web que intente actualizar la respuesta del servidor con ciertos agentes de usuario.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-319	Transmisión de Informacion Sensible por Texto Claro	*Sensitive Data Exposure* Puede hacer que se vuelva a enviar el contenido de la solicitud HTTP POST original.
OWASP A02:2021	Fallos Criptográficos	**Mitigación:** Aplicar un patrón de diseño de desarrollo web que evita algunos envíos de formularios duplicados como es Post/Redirect/Get (PRG) el cual permite crear una interfaz más intuitiva para los agentes de usuario. PRG implementa marcadores de actualización de una manera predecible y no crea envíos de formularios duplicados.
OWASP A03:2021	Inyección	
CWE-16	Configuración	*Content Security Policy Header* **Ataque:** Explotar la confianza del navegador en el contenido recibido del servidor. Scripts maliciosos son ejecutados por el navegador de la víctima por la confianza creada en el navegador en la fuente del contenido de servidor. **Mitigación:** Aplicar políticas de contenido Seguro (CSP) ya que un navegador compatible con CSP solo ejecutará secuencias de comandos cargadas en archivos de origen recibidos de dominios incluidos en listas de permitidos.
OWASP A05:2021	Configuración Incorrecta de Seguridad	

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-16	Configuración	*Content Security Policy Header* **Mitigación**: Aplicar las CSP hacen posible se reduzcan los vectores por los cuales XSS puede ocurrir al especificar los dominios que el navegador debe considerar como fuentes válidas de scripts ejecutables.
OWASP A05:2021	Configuración Incorrecta de Seguridad	
CWE-352	Cross-site Request Forgery (CSRF)	*Cross-site Request Forgery (CSRF)* **Ataque:** Un ataque engaña a la víctima para que cargue una página que contiene una solicitud maliciosa, esta hereda la identidad y los privilegios de la víctima para realizar una función no deseada en nombre de la víctima, como cambiar la dirección de correo electrónico, la dirección particular o la contraseña de la víctima. **Mitigación:** Requerir que el cliente proporcione datos de autenticación en la misma solicitud HTTP utilizada para realizar cualquier operación con implicaciones de seguridad. Limitar la vida útil de las cookies de sesión.
OWASP A01:2021	Control de Acceso Roto	

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-352	Cross-site Request Forgery (CSRF)	*Cross-site Request Forgery (CSRF)* **Mitigación:** Utilizar un filtro CSRF como el CSRFGuard de OWASP. El filtro intercepta las respuestas, detecta si se trata de un documento html e inserta un token en los formularios y, opcionalmente, inserta un script para insertar tokens en las funciones ajax.
OWASP A01:2021	Control de Acceso Roto	
CWE-530	Exposición de Ruta de Archivo	*Predictable Resource Location* **Ataque:** Según el contenido del archivo, puede revelar interfaces de administrador o direcciones URL alternativas que se supone que están ocultas para los usuarios y el atacante puede capturar informacion para explotar un ataque, ejemplo: conociendo la version del PHP. **Mitigación:** Restrinjir el acceso a directorios confidenciales (por ejemplo, admin) por contraseña y dirección IP o ubicación de red. Modificar la política de implementación para incluir la eliminación de directorios confidenciales.
OWASP A01:2021	Control de Acceso Roto	
OWASP A05:2021	Configuración Incorrecta de Seguridad	

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
CWE-614 OWASP A05:2021	Cookie Confidencial en la sesión HTTPS Configuración Incorrecta de Seguridad	*Cookie Attributes* **Ataque:** Un atacante secuestra una sesión de usuario válida. el estado de autenticación del valor de la cookie de ID de sesión para que pueda continuar usando el mismo valor de cookie. **Mitigación:** Parametizar el atributo SameSite, con un valor de "estricto" debe usarse para cualquier cookie de sesión o aquella que contenga información confidencial y que pueda enviarse en una solicitud.
CWE-1211 OWASP A07:2021	Errores de Autenticación Identificación y fallos de Autenticación	*Privacy Policy Check* **Ataque:** los identificadores ee linea pueden dejar rastros que, en particular cuando se combinan con identificadores únicos y otra información recibida por los servidores, por tanto, el atacante puede usar esos rastros para crear perfiles de las personas físicas e identificarlas. **Mitigación:** Establecer política de privacidad para obtener el consentimiento de los usuarios para almacenar cualquier información personal en un equipo que gestione datos.

Continuación Tabla 2.

Vulnerabilidad	Tipo	Vector
OWASP A07:2021	Identificación y fallos de Autenticación	*Reflection* **Ataque:** La variable krd opera un parámetro de scripts inserción malicioso vulnerabilidad activa directo con el Front del aplicativo, dejando que el atacante ejecute HTML y JavaScript en el navegador de la víctima, cuando se engaña a un usuario para que haga clic en el enlace malicioso, el código inyectado se dirige al sitio web vulnerable. **Mitigación:** Cumplir con el filtrado estricto de las codificaciones de caracteres HTML. Filtrar toda la información enviada al servidor a través de formularios POST/GET y parámetros de consulta de URL con un énfasis particular en el filtrado de caracteres específicos de HTML
OWASP A03:2021	Inyección	

Se concluye que la herramienta ORFEO cuenta con varias vulnerabilidades significativas que deben ser atendidas con prontitud, si la herramienta se despliega en producción, ya que estará expuesta y sensible a varios ataques que de ser materializados o explotados generarán un gran impacto en los procesos operativos de la entidad, ya que este sistema de información es transversal a toda la gestión de esta.

6.3 DESARROLLO DE OBJETIVO 3.

Proponer recomendaciones para mitigar el riesgo de las vulnerabilidades encontradas dentro del sistema de gestión documental ORFEO para la Autoridad aplicando la evaluación de los ítems identificados.

6.3.1 Recomendaciones Mitigación De Vulnerabilidades Encontradas.

En la siguiente tabla se presentan las diferentes recomendaciones que se pueden aplicar para la mitigación de las vulnerabilidades encontradas por las pruebas realizadas SASY y DAST al código fuente del proyecto ORFEO para la Autoridad Nacional de Licencias Ambientales.

Tabla 3. Vector y Recomendación

Vector	Recomendación
Add curly braces around the nested statement	Aplicar las políticas de buenas prácticas, no dejando líneas de código abiertas, cerrando las sentencias de ejecución. Adoptar conjuntos de estándares de codificación para el equipo de trabajo.
Refactor this function to reduce its Cognitive Complexity	Actualización de cambio de PHP 5. Gestionar las mejores prácticas de escritura, desarrollo y mantenimiento del código reduciendo la complejidad por los métodos a nivel de clases.
Enable server certificate validation on this SSL/TLS	Implementar certificados válidos para la encriptación de datos.

Continuación Tabla 3.

Vector	Recomendación
Enable server certificate validation on this SSL/TLS	Utilizar algoritmos o mecanismos de cifrado simétrico o asimétricos con hashes robustos SHA2, SHA3, SHA-256. Cifrar la data con protocolos seguros como TLS.
BUGS	Asignación de valores a las variables, gestionar bibliotecas de cadenas seguras y abstracción de contenedores. Eliminar, depurar el código inactivo y ejecutar escaneos con herramientas especializadas. Ejecutar el compilador para validar la falla, observando el código que no se despliegue en el control.
Password detected in this variable	Cambiar la variable en el código, sin registro de visibilidad o texto claro. Usar cifrado de datos para encriptar las contraseñas
Duplicated Lines	Mantener las buenas prácticas en medidas de calidad de desarrollo en el código. Depurar las líneas de código duplicadas que no ejecuten procedimientos o manejo de variables de datos.

Continuación Tabla 3.

Vector	Recomendación
Duplicated Lines	Implementar funcionalidades frecuentes en una sola, para hacer un llamado desde la base del código de ser necesario y requerido.
HardCoded (Código Quemado)	Usar clases constantes para el alanceando de los valores. Implementar métodos de inicio de sesión para que el usuario ingrese una contraseña segura.
Cross-site-Scripting (XSS)	Cumplir con el filtrado estricto de las codificaciones de caracteres HTML. Filtrar toda la información enviada al servidor a través de formularios POST/GET y parámetros de consulta de URL con un énfasis particular en el filtrado de caracteres específicos de HTML.
Session Fixation	Metodo para cambiar la ID de sesión cuando los usuarios inician sesión. No aceptar ID de sesión en los parámetros GET o POST. Usar cookies HTTP como para codificar la información de la sesión. Regenerar el SID por en los sistemas que lo admiten, usando identificadores de sesión SSL/TLS.

Continuación Tabla 3.

Vector	Recomendación
Sensitive Data Exposure	Aplicar un patrón de diseño de desarrollo web que evita algunos envíos de formularios duplicados como es Post/Redirect/Get (PRG) el cual permite crear una interfaz más intuitiva para los agentes de usuario. PRG implementa marcadores y el botón de actualización de una manera predecible que no crea envíos de formularios duplicados.
Content Security Policy Header	Aplicar politicas de contenido Seguro (CSP) ya que un navegador compatible con CSP solo ejecutará secuencias de comandos cargadas en archivos de origen recibidos de esos dominios incluidos en la lista de permitidos. Aplicar las CSP hacen posible se reduzcan los vectores por los cuales XSS puede ocurrir al especificar los dominios que el navegador debe considerar como fuentes válidas de scripts ejecutables.
Cross-site Request Forgery (CSRF)	Utilizar un filtro CSRF como el CSRFGuard de OWASP. El filtro intercepta las respuestas, detecta si se trata de un documento html e inserta un token en los formularios y, opcionalmente, inserta un script para insertar tokens en las funciones ajax.

Continuación Tabla 3.

Vector	Recomendación
Cross-site Request Forgery (CSRF)	Requerir que el cliente proporcione datos de autenticación en la misma solicitud HTTP utilizada para realizar cualquier operación con implicaciones de seguridad.
Predictable Resource Location	Restrinjir el acceso a directorios confidenciales (por ejemplo, admin) por contraseña y dirección IP o ubicación de red. Modificar la política de implementación para incluir la eliminación de directorios confidenciales.
Cookie Attributes	Parametizar el atributo SameSite, con un valor de "estricto" debe usarse para cualquier cookie de sesión o aquella que contenga información confidencial y que pueda enviarse en una solicitud.
Privacy Policy Check	Establecer políticas de privacidad para obtener el consentimiento de los usuarios para almacenar o recuperar cualquier información en un equipo que gestione datos e información personal.

Esta tabla 3. Vector y Recomendación se realiza con el objetivo de suministrar recursos y herramientas a la entidad para tome medidas frente los hallazgos encontrados y se establezcan ejercicios como la implementación de políticas de seguridad en el desarrollo de código, implementar buenas prácticas de desarrollo y llegar a la implementación de automatización de despliegues continuos.

7. CONCLUSIONES

Se realiza la prueba de caja blanca o Static Application Security Testing (SAST) con la herramienta SonarQube, permitiendo evaluar el código fuente del sistema de gestión documental ORFEO, este análisis a fondo permitió hallar vulnerabilidades y fallas en el código reflejando 198 Bugs, 9 vulnerabilidades y 418 puntos de acceso de seguridad, 164 deudas técnicas o falta de mantenimiento del código, errores en las líneas de código y presenta un 86.25 % de duplicidad en el código.

Es de vital importancia tener o adurir conocimiento en herramientas de gestión de análisis de vulnerabilidades específicas, que se ajusten a la necesidad de la organización, para que estas sean un apoyo en tener la visión clara del software que se está desarrollando, dando oportunidad de mejora continua, en la optimización recursos y en la calidad, vinculando la seguridad como un componente principal para tal fin. Es de aclarar que el desarrollo del proyecto aportó conocimiento especifico en la instalación, validación y ejecución de pruebas SAST y DAST que se pueden aplicar a cualquier lenguaje y desarrollo.

Se recomienda establecer una política para la ejecución de estas pruebas y análisis de vulnerabilidades del software que se desarrolla in-house ya que en la Autoridad no se cuenta con dicho procedimiento, esto permitiría un estándar de aceptación y puesta en producción de las aplicaciones más seguras. Igualmente gestionar la integración con automatización y despliegue continuo con las herramientas establecidas para tal fin.

RECOMENDACIONES

Se debe priorizar de forma proactiva los problemas identificados para garantizar una mayor rapidez en la remediación de los hallazgos para estandarizar de forma segura el sistema de información y dar cumplimiento normativo según el estándar de la entidad. Por medio del uso de herramientas que permitan la automatización y despliegues seguros y continuos como Azure DevOps, que es totalmente compatible con las herramientas que se ejecutaron para el análisis del código fuente del ORFEO, SonarQube y Engine Scan.

Implementar la metodología DevSecOps en la entidad, la cual busca interiorizar la cultura de seguridad dentro de los estándares y procesos de desarrollo, en la gestión diaria con la inclusión de prácticas asociadas a seguridad, como el análisis de código seguro, estando estático o dinámico, en conjunto con las áreas involucradas (Desarrollo, seguridad y operación) para que logren estar en la misma sinergia para tener en conjunto un centro y una visión en común la cual es la búsqueda del aseguramiento en todas las fases o ciclo de vida de un proyecto.

Gestionar una política de aplicación de pruebas al código fuente de los desarrollos ejecutados en la entidad de forma constante para evaluar y detectar problemas con la recopilación de datos insegura formularios, presencia de cookies, enlaces de terceros, vulnerabilidades de secuencias de comandos entre sitios y vulnerabilidades de inyección SQL, para entregar un producto con valor agregado en calidad, seguridad y funcionamiento

Una vez la herramienta de Gestión Documental se encuentre en ambiente productivo, se puede utilizar que presentan versatilidad y funcionalidad que se ajustan a la necesidad de la entidad y con apoyo de las sugerencias de Open Web Aplication Security Proyect (OWASP) que es un proyecto de código abierto cuya finalidad es determinar y mitigar la inseguridad del software que presenta para proyectos Web.

BIBLIOGRAFÍA

ARCIHIVO GENERAL DE LA NACION. [Sitio Web]. Bogotá: AGN. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.archivogeneral.gov.co/sites/default/files/Estructura_Web/5_Consulte/Recursos/Publicacionees/ImplementacionSGDEA.pdf>

COLOMBIA. CONGRESO DE LA REPUBLICA. Ley 527. (18, agosto, 1999). Por lo cual se define y reglamenta el uso de datos y comercio electrónico. Bogotá: Congreso de la República, 1999.

COMPAÑ ROSIQUE, Patricia, *et al*. Explicando el bajo nivel de programación de los estudiantes: (Ejemplar dedicado a: Investigación en Docencia Universitaria de la Informática) [En línea]. Universidad de Alicante, 2018. [Consultado del 15 de mayo 2022]. Disponible en <https://dialnet.unirioja.es/servlet/articulo?codigo=6264614>

DEPARTAMENTO ADMINISTRATIVO DE LA FUNCIÓN PÚBLICA. [Sitio Web]. Bogotá: FUNCIONPUBLICA. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.funcionpublica.gov.co/web/eva/biblioteca-virtual/-document_library/bGsp2IjUBdeu/view_file/34268003>

DEVSECOPS LATINO AMERICA. Una guía práctica sobre la nueva era del testing de seguridad. Argentina: DevSecOps Argentina, 2020. E-book. [En línea]. (Recuperado en 20 de abril de 2022). Disponible en <https://docs.google.com/viewer?url=https://devsecops-latam.org/contenidos/ebook-devsecops-calms.pdf&embedded=true>

DIGITAL GUIDE IONOS. [Sitio Web]. España: IONOS. [Consulta: 13 de marzo 2022]. Disponible en: <https://www.ionos.es/digitalguide/servidores/seguridad/que-es-devsecops>

ESTRADA, ALBA, MARTIN. Fundamentos para implementar y certificar un sistema de gestión de seguridad informática bajo la norma ISO/IEC

27001. Serie científica de la Universidad de las Ciencias Informáticas. 2012

GÓMEZ, Laureano Felipe. Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir. *Códices* [en línea]. Bogotá (Colombia): Universidad de la Salle, 30 de junio de 2007, vol. 3, nro. 01 [Consultado 15, marzo,2022]. ISSN 1794-9815. Disponible en: <https://ciencia.lasalle.edu.co/co/vol3/iss1/2/>

MINISTERIO DE TECNOLOGIAS DE LA INFORMACION Y COMUNICACIONES. [Sitio Web]. Bogotá: MINTIC. [Consulta: 12 de marzo 2022]. Disponible en: <https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/149186:MinTIC-publica-el-Marco-de-Transformacion-Digital-para-mejorar-la-relacion-Estado-ciudadano>

ORFEO LIBRE ORG. [Sitio Web]. Bogotá: ORFEOLIBRE.ORG. [Consulta: 12 de marzo 2022]. Disponible en <https://orfeolibre.org/inicio/detalles-de-orfeo-sistema-de-gestion-documental>

OSPINA DELGADO, Juan Pablo. Análisis de seguridad y calidad de aplicaciones (Sonarqube) [En línea]. Tesis Maestría. Universidad Obterta de Cataluyna, 2015. [Consultado 15 mayo 2022]. Disponible en < http://openaccess.uoc.edu/webapps/o2/handle/10609/43263>

RAPID7. Escáner de vulnerabilidades Nexpose. [en línea]. [Consulta: 15 de mayo 2022]. Disponible en <https://www.rapid7.com/products/nexpose>

RANGANATHAN, S. R.: The Five Laws of Librarv Science. Bangalore: Sarada Ranganathan Endowment for Library Science, 1993 Estas leyes son completadas de cara a los sistemas de información, en relación con el coste y aprovechamiento del servicio, en Vickery.B. C., y Vickery, A. Bowker-Saur.1992, p. 260.

RED HAT. [Sitio Web]. Estados Unidos: RED HAT. [Consulta: 13 de marzo 2022]. Disponible en: <https://www.redhat.com/es/topics/devops/what-is-devsecops>

ROSENCRANCE, Linda. SecOps o DevSecOps. ComputerWeekly [en línea]. 2021, octubre, [Consultado 12 de marzo 2022]. Disponible en: <https://www.computerweekly.com/es/definicion/SecOps-o-DevSecOps?_gl=1*1olh05s*_ga*MjEwNzAwNzgzMC4xNjUzNzQ5NTIx*_ga_TQKE4GS5P9*MTY1Mzc0OTUyMS4xLjEuMTY1Mzc0OTYxNS4w&_ga=2.38138851.1378944387.1653749521-107007830.1653749521>

SINISTERRA, María Mercedes; HENAO DIAZ, Tania Marcela y LÓPEZ RUIZ, Erik Giancarlo. Clúster de balanceo de carga y alta disponibilidad para servicios web y mail. Informador técnico, [En línea]. Universidad de la Rioja, 2012. [Consultado 15 de mayo 2022]. Disponible en <http://Dialnet-ClusterDeBalanceoDeCargaYAltaDisponibilidadParaSer-4364562.pdfno 76, p. 93-102>

ANEXOS

ANEXO A

Autorización Empresa

Bogotá, 16 de 03 de 2022

Señor:
John Jairo
Jefe de Oficina Tecnologías de la Información

Asunto: Autorización para la ejecución del proyecto titulado: Remediación de Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad

Cordial saludo estimado Ingeniero,

Como es de su conocimiento, actualmente me encuentro adelantando estudios de posgrado en la Especialización en Seguridad Informática ofertado por la Universidad Nacional Abierta y a Distancia "UNAD". Para finalizar mi proceso académico es mi objetivo desarrollar un trabajo de grado aplicado a Autoridad
de manera que pueda aportar mis conocimientos adquiridos y generar un impacto positivo en la empresa, relacionado con los temas de Seguridad Informática, motivo por el cual, muy comedidamente solicito su autorización y aprobación para la ejecución del proyecto titulado: Remediación de Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad

El cual se encuentra avalado por parte la Institución de educación superior "UNAD".

El proyecto en su objetivo general describe lo siguiente: Realizar la Remediación de Vulnerabilidades aplicando el marco DevSecOps para mejorar la seguridad del software de gestión documental ORFEO, en la Autoridad para el año 2022;
al mismo tiempo será apoyado por los objetivos específicos:

1. Identificar las vulnerabilidades del sistema de información de gestión documental ORFEO en la infraestructura tecnológica de Autoridad por medio de pruebas de seguridad estáticas (SAST) con herramientas OpenSource como FindSecBugs.
2. Proponer los respectivos controles para mitigar el riesgo de las vulnerabilidades encontradas dentro del sistema de gestión documental ORFEO para la Autoridad
Gestionando las vulnerabilidades con herramientas OpenSource como ArcherySec.
3. Gestionar la implementación de los controles asociados al plan de mitigación de vulnerabilidades del gestor documental ORFEO para la Autoridad (Plan de gestión de implementación y diagnóstico de implementaciones ejecutadas)
4. Entregar la respectiva documentación, análisis y pruebas realizadas al sistema de información ORFEO, para que sean aplicadas conforme a las políticas establecidas por la Autoridad
Por medio de la formalización del cumplimiento del proyecto.

De obtener esta autorización, se elaborará un acuerdo de confidencialidad para proteger la identidad la empresa y sus activos de información; a su vez se destacan los siguientes procesos para ser garantes en la transaparencia de la ejecución del proyecto:

- Se prohíbe la ejecución de cualquier tipo de pruebas de seguridad que no estén autorizadas expresamente por Autoridad

- La empresa Autoridad
deberá establecer qué tipo de información es privada y cuál es pública para delimintar el acceso de pruebas en la ejecución del proyecto.
- La solicitud de información al igual que ejecución de prubas deben quedar por escrito y se genera un informe de resultados

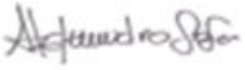
semanalmente el cual será compartido con el gerente de la organización o empresa.

- La persona autorizada siempre debe operar dentro de la ley 1273 de 2009 y de las demás regulaciones establecidas en la empresa.
- Respetar la privacidad de todos los individuos y mantener su privacidad en los reportes. Se encuentra prohibida la divulgación de información personal en tales reportes.

El resultado del proyecto se verá reflejado en un documento el cual será cargado al repositorio institucional de la Universidad Nacional Abierta y a Distancia "UNAD". El dcumento ampara la confidencialidad y anonimato de la empresa, estos aspectos se encuentran estipulados en el acuerdo de confidencialidad; agradezco el apoyo prestado en esta etapa de mi carrerar profesional.

Firman en Bogotá D.C., a los (18) días del mes de (marzo) de 2022

Cordialmente,



YAIR ALEJANDRO SOSA BARRETO
Estudiante UNAD.

John Jairo
Jefe de Oficina Tecnologías de la Información

ANEXO B

Acuerdo de Confidencialidad

ACUERDO DE CONFIDENCIALIDAD ENTRE YAIR ALEJANDRO SOSA BARRETO Y Autoridad

Por la **parte reveladora**
Nombre: Autoridad
Dirección:
Teléfono:
E-mail:

Por la parte **receptora de la información**
Nombre: Yair Aleiandro Sosa Barreto
Dirección: C
Teléfono: 3125937968
E-mail: ya

Identificación del proyecto

Entre los firmantes, identificados anteriormente, hemos convenido en celebrar el presente acuerdo de confidencialidad previa las siguientes **CONSIDERACIONES**

1. Que la información compartida en virtud del presente acuerdo pertenece a la Autoridad Nacional de Licencias Ambientales (ANLA), y la misma es considerada sensible y de carácter restringido en su divulgación, manejo y utilización. Dicha información es compartida en virtud del desarrollo del proyecto aplicado con el título: Remediación de Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad

2. Que la información de propiedad de Autoridad ha sido desarrollada u obtenido

legalmente, como resultado de sus procesos, programas o proyectos y, en consecuencias abarca documentos, datos, tecnología y/o material que considera

único y confidencial, o que es objeto de protección a título de secreto industrial.

3. Que el presente acuerdo se realiza por un lado entre la parte receptora de la información como integrante del proyecto de investigación y aplicación de Remediación de Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad Yair Alejandro Sosa Barreto que, para el presente caso actual como **revelador, guarda y administrados** de la información de propiedad de Autoridad

En consecuencia, **las partes** se suscriben a las siguientes cláusulas:

Primera. Objeto: en virtud del presente **acuerdo de confidencialidad,** la **parte receptora**, se obliga a no divulgar directa, indirecta, próxima a remotamente, ni a través de ninguna otra persona o de sus subalternos o funcionarios, asesores o cualquier persona relacionada con ella, la **información confidencial** perteneciente a la Autoridad así como también a no utilizar dicha información en beneficio propio ni de terceros, sólo con fines estadísticos y de mejoramiento de la Autoridad

Segunda. Definición de información confidencial: se entiende como **Información Confidencial**, para los efectos del presente acuerdo:

1. La información que no sea pública y sea conocida por la **parte receptora** con ocasión de del proyecto de investigación y/ extensión.

2. Cualquier información societaria, técnica, jurídica, financiera, comercial, de mercado, estratégica, de productos, nuevas tecnologías, patentes, modelos de utilidad, diseños industriales, modelos de negocios, información del personal de la organización

y/o cualquier otra relacionada con el proyecto Remediacion de Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad
lograr tales fines, y/o cualquier otro ente relacionado con la estructura organizacional, bien sea que la misma sea escrita, oral o visual, o en cualquier forma tangible o no, incluidos los mensajes de datos (en la forma definida en la ley), de la cual, la **parte receptora** tenga conocimiento o a la que tenga acceso por cualquier medio o circunstancia en virtud de las reuniones sostenidas y/o documentos suministrados.

3. La que corresponda o deba considerarse como tal para garantizar el derecho constitucional a la intimidad, la honra y el buen nombre de las personas y deba guardarse la debida diligencia en su discreción y manejo en el desempeño de sus funciones.

Tercera. Origen de la información confidencial: provendrá de documentos suministrados en el desarrollo del proyecto y que tiene que ver con las creaciones del intelecto, a la naturaleza, medios, formas de distribución, comercialización de productos o de prestación de servicios, transmitida verbal, visual o materialmente, por escrito en los documentos, medios electrónicos, discos ópticos, microfilmes, películas, e-mail u otros elementos similares suministrados de manera tangible o intangible, independiente de su fuente o soporte y sin que requiera advertir su carácter confidencial.

Cuarta. Obligaciones de la parte receptora: Se considerará como **parte receptora** de la **información confidencial** a la persona que recibe la información, o que tenga acceso a ella. La parte receptora se obliga a:

De ser necesario o conveniente según la necesidad del titular de la información, se adicionarán las obligaciones que se consideren pertinentes:

1. Mantener la **información confidencial** segura, usarla solamente para los propósitos relacionados con él, en caso de ser solicitada, devolverla toda (incluyendo copias de esta) en el momento en que ya no requiera hacer uso de la misma o cuando termine la

2. Proteger la **información confidencial**, sea verbal, escrita, visual, tangible, intangible o que por cualquier otro medio reciba, siendo legitima poseedora de la misma Autoridad
restringiendo su uso exclusivamente a las personas que tengan absoluta necesidad de conocerla.

3. Abstenerse de publicar la **información confidencial** que conozca, reciba o intercambie con ocasión de las reuniones sostenidas.

4. Usar la **información confidencial** que se le entregue, únicamente para los efectos señalados al momento de la entrega de dicha información.

5. Mantener la **información confidencial** en reserva hasta tanto adquiera el carácter de pública.

6. Responder por el mal uso que le den sus representantes a la **información confidencial.**

7. Guardar la reserva de la **información confidencial** como mínimo, con el mismo cuidado con la que protege la **información confidencial.**

8. La **parte receptora** se obliga a no transmitir, comunicar revelar o de cualquier otra forma divulgar total o parcialmente, pública o privadamente, la **información confidencial** sin el previo consentimiento por escrito por parte de Autoridad

9. La **parte receptora** se compromete a establecer que los datos a utilizar son: direccionamiento IP, nombres de los servidores a explorar, nombres de dispositivos de seguridad.

10. La información capturada por la **parte receptora** se observará como datos de valoración de vulnerabilidades para ajustes de

políticas de seguridad informática, no existirá ningún tipo de ganancia económica, es netamente educativo.

11. La identidad de toda persona de la Autoridad no será revelada, dado que no se capturará sus nombres completos ni algún otro tipo de información que revele su identidad física o digital.

12. Las pruebas realizadas por la **parte receptora** nunca pondrán en peligro los activos tecnológicos de Autoridad ni violentará la ley de delitos informáticos colombiana 1273 de 2009 estando en el margen de las buenas prácticas y los procesos legales pertinentes.

13. El estudiante Yair Alejandro Sosa Barreto se compromete a difuminar, bloquear y ocultar toda información que revele la identidad de la empresa Autoridad para salvaguardar la confidencialidad e identidad de la empresa en el documento final del proyecto el cual será publicado en el repositorio institucional y de acceso público.

14. El título del proyecto no podrá contener el nombre de la empresa u organización con la que se firma el presente acuerdo de confidencialidad, este nombre deberá ser reemplazado.

Parágrafo: Cualquier divulgación autorizada de la **información confidencial** a terceras personas estará sujeta a las mismas obligaciones de confidencialidad derivadas del presente **Acuerdo** y la **parte receptora** deberá informar estas restricciones incluyendo la identificación de la información cono confidencial.

Quinta. Obligaciones de la parte reveladora: Son obligaciones de la parte reveladora:

1. Mantener la reserva de la **información confidencial** hasta tanto adquiera el carácter de pública.

2. Documentar toda la **información confidencial** que transmita de manera escrita, oral o visual, mediante documentos, medios electrónicos, discos ópticos, microfilmes, películas, e-mails u otros elementos similares o en cualquier forma tangible o no, incluidos los mensajes de datos, como registro de la misma para la determinación de sus alcances, e indicar específicamente y de manera clara e inequívoca el carácter confidencia de la información suministrada de la **parte receptora**.

Sexta. Exclusiones a la confidencialidad: La **parte receptora** queda relevada o eximida de la obligación de confidencialidad, únicamente en los siguientes casos:

1. Cuando la **información confidencial** haya sido o sea de dominio público. Si la información se hace de dominio público durante el plazo del presente acuerdo, por un hecho ajeno a la **parte receptora**, esta conservará su deber de reserva sobre la información que no haya sido afectada.

2. Cuando la **información confidencial** deba ser revelada por sentencia en firme de un tribunal o autoridades competentes en desarrollo de sus funciones que ordenen el levantamiento de la reserva y soliciten el suministro de esta información. No obstante, en este caso la parte reveladora será la encargada de dar cumplimiento a la orden, restringiendo la divulgación a la información estrictamente necesaria, y en el evento de que la confidencialidad se mantenga, no eximirá a la parte receptora del deber de reserva.

3. Cuando la **parte receptora pruebe** que la **información confidencial** ha sido obtenida por otras fuentes.

4. Cuando la **información confidencial** ya la tenía en su poder la parte receptora antes de la entrega de la información reservada.

Séptima. Responsabilidad: la parte que contravenga el acuerdo será responsable ante la otra parte o ante los terceros de buena fe sobre los cuales se demuestre que se han visto afectados por la inobservancia del presente **acuerdo**, por los perjuicios morales y económicos que

estos puedan sufrir como resultado del incumplimiento de las obligaciones aquí contenidas.

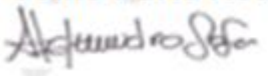
Octava. Solución de controversias: Las partes Yair Aleiandro Sosa Barreto - Autoridad Oficina de Tecnologías de la Información se comprometen a esforzarse en resolver mediante los mecanismos alternativos de solución de conflictos cualquier diferencia que surja con motivo de la ejecución del presente **acuerdo.** En caso de no llegar a una solución directa para la controversia planteada, someterán la cuestión controvertida a las leyes colombianas y a la jurisdicción competente en el momento de presentarse la diferencia. La Universidad Nacional Abierta y a Distancia como institución educativa no se hace responsable del no cumplimiento de las cláusulas del presente acuerdo de confidencialidad por parte de Yair Alejandro Sosa Barreto.

Novena. Legislación aplicable: Este **acuerdo** se regirá por las leyes de la República de Colombia y se interpretará de acuerdo con las mismas.

Décima. Aceptación del Acuerdo: Las partes han leído y estudiado de manera detenida los términos y el contenido del presente **Acuerdo** y por tanto manifiestan estar conformes y aceptan todas las condiciones.

Firman en Bogotá D.C., a los (18) días del mes de (marzo) de 2022

Como Parte Receptora:



Yair Alejandro Sosa Barreto

Estudiante UNAD.
C.C. No.80.002.937 de Bogotá

Por la parte reveladora:

John Jairo
Jefe de oficina de Tecnologías de la Información
Autoridad

C.C. Bogotá

ANEXO C

Documento Oficial del análisis del proyecto

Bogotá, 1 de Dic de 2022

Señor:
Jhon Jairo
Jefe de Oficina Tecnologías de la Información

Asunto: Entrega Documentación Oficial proyecto titulado: Estrategias de mejora que permitan mitigar las Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad

Cordial saludo estimado Jefe,

Con el objetivo de cumplir a cabalidad con los compromisos adquiridos referente a la ejecución del proyecto titulado: Estrategias de mejora que permitan mitigar las Vulnerabilidades del aplicativo Gestor Documental ORFEO en la entidad Autoridad
cuyo objetivo general era: Proponer acciones que permitan mitigar las vulnerabilidades del software de gestión documental ORFEO aplicando el marco DevSecOps para mejorar la seguridad en la Autoridad
al mismo tiempo seria apoyado por los objetivos especificos:

1. Analizar las vulnerabilidades del sistema de información de gestión documental ORFEO en la infraestructura tecnológica de la Autoridad por medio de la ejecución de pruebas de seguridad (SAST) y (DAST) en el marco de las etapas del DevSecOps.

2. Proponer recomendaciones para mitigar el riesgo de las vulnerabilidades encontradas dentro del sistema de gestión documental ORFEO para la Autoridad
aplicando la evaluación de los ítems identificados.

3. Elaborar la documentación sobre el análisis y pruebas realizadas al sistema de información ORFEO, para que sean aplicadas conforme a las políticas establecidas por la Autoridad
, por medio de la formalización del cumplimiento del proyecto. Por medio de la formalización del cumplimiento del proyecto.

En el presente apartado se formaliza la documentación técnica de los eventos encontrados por las pruebas realizadas de la siguiente forma:

SOFTWARE APLICADO PARA LAS PRUEBAS

Static Application Security Testing (SAST): Realizadas con el Software Libre SonarQube versión 9.4, instalado local para la ejecución del análisis del código fuente.

S.O: Windows 11

Procesador: Xeon 2.0 Ghz

Memoria: 64 RAM

HD:1 TB

Dynamic Application Security Testing (DAST): Realizadas con el aplicativo Nexpose Scan Engine 1.0.48 de Rapid 7, instalado el agente en un servidor de la red corporativa suministrado por el personal de Infraestructura Tecnológica para la ejecución del análisis del código fuente.

S.O: Windows Server 2019

Procesador: Xeon 2.4 Ghz

Memoria: 12 RAM

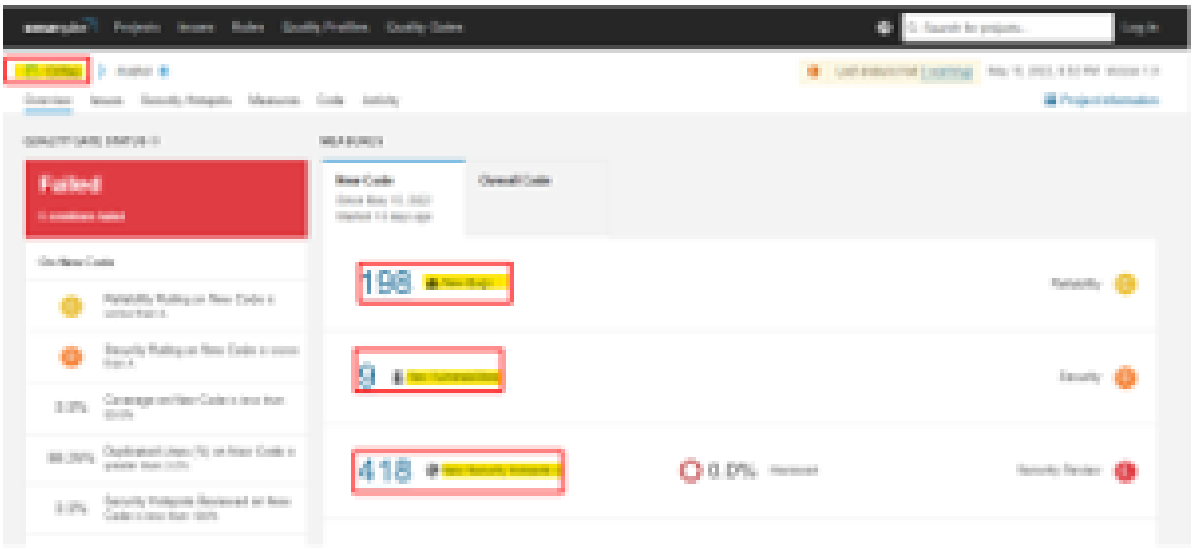
HD: 200 GB

RESULTADOS

Static Application Security Testing (SAST)

Con el uso del Software Libre SonarQube versión 9.4, se evidencia 198 Bugs, 9 vulnerabilidades y 418 puntos de acceso de seguridad, 164 deudas técnicas o falta de mantenimiento del código, errores en las líneas de código y presenta un 86.25 % de duplicidad en el código. Como se evidencia en la ilustración 1.

Ilustración 1. Resultados de Análisis Finales

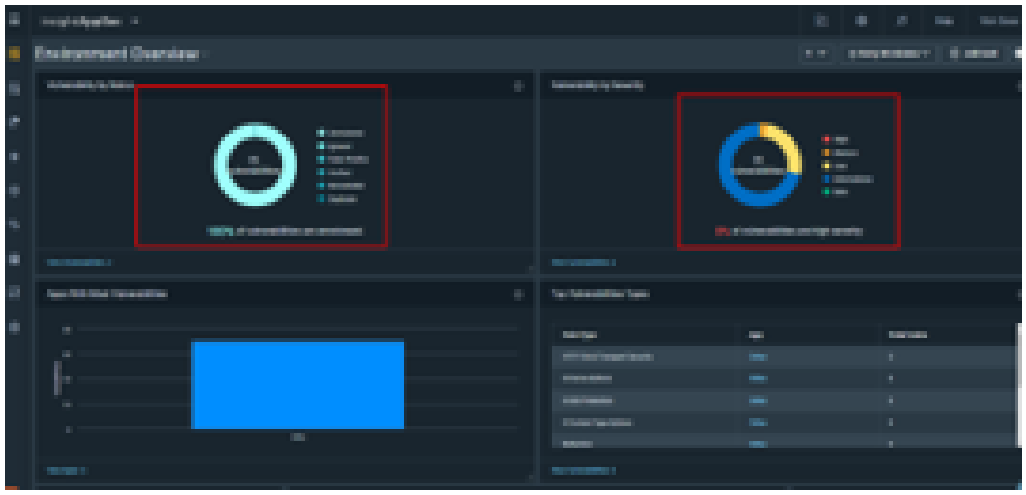


Fuente1. Elaboración propia, captura de pantalla

Dynamic Application Security Testing (DAST)

Con el uso del software propietario, con licencia perteneciente a la entidad Nexpose Scan Engine 1.0.48 se evidencia que el aplicativo con las pruebas automáticas dinámicas al aplicativo ORFEO refleja 35 vulnerabilidades, entre ellas fijación de sesión, datos expuestos, falsificación de sitio, etc. Como se observa en la ilustración 49.

Ilustración 2. Análisis Vulnerabilidades DAST Finales



Fuente 2. Elaboración propia, captura de pantalla

Esta información se ha recopilado durante análisis de la aplicación web, validando propiedades en línea del código con el objetivo de detectar problemas como la recopilación de datos insegura formularios, presencia de cookies, enlaces de terceros, vulnerabilidades de secuencias de comandos entre sitios y vulnerabilidades de inyección SQL. El análisis presentado revela fallos de cumplimiento y por tanto, se debe priorizar de forma proactiva los problemas identificados para garantizar una mayor rapidez la remediación de los hallazgos para estandarizar de forma segura el Sistema de información y dar cumplimiento normative según el estándar de la entidad.

Las vulnerabilidades halladas están relacionadas con el Top 10 de Open Web Application Security OWASP, igualmente con Common Weakness Enumeration CWE y Common Vulnerability Exposure CVE que son listados o bases de conocimiento a nivel mundial de debilidades y vulnerabilidades que contribuye la organización MITRE ATT&CK.

Tabla 1 VULNERABILIDADES, ATAQUES y MITIGACIONES

Vulnerabilidad	Tipo	Vector
CWE-1006	Malas prácticas de codificación	Add curly braces around the nested statement **Ataque:** Prácticas de codificación de consideración inseguras y con posibilidades que sea explotada una vulnerabilidad, indica que el producto no se ha desarrollado cuidadosamente en aspectos seguros. Deja espacio para que un atacante realice inserción de código aprovechando que las llaves {} no se cierran, esta inserción podría llevar al usuario a una página maliciosa o URL no valida y capturar datas sensibles. Genera errores al momento de ejecución del código. **Mitigación:** Aplicar las políticas de buenas prácticas, no dejando líneas de código abiertas, cerrando las sentencias de ejecución. Adoptar conjuntos de estándares de codificación para el equipo de trabajo.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CVE-2015-4024	Vulnerabilidad de complejidad algorítmica (PHP)	Refactor this function to reduce its Cognitive Complexity **Ataque:** Esta vulnerabilidad permite a los atacantes de forma remota provocar alguna denegación de servicio por consumo de recursos del equipo, bien sea procesador o memoria usando formularios para el crecimiento inadecuado del flujo lineal del código e incrementar las estructuras que rompen el flujo estando anidadas. versión de PH 5.4, 5.5 y 5.6 **Mitigación:** Actualización de cambio de PHP 5. Gestionar las mejores prácticas de escritura, desarrollo y manteamiento del código reduciendo la complejidad por los métodos a nivel de clases y aplicaciones.
CWE-295	Validación de Certificado Incorrecta	Enable server certificate validation on this SSL/TLS **Ataque:** Cuando un certificado no es válido, permite que cualquier atacante suplante el servidor web o la entidad de confianza para conectar a un host malicioso.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-295	Validación de Certificado Incorrecta	Enable server certificate validation on this SSL/TLS **Mitigación:** Implementar certificados válidos para la encriptación de datos.
CWE:327	Algoritmo Criptográfico en Riesgo	Enable server certificate validation on this SSL/TLS **Ataque:** El atacante puede descifrar el algoritmo y comprometer datos confidenciales y data sensible. **Mitigación:** Utilizar algoritmos o mecanismos de cifrado simétrico o asimétricos con hashes robustos SHA2, SHA3, SHA-256.
A02:2021	Fallas Criptográficas	Enable server certificate validation on this SSL/TLS **Ataque:** Se puede ejecutar el monitoreo de trafico de red, degradando las conexiones HTTPS interceptando solicitudes y secuestrando la sesión. **Mitigación:** Cifrar la data con protocolos seguros como TLS.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-457	Uso de Variable no inicializada	BUGS **Ataque:** Se puede controlar o leer los contenidos de la memoria de pila generando implicaciones de seguridad pre inicializando la variable para inyectar código. **Mitigación:** Asignación de valores a las variables, gestionar bibliotecas de cadenas seguras y abstracción de contenedores.
CWE-561	Código Muerto	BUGS **Ataque:** Invocación del código inactivo puede generar bucles infinitos, generar cadenas constantes en los nombres de un campo relevantes. **Mitigación:** Eliminar, depurar el código inactivo y ejecutar escaneos con herramientas especializadas.
CWE-628	Invocar funciones con argumentos incorrectamente	BUGS **Ataque:** Invocar funciones o rutinas no específicas, generando comportamiento incorrecto suministrando al atacante recursos del sistema. **Mitigación:** Ejecutar el compilador para validar la falla, observando el código que no se despliegue en el control.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CVE-2021-42536	Exposición de Recursos e Información Confidencial	Password detected in this variable **Ataque:** Información vulnerable a la divulgación del nombre de usuario y la contraseña del par al permitir que todos los usuarios accedan para leer variables globales, pueden escalar privilegios. **Mitigación:** Cambiar la variable en el código, sin registro de visibilidad o texto claro. Usar cifrado de datos para encriptar las contraseñas.
CWE-1041	Uso Código Redundante	Duplicated Lines **Ataque:** Invocar funciones o rutinas no específicas, generando comportamiento incorrecto en el sistema de información, elevando el consumo de recursos del servidor, generando tiempos de espera en respuesta debido al procesamiento de las líneas de código redundantes, haciendo la mantenibilidad del código compleja, con tendencia a errores y no ejecución de cambios realizados. **Mitigación**: Mantener las buenas prácticas en medidas de calidad de desarrollo en el código.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-1041	Uso Código Redundante	Duplicated Lines **Mitigación:** Depurar las líneas de código duplicadas que no ejecuten procedimientos o manejo de variables de datos. Implementar funcionalidades frecuentes en una sola, para hacer un llamado desde la base del código de ser necesario y requerido.
CWE-259 CWE-321 CWE-798	HardCoded	HardCoded (Código Quemado) **Ataque:** el atacante puede tener acceso a los valores estipulados o preestablecidos y explotar fallas de seguridad cambiando las credenciales o el HardCoded. **Mitigación:** Usar clases constantes para el alanceando de los valores. Implementar métodos de inicio de sesión para que el usuario ingrese una contraseña segura.
OWASP A07:2021	Identificación y fallos de Autenticación	Cross-site-Scripting (XSS) **Ataque:** La variable kid opera un parámetro de scripts
OWASP A03:2021	Inyección	inserción malicioso vulnerabilidad activa directo con el Front del aplicativo.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
OWASP A07:2021	Identificación y fallos de Autenticación	Cross-site-Scripting (XSS) **Ataque:** Dejando que el atacante ejecute HTML y JavaScript en el navegador de la víctima, cuando se engaña a un usuario para que haga clic en el enlace malicioso, el código inyectado se dirige al sitio web vulnerable. **Mitigación:** Cumplir con el filtrado estricto de las codificaciones de caracteres HTML. Filtrar toda la información enviada al servidor a través de formularios POST/GET y parámetros de consulta de URL con un énfasis particular en el filtrado de caracteres específicos de HTML.
OWASP A03:2021	Inyección	
CWE-384	Fijación de Sesión	Session Fixation **Ataque:** Un atacante secuestra una sesión de usuario válida del valor de la cookie de ID de sesión para que pueda continuar usando el mismo valor de cookie. El atacante puede determinar el nombre y el valor de la ID de la sesión, donde el servidor acepta cualquier peticion del atacante, este visita el sitio y rastrea en la respuesta llevando a la victima a una web maliciosa.
OWASP A07:2021	Identificación y fallos de Autenticación	
OWASP A02:2021	Fallos Criptográficos	

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-384	Fijación de Sesión	Session Fixation **Mitigación:** Metodo para cambiar la ID de sesión cuando los usuarios inician sesión.
OWASP A07:2021	Identificación y fallos de Autenticación	No aceptar ID de sesión en los parámetros GET o POST.
OWASP A02:2021	Fallos Criptográficos	Usar cookies HTTP como para codificar la información de la sesión. Regenerar el SID por en los sistemas que lo admiten, usando identificadores de sesión SSL/TLS.
CWE-319	Transmisión de Informacion Sensible por Texto Claro	Sensitive Data Exposure **Ataque:** El atacante puede indicir a un usuario web que intente actualizar la respuesta del servidor en ciertos agentes de usuario puede hacer que se vuelva a enviar el contenido de la solicitud HTTP POST original.
OWASP A02:2021	Fallos Criptográficos	**Mitigación:** Aplicar un patrón de diseño de desarrollo web que evita algunos envíos de formularios duplicados como es Post/Redirect/Get (PRG) el cual permite crear una interfaz más intuitiva para los agentes de usuario.
OWASP A03:2021	Inyección	

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-319	Transmisión de Informacion Sensible por Texto Claro	Sensitive Data Exposure **Mitigación:** PRG implementa marcadores y el botón de actualización de una manera predecible que no crea envíos de formularios duplicados.
OWASP A02:2021	Fallos Criptográficos	
OWASP A03:2021	Inyección	
CWE-16	Configuración	Content Security Policy Header **Ataque:** Se puede explotar la confianza del navegador en el contenido recibido del servidor. Los scripts maliciosos son ejecutados por el navegador de la víctima por la confianza creada en el navegador en la fuente del contenido de servidor. **Mitigación:** Aplicar politicas de contenido Seguro (CSP) ya que un navegador compatible con CSP solo ejecutará secuencias de comandos cargadas en archivos de origen recibidos de esos dominios incluidos en la lista de permitidos.
OWASP A05:2021	Configuración Incorrecta de Seguridad	

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-16	Configuración	Content Security Policy Header **Mitigación**: Aplicar las CSP hacen posible se reduzcan los vectores por los cuales XSS puede ocurrir al especificar los dominios que el navegador debe considerar como fuentes válidas de scripts ejecutables.
OWASP A05:2021	Configuración Incorrecta de Seguridad	
CWE-352	Cross-site Request Forgery (CSRF)	Cross-site Request Forgery (CSRF) **Ataque:** Un ataque engaña a la víctima para que cargue una página que contiene una solicitud maliciosa, esta hereda la identidad y los privilegios de la víctima para realizar una función no deseada en nombre de la víctima, como cambiar la dirección de correo electrónico, la dirección particular o la contraseña de la víctima. **Mitigación:** Requerir que el cliente proporcione datos de autenticación en la misma solicitud HTTP utilizada para realizar cualquier operación con implicaciones de seguridad.
OWASP A01:2021	Control de Acceso Roto	Limitar la vida útil de las cookies de sesión.

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-352	Cross-site Request Forgery (CSRF)	Cross-site Request Forgery (CSRF) **Mitigación:** Utilizar un filtro CSRF como el CSRFGuard de OWASP. El filtro intercepta las respuestas, detecta si se trata de un documento html e inserta un token en los formularios y, opcionalmente, inserta un script para insertar tokens en las funciones ajax.
OWASP A01:2021	Control de Acceso Roto	
CWE-530	Exposición de Ruta de Archivo	Predictable Resource Location **Ataque:** Según el contenido del archivo, puede revelar interfaces de administrador o direcciones URL alternativas que se supone que están ocultas para los usuarios y el atacante puede capturar informacion para explotar un ataque, ejemplo: conociendo la version del PHP. **Mitigación:** Restrinjir el acceso a directorios confidenciales (por ejemplo, admin) por contraseña y dirección IP o ubicación de red. Modificar la política de implementación para incluir la eliminación de directorios confidenciales.
OWASP A01:2021	Control de Acceso Roto	
OWASP A05:2021	Configuración Incorrecta de Seguridad	

Tabla 1 (Continuación)

Vulnerabilidad	Tipo	Vector
CWE-614 OWASP A05:2021	Cookie Confidencial en la sesión HTTPS Configuración Incorrecta de Seguridad	Cookie Attributes **Ataque:** Un atacante secuestrar una sesión de usuario válida. el estado de autenticación del valor de la cookie de ID de sesión para que pueda continuar usando el mismo valor de cookie. **Mitigación:** Parametizar el atributo SameSite, con un valor de "estricto" debe usarse para cualquier cookie de sesión o aquella que contenga información confidencial y que pueda enviarse en una solicitud.
CWE-1211 OWASP A07:2021	Errores de Autenticación Identificación y fallos de Autenticación	Privacy Policy Check **Ataque:** los identificadores ee linea pueden dejar rastros que, en particular cuando se combinan con identificadores únicos y otra información recibida por los servidores, por tanto, el atacante puede usar esos rastros para crear perfiles de las personas físicas e identificarlas. **Mitigación:** Establecer política de privacidad para obtener el consentimiento de los usuarios para almacenar cualquier información personal en un equipo que gestione datos.

Tabla 1 (Continuación)

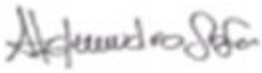
Vulnerabilidad	Tipo	Vector
OWASP A07:2021	Identificación y fallos de Autenticación	Reflection **Ataque:** La variable krd opera un parámetro de scripts inserción malicioso vulnerabilidad activa directo con el Front del aplicativo, dejando que el atacante ejecute HTML y JavaScript en el navegador de la víctima, cuando se engaña a un usuario para que haga clic en el enlace malicioso, el código inyectado se dirige al sitio web vulnerable. **Mitigación:** Cumplir con el filtrado estricto de las codificaciones de caracteres HTML. Filtrar toda la información enviada al servidor a través de formularios POST/GET y parámetros de consulta de URL con un énfasis particular en el filtrado de caracteres específicos de HTML
OWASP A03:2021	Inyección	

Se concluye que la herramienta ORFEO cuenta con varias vulnerabilidades significativas que deben ser atendidas con prontitud, si la herramienta se despliega en producción, ya que estará expuesta y sensible a varios ataques que de ser materializados o explotados generaran un gran impacto en los procesos operativos de la entidad, ya que este sistema de información es transversal a toda la gestión de esta.

Quedo a su disposición para atender cualquier solicitud referente al proyecto y poder aportar mis conocimientos a la gestión segura de sistema de información Gestor Documental ORFEO

Firman en Bogotá D.C., a los (1) días del mes de (Diciembre) de 2022

Cordialmente,



YAIR ALEJANDRO SOSA BARRETO
Estudiante UNAD.

Jhon Jairo
Jefe de Oficina Tecnologías de la Información

Estructura del documento para la estructura del Resumen Analítica Especializado -RAE

Fecha de Realización:	20/12/2022
Programa:	Especialización En Seguridad Informática
Línea de Investigación:	Proyecto Aplicado
Título:	Estrategias de mejora que permitan mitigar las vulnerabilidades del aplicativo gestor documental ORFEO en la Autoridad.
Autor(es):	Yair Alejandro Sosa Barreto
Palabras Claves:	DevSecOps, Vulnerabilidades, Mitigación, ORFEO
Descripción:	El presente trabajo tiene como finalidad el fortalecimiento a nivel de seguridad informática del aplicativo de Gestión Documental (ORFEO) desarrollado por la Super Intendencia de Servicio Públicos Domiciliarios (SSPD) debido a que está solución esta implementada en la mayoría de las entidades estatales del país. Para este efecto se hará el análisis de pruebas *Static Application Security Testing* (SAST) y *Dynamic Application Security Testing* (DAST) al aplicativo ORFEO para la Autoridad. Por tal motivo se buscará implementar mejoras de seguridad en los procesos, actividades o tareas específicas dentro del aplicativo ORFEO. Para dicho fin se realizará un análisis bajo el marco de colaboración DevSecOps cuya finalidad es lograr la planificación, desarrollo, automatización y monitoreo bajo normas de seguridad en todas las fases del ciclo de vida de un software.

Fuentes bibliográficas destacadas:

DEVSECOPS LATINO AMERICA. Una guía práctica sobre la nueva era del testing de seguridad. Argentina: DevSecOps Argentina, 2020. E-book. [En línea]. (Recuperado en 20 de abril de 2022). Disponible en <https://docs.google.com/viewer?url=https://devsecops-latam.org/contenidos/ebook-devsecops-calms.pdf&embedded=true>

GÓMEZ, Laureano Felipe. Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir. Códices [en línea]. Bogotá (Colombia): Universidad de la Salle, 30 de junio de 2007, vol. 3, nro. 01 [Consultado 15, marzo,2022]. ISSN 1794-9815. Disponible en: <https://ciencia.lasalle.edu.co/co/vol3/iss1/2/>

OSPINA DELGADO, Juan Pablo. Análisis de seguridad y calidad de aplicaciones (Sonarqube) [En línea]. Tesis Maestría. Universidad Obterta de Cataluyna, 2015. [Consultado 15 mayo 2022]. Disponible en < http://openaccess.uoc.edu/webapps/o2/handle/10609/43263>

RANGANATHAN, S. R.: The Five Laws of Librarv Science. Bangalore: Sarada Ranganathan Endowment for Library Science, 1993 Estas leyes son completadas de cara a los sistemas de información, en relación con el coste y aprovechamiento del servicio, en Vickery.B. C., y Vickery, A. Bowker-Saur.1992, p. 260

Contenido del documento:	El documento consta de la definición del problema bajo los antecedentes de este junto con la pregunta problema, la justificación del proyecto y los objetivos tanto general como específicos, igualmente, el marco referencial que contiene los marcos teóricos, conceptual, legal y antecedentes. Luego se ejecuta el desarrollo de cada actividad con las cuales se lograron los objetivos, general y específicos, se presenta el análisis de pruebas realizadas al aplicativo junto con todos los documentos anexos del mismo como son análisis del proyecto, acuerdo de confidencialidad y la autorización de la empresa.
Marco Metodológico:	Se gestiono bajo el Marco de operación DevSecOps
Conceptos adquiridos:	Se adquirieron conocimientos en la instalación y ejecución de software para las pruebas *Static Application Security Testing* (SAST), igualmente en el análisis del código fuente junto con las vulnerabilidades halladas, se aprendió a investigar la mitigación de vulnerabilidades específicas de código fuente.
Conclusiones:	Buscar la Implementación de la metodología DevSecOps, la cual busca interiorizar la cultura de seguridad dentro de los estándares y procesos de desarrollo, en la gestión diaria con

	la inclusión de prácticas asociadas a seguridad, como el análisis de código seguro, estando estático o dinámico, en conjunto con las áreas involucradas (Desarrollo, seguridad y operación) para que logren estar en la misma sinergia para tener en conjunto un centro y una visión en común la cual es la búsqueda del aseguramiento en todas las fases o ciclo de vida de un proyecto, mejorar de forma óptima los procesos de cualquier entidad