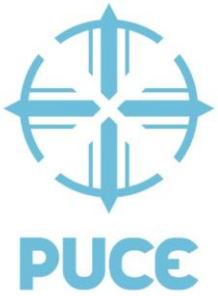


PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

ESCUELA DE INFORMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

ANÁLISIS FORENSE DE CÓDIGO Y PROCESOS DE LA APLICACIÓN UBIDESK DE LA EMPRESA UBORATECH

BRYAN ANDERSON VILAÑEZ ORDOÑEZ

TUTOR: GALO HERNÁN PUETATE HUERA

IBARRA – ECUADOR

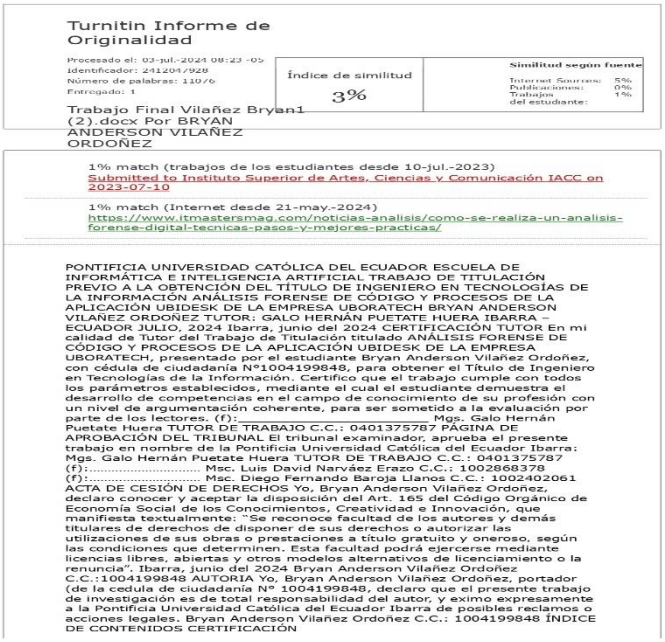
JULIO, 2024

Ibarra, Julio del 2024

CERTIFICACIÓN TUTOR

En mi calidad de Tutor del Trabajo de Titulación titulado **ANÁLISIS FORENSE DE CÓDIGO Y PROCESOS DE LA APLICACIÓN UBIDESK DE LA EMPRESA UBORATECH,** presentado por el estudiante Bryan Anderson Vilañez Ordoñez**,** con cédula de ciudadanía N°1004199848, para obtener el Título de Ingeniero en Tecnologías de la Información. Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores.

Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN.



Turnitin Informe de Originalidad

Procesado el: 03-jul.-2024 08:23 -05
Identificador: 2412047928
Número de palabras: 11076
Entregado: 1

Trabajo Final Vilañez Bryan1 (2).docx Por BRYAN ANDERSON VILAÑEZ ORDOÑEZ

Índice de similitud: 3%

Similitud según fuente
Internet Sources: 5%
Publicaciones: 0%
Trabajos del estudiante: 1%

1% match (trabajos de los estudiantes desde 10-jul.-2023)
Submitted to Instituto Superior de Artes, Ciencias y Comunicación IACC on 2023-07-10

1% match (Internet desde 21-may.-2024)
https://www.itmastersmag.com/noticias-analisis/como-se-realiza-un-analisis-forense-digital-tecnicas-pasos-y-mejores-practicas/

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR ESCUELA DE INFORMÁTICA E INTELIGENCIA ARTIFICIAL TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN ANÁLISIS FORENSE DE CÓDIGO Y PROCESOS DE LA APLICACIÓN UBIDESK DE LA EMPRESA UBORATECH BRYAN ANDERSON VILAÑEZ ORDOÑEZ TUTOR: GALO HERNÁN PUETATE HUERA IBARRA – ECUADOR JULIO, 2024 Ibarra, junio del 2024 CERTIFICACIÓN TUTOR En mi calidad de Tutor del Trabajo de Titulación titulado ANÁLISIS FORENSE DE CÓDIGO Y PROCESOS DE LA APLICACIÓN UBIDESK DE LA EMPRESA UBORATECH, presentado por el estudiante Bryan Anderson Vilañez Ordoñez, con cédula de ciudadanía N°1004199848, para obtener el Título de Ingeniero en Tecnologías de la Información. Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores. (f):..... Mgs. Galo Hernán Puetate Huera TUTOR DE TRABAJO C.C.: 0401375787 PÁGINA DE APROBACIÓN DEL TRIBUNAL El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra: Mgs. Galo Hernán Puetate Huera TUTOR DE TRABAJO C.C.: 0401375787 (f):..... Msc. Luis David Narváez Erazo C.C.: 1002868378 (f):..... Msc. Diego Fernando Baroja Llanos C.C.: 1002402061 ACTA DE CESIÓN DE DERECHOS Yo, Bryan Anderson Vilañez Ordoñez, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: "Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia". Ibarra, junio del 2024 Bryan Anderson Vilañez Ordoñez C.C.:1004199848 AUTORIA Yo, Bryan Anderson Vilañez Ordoñez, portador (de la cedula de ciudadanía Nº 1004199848, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales. Bryan Anderson Vilañez Ordoñez C.C.: 1004199848 ÍNDICE DE CONTENIDOS CERTIFICACIÓN

(f):_____
Mgs. Galo Hernán Puetate Huera
TUTOR DE TRABAJO
C.C.: 0401375787

PÁGINA DE APROBACIÓN DEL TRIBUNAL

El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra:

Mgs. Galo Hernán Puetate Huera
TUTOR DE TRABAJO
C.C.: 0401375787

(f):_____…

Msc. Luis David Narváez Erazo

C.C.: 1002868378

(f):……

Msc. Diego Fernando Baroja Llanos

C.C.: 1002402061

ACTA DE CESIÓN DE DERECHOS

Yo, Bryan Anderson Vilañez Ordoñez, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: “Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia”.

Ibarra, junio del 2024

Bryan Anderson Vilañez Ordoñez

C.C.:1004199848

AUTORIA

Yo, Bryan Anderson Vilañez Ordoñez, portador (de la cedula de ciudadanía N° 1004199848, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales.

Bryan Anderson Vilañez Ordoñez

C.C.: 1004199848

DEDICATORIA

Este trabajo lo dedico en primer lugar a Dios y a la Virgen del Quinche por la vida, la salud, por sus múltiples bendiciones, guiando mis pasos y fortaleciendo mi fe durante este largo camino académico y no permitir que los desafíos que se me presentaron fueran obstáculos en alcanzar mis objetivos, me han enseñado a enfrentar siempre las dificultades sin perder nunca la determinación ni rendirme en el intento.

A mis padres, pilar fundamental de mi vida, y a mi familia, por su amor incondicional y apoyo constante, les dedico este logro. Sus sacrificios y aliento fueron la luz que iluminó cada página de este trabajo. A todos ustedes, mi gratitud eterna.

A mi hermano y a mi querida abuelita por ser mi mayor ejemplo de esfuerzo y dedicación, demostrando que en la familia siempre podemos encontrar el apoyo y la inspiración necesarios para crecer y alcanzar nuestros sueños.

A todas las personas que de alguna manera han creído en mí, les dedico este trabajo como un testimonio de gratitud y reconocimiento por su invaluable respaldo. Así mismo a mis amigos y docentes que me han brindado su confianza y orientación.

“Todo a su tiempo” Ten paciencia,
a veces dolerá el proceso,
pero Dios sabe lo que es mejor para ti
y su tiempo es perfecto.

AGRADECIMIENTO

A mis padres

Agradezco de todo corazón a mis padres, por inculcarme grandes valores, por su amor incondicional y el gran sacrificio que hicieron durante todo este tiempo, han sido el motor detrás de cada uno de mis logros.

A mi tutor, Galo Puetate

A mi distinguido tutor, por su orientación experta, paciencia y dedicación incansable a lo largo de este proceso. Sus consejos y conocimientos han sido fundamentales para el desarrollo de este trabajo. Siempre estaré agradecido con usted.

A mis compañeros

Quiero expresar mi más sincero agradecimiento a mis compañeros y amigos por estar siempre a mi lado durante este camino. Su apoyo, comprensión y aliento fueron fundamentales para alcanzar este logro. Cada día compartido y cada momento de colaboración han sido muy significativos para mí.

A mi familia

Por su apoyo inquebrantable y por celebrar cada uno de mis avances con entusiasmo y orgullo, les debo parte de este éxito. Cada palabra de aliento, cada gesto de ánimo, ha sido un impulso que ha fortalecido mi determinación y perseverancia.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El análisis forense de la aplicación Ubidesk de Uboratech se realizó con el objetivo de identificar y mitigar vulnerabilidades de seguridad, asegurar la integridad del código y los procesos, y garantizar la confidencialidad y disponibilidad de los datos. La metodología empleada incluyó varias fases: recolección de información, análisis estático y dinámico de código, revisión de registros, pruebas de penetración, y evaluación de políticas y procedimientos de seguridad. En la fase de recolección de información, se obtuvo el código fuente, documentación técnica, registros de actividad y bases de datos. El análisis estático reveló vulnerabilidades como inyecciones de código y malas prácticas de programación. En el análisis dinámico, se monitoreó el comportamiento de la aplicación en un entorno controlado, identificando patrones de uso anómalos y tráfico de red sospechoso, finalmente, la evaluación de políticas y procedimientos de seguridad reveló áreas de mejora en las prácticas de Uboratech. Los resultados mostraron varias vulnerabilidades críticas en el código fuente, errores de implementación que podrían causar fallos y anomalías en los registros que sugieren intentos de acceso no autorizado. Se propusieron recomendaciones como la implementación de validaciones de entrada más estrictas, el uso de cifrado fuerte y la revisión periódica de los registros. En conclusión, el análisis forense permitió identificar y mitigar vulnerabilidades críticas y debilidades en la arquitectura mejorando significativamente la seguridad y eficiencia de la aplicación. Por lo tanto, Uboratech debe implementar las recomendaciones y continuar revisando periódicamente la seguridad de Ubidesk, siguiendo las mejores prácticas de la industria para mantener la integridad y confiabilidad de la aplicación y proteger los datos de los usuarios contra posibles amenazas.

Palabras clave: Análisis forense de código, vulnerabilidades de seguridad, análisis estático de código, mitigación de vulnerabilidades, Ubidesk

ABSTRACT

The forensic analysis of Uboratech's Ubidesk application was conducted with the aim of identifying and mitigating security vulnerabilities, ensuring the integrity of the code and processes, and guaranteeing the confidentiality and availability of data. The methodology employed included several phases: information gathering, static and dynamic code analysis, log review, penetration testing, and evaluation of security policies and procedures. In the information gathering phase, the source code, technical documentation, activity logs, and databases were obtained. The static analysis revealed vulnerabilities such as code injections and poor programming practices. In the dynamic analysis, the application's behavior was monitored in a controlled environment, identifying anomalous usage patterns and suspicious network traffic. Finally, the evaluation of security policies and procedures revealed areas for improvement in Uboratech's practices. The results showed several critical vulnerabilities in the source code, implementation errors that could cause failures, and anomalies in the logs suggesting unauthorized access attempts. Recommendations included the implementation of stricter input validations, the use of strong encryption, and periodic log reviews. In conclusion, the forensic analysis allowed for the identification and mitigation of critical vulnerabilities and weaknesses in the architecture, significantly improving the security and efficiency of the application. Therefore, Uboratech should implement the recommendations and continue to periodically review the security of Ubidesk, following industry best practices to maintain the integrity and reliability of the application and protect user data against potential threats.

Keywords: forensic code analysis, security vulnerabilities, static code analysis, vulnerability mitigation, Ubidesk

INTRODUCCIÓN

La seguridad de la información en la actualidad es un factor clave para las organizaciones, más aún a partir de la pandemia COVID-19, que impulsó la inserción de sistemas y aplicaciones informáticas en la administración, gestión y control de las operaciones de los distintos modelos de negocio, esto generó un aumento y dependencia de la tecnología y los datos que se generan producto de esta trasformación organizacional que utilizan diariamente en sus operaciones. Por lo tanto, cada vez es necesario garantizar la disponibilidad, la integridad y confidencialidad de esta información. (Cedeño Mosquera, 2023)

Bajo este contexto, la empresa UboraTech de la ciudad de Ibarra, es una institución que provee y genera servicios tecnológicos a los clientes a través de un sistema centralizado sobre los datos e información de los distintos servicios y clientes, que se administran desde una única plataforma centralizada denominada Ubidesk, que es una aplicación de software esencial para la continuidad del negocio, ya que controla, administra y gestiona el pool de clientes y demás datos e información relacionada con métodos de pago, facturación, direcciones, teléfonos, entre otros datos sensibles para los clientes.

Ubidesk, es una aplicación que se encuentra alojada en un hosting que se accede a través de la web, lo que la hace vulnerable a cualquier amenaza de seguridad. De ahí la necesidad de analizar y mejorar el esquema de seguridad de la aplicación, sobre todo en el ámbito de ciberseguridad, políticas y medidas que se deben tomar para proteger el tráfico de red y sobremanera los datos sensibles que se manejan con el fin de proteger y salvaguardar los datos e información de los clientes, garantizando de esta forma la confianza e integridad de las operaciones y de los servicios de los clientes.

Del análisis en el ámbito de la seguridad y del desarrollo de la aplicación, se ha detectado que la aplicación Ubidesk no cumple con estándares de seguridad necesarios para proteger los datos que se ejecutan en la estructura de la lógica de negocio (código fuente), transacciones de base de datos, así como del modelo de desarrollo de la aplicación, que es información sensible de los distintos usuarios y la empresa. Al no tener el control de la seguridad, repercute en graves consecuencias,

como la pérdida de información crítica (datos de transacciones de pago), la violación de la privacidad de los usuarios y la reputación de la empresa. Además, la falta de un adecuado sistema de monitoreo y respuesta a incidentes de seguridad dificulta la detección y resolución rápida de los problemas, fallos y riesgos de la seguridad.

El presente trabajo aborda la problemática de seguridad del software mediante un análisis forense detallado de código y procesos en la aplicación UbiDesk. Se busca identificar vulnerabilidades y debilidades tanto en el código fuente como en los procesos inherentes al desarrollo y mantenimiento de la aplicación. Se consideran los riesgos y vulnerabilidades de seguridad conforme a las buenas prácticas y estándares de ciberseguridad de la industria del desarrollo de software, abordando aspectos como privilegios de acceso, autenticación y autorización de usuarios, criptografía de datos, e implementación de protocolos de comunicación seguros.

Objetivos:

Objetivo General: Analizar el código fuente y procesos de la aplicación Ubidesk utilizando la herramienta SonarQube para la empresa UBORATECH.

Objetivos Específicos:

1. Realizar un análisis exhaustivo de la literatura técnica y académica para identificar y comprender los diferentes tipos de vulnerabilidades, fallos y errores presentes en aplicaciones de software.
2. Identificar las vulnerabilidades de seguridad específicas de la aplicación Ubidesk mediante un análisis minucioso del código, enfocándose en la detección de vulnerabilidades como inyecciones de código, errores de autenticación o configuraciones inseguras.
3. Proponer e implementar un conjunto de buenas prácticas de seguridad en el diseño, codificación y gestión de actualizaciones para la aplicación Ubidesk, con el objetivo de mitigar las vulnerabilidades identificadas y fortalecer la seguridad del sistema.

Este estudio se realizará en colaboración con UBORATECH, con el propósito de mejorar la seguridad y confiabilidad de la aplicación Ubidesk, asegurando así la protección de los datos y la integridad del sistema para sus usuarios.

CAPÍTULO I
ESTADO DEL ARTE

1.1.Análisis forense de código y procesos de la aplicación

El análisis forense digital es una disciplina crucial en el mundo actual, se enfoca en la investigación y recolección de evidencias digitales en el contexto de distintos delitos, incluyendo los informáticos. Los expertos en informática forense utilizan técnicas y herramientas especializadas para examinar las diversas soluciones tecnológicas en contextos de la seguridad, sobremanera en el ámbito de código fuente, y proceso de desarrollo de productos de software, con el objetivo de analizar los datos almacenados en ellos, las transacciones, operaciones y el robo de datos.

Un forense informático, también conocido como experto en informática, es un profesional especializado en el campo de la seguridad informática y la investigación de delitos cibernéticos. Su principal objetivo es recopilar, analizar y presentar pruebas digitales para su uso en investigaciones legales. El papel del forense informático es crucial en un mundo cada vez más digitalizado, donde los delitos informáticos y las violaciones de la seguridad en línea están en aumento (Cloudflare, 2023).

El proceso de análisis forense informático se lleva a cabo de manera rigurosa y siguiendo procedimientos específicos para garantizar la integridad de la evidencia y su utilidad en un entorno legal. Comienza con la identificación y recopilación de evidencia digital relevante. Los expertos utilizan dispositivos de análisis forense especializados y técnicas de recuperación de datos para asegurar la integridad de la información recolectada. Una vez que se han recopilado las evidencias, se procede al análisis en sí (Gasco, 2023).

"La preservación de la confidencialidad, integridad y disponibilidad de la información mediante la aplicación de un enfoque de gestión de riesgos y la implementación de controles adecuados" (Unad, 2020, p. 8).

La seguridad informática abarca la protección de información, ya sea de carácter personal, organizacional o gubernamental, y se extiende a todos los dispositivos de uso común, como computadoras portátiles, computadoras de escritorio, teléfonos móviles, entre otros. Su objetivo principal es identificar y mitigar las amenazas que puedan comprometer la integridad de la información almacenada y procesada por estos dispositivos. (Suárez, 2021).

Dentro de cualquier organización, es posible que ocurran eventos que representen una amenaza para los activos de la empresa. Estos eventos se conocen como amenazas, y la probabilidad de que se materialicen se denomina vulnerabilidad, dichas eventualidades pueden generar impactos de diversos niveles dentro de la organización. Para (Palacios, 2020), el término utilizado para describir el tipo de impacto que estas vulnerabilidades pueden ocasionar en la organización es riesgo. El objetivo de la seguridad informática es minimizar estos riesgos, de modo que las vulnerabilidades existentes no puedan ser explotadas.

La seguridad informática abarca todas las medidas implementadas para prevenir la ejecución de operaciones no autorizadas en sistemas informáticos, las cuales podrían comprometer su integridad, autenticidad y confidencialidad. El propósito fundamental de estas medidas es minimizar riesgos y garantizar el uso apropiado de los recursos del sistema. (Mena, 2021).

La seguridad informática se enfoca en proteger información de diversos tipos, que puede residir no solo en la red, sino también en dispositivos tecnológicos, el objetivo es salvaguardar esta información contra cualquier amenaza que pueda comprometer su integridad. Además, destaca que una seguridad efectiva no solo se centra en la prevención de ataques, sino también en la detección y corrección de los mismos, de ahí la necesidad de que los desarrolladores de aplicaciones tecnológicas adopten un enfoque integral que abarque la implementación de medidas preventivas y buenas prácticas de seguridad, para dar respuesta frente a incidentes para mitigar cualquier daño potencial a las aplicaciones tecnológicas. (Suárez, 2021).

La seguridad informática es fundamental en la protección de la información, especialmente en entornos de producción, ya que se debe garantizar la confidencialidad, integridad y disponibilidad de los datos, lo cual es esencial para mantener la privacidad y seguridad de los usuarios, clientes, proveedores, entre otros. (Moya, 2023).

En el contexto de instituciones proveedoras de servicios de tecnología, como es el caso de UboraTech, es crucial destacar la importancia de manejar buenas prácticas y aplicar normas internacionales para controlar el manejo de la información. Más aún donde la dependencia de las herramientas digitales es cada vez mayor, es imperativo establecer políticas y procedimientos claros para proteger los datos sensibles.

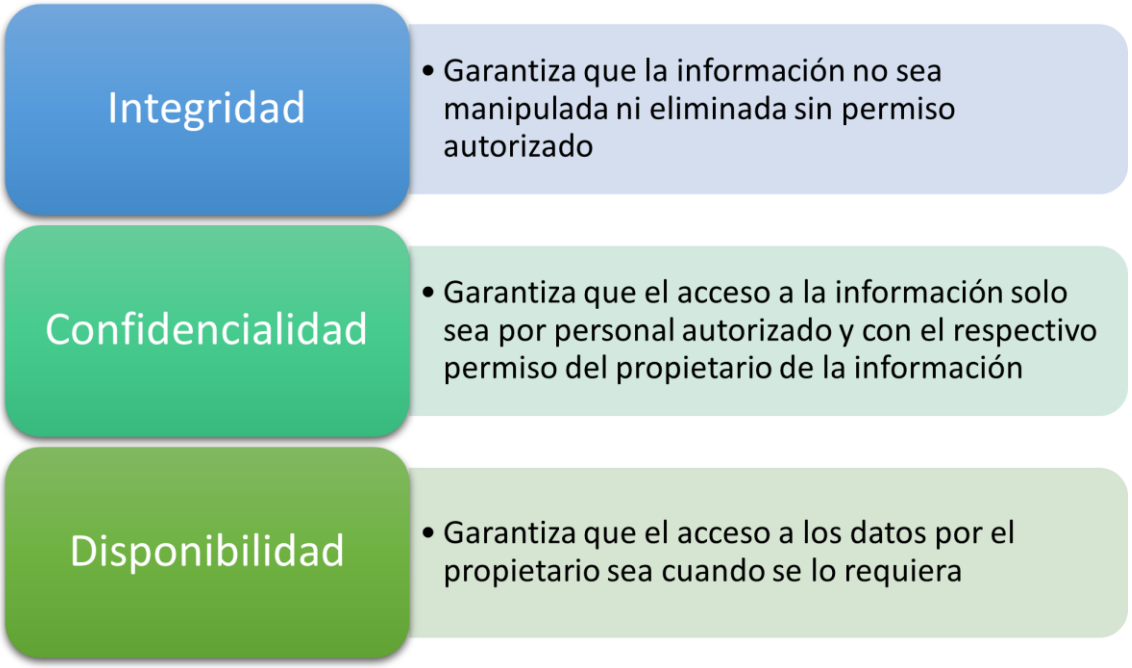
Esto implica no solo implementar medidas técnicas, como firewalls y cifrado de datos, sino también educar a los usuarios sobre la importancia de utilizar contraseñas seguras, evitar compartir información confidencial y estar alerta ante posibles amenazas cibernéticas, como el phishing. Además, es fundamental adherirse a normativas y estándares internacionales, como la Ley de Protección de Datos Personales, para garantizar el cumplimiento de requisitos legales y éticos en el manejo de la información.

Existen tres conceptos principales que abarcan la seguridad informática que son: la integridad, la confidencialidad y la disponibilidad. A estos se los ha denominado como la Tríada de la seguridad de la información, enfocándose principalmente en la protección de los datos. Estas propiedades son las que ayudan a mantener un sistema robusto, minimizando el riesgo de sufrir eventos que puedan comprometer la seguridad de la información (Tejada, 2023).

1.1.1. Fundamentación de la seguridad de la información

La seguridad informática se basa en tres conceptos principales conocidos como la tríada de la seguridad de la información: integridad, confidencialidad y disponibilidad. Estos aspectos se centran en proteger los datos y son fundamentales para mantener un sistema robusto, reduciendo el riesgo de sufrir eventos que puedan comprometer la seguridad de la información. La integridad garantiza que los datos no sean modificados de manera no autorizada, la confidencialidad asegura que la información solo sea accesible para quienes estén autorizados, y la disponibilidad se refiere a que los datos estén disponibles cuando se necesiten. Estos principios son fundamentales en la protección de la información en el entorno digital actual. La seguridad informática es un aspecto crítico en el mundo digital actual, donde la protección de la información sensible es fundamental para garantizar la confianza y la integridad de los sistemas y datos. (Tejada, 2023). En la figura 1 se detalla el concepto de la tríada de la seguridad de la información.

Figura 1
Tríada en la que se fundamenta la seguridad de la información



Fuente: Adaptado de Gestión de incidentes de seguridad informática (Tejada, 2023)

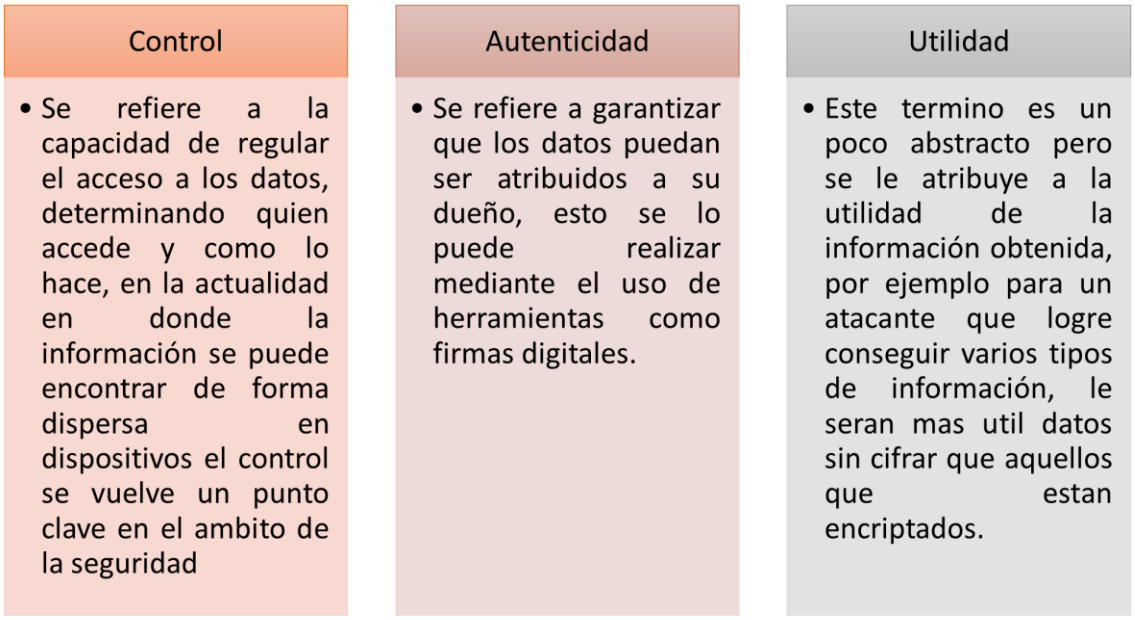
Es esencial para las organizaciones adoptar un enfoque integral de seguridad informática, que incluya medidas preventivas, detección de amenazas y respuesta a incidentes, así como el cumplimiento de normativas y estándares internacionales. La implementación de buenas prácticas de seguridad y la educación de los usuarios son clave para mitigar riesgos y mantener un entorno digital seguro y confiable.

Por lo tanto, la seguridad informática es un aspecto crítico en la protección de la información en el mundo digital actual, más aún para UboraTech, que centraliza los datos e información a través de una única aplicación web, y que no está exenta de sufrir ataques o vulneraciones a los principios de confidencialidad, integridad y disponibilidad de datos y sistemas.

En la actualidad, se ha generado un amplio debate sobre si los conceptos tradicionales de seguridad informática son suficientes para abordar todas las consideraciones necesarias, especialmente debido a la rápida evolución tecnológica y los cambios en la gestión de la información. Para algunos expertos, la Tríada de la seguridad de la información, compuesta por los principios de integridad, confidencialidad y disponibilidad, podría quedarse corta en ciertos aspectos.

Como consecuencia, han surgido nuevos enfoques y conceptos que buscan complementar la Tríada. Estos conceptos adicionales incluyen: (Vega, 2021). En la figura 2 se detallan las definiciones de los nuevos conceptos.

Figura 2
Nuevos enfoques de la seguridad de la información



Fuente: Adaptado de Gestión de incidentes de seguridad informática (Tejada, 2023)

- **Control:** Se refiere a la capacidad de regular y gestionar el acceso a la información y los recursos tecnológicos. Esto implica establecer políticas, procedimientos y medidas de seguridad para garantizar que solo los usuarios autorizados puedan acceder a la información y realizar acciones específicas.
- **Autenticidad:** Este concepto se enfoca en verificar la identidad de los usuarios y la procedencia de la información. Es esencial asegurar que tanto los usuarios como los sistemas sean auténticos, para evitar la manipulación de la información o el acceso no autorizado a los sistemas.

- **Utilidad:** La seguridad informática no solo se trata de proteger la información, sino también de garantizar que esta sea útil y esté disponible cuando se necesite. Es importante

encontrar un equilibrio entre la seguridad y la accesibilidad de la información, de modo que pueda cumplir su propósito sin comprometer su integridad o confidencialidad.

Estos nuevos conceptos complementarios a la Tríada de la seguridad de la información reflejan la necesidad de una perspectiva más amplia y completa de la seguridad informática, que abarque aspectos como el control, la autenticidad y la utilidad, además de los principios tradicionales de integridad, confidencialidad y disponibilidad.

1.1.2. Incidencias de seguridad de información

En la era digital actual, la seguridad informática se ha convertido en un tema de importancia debido al crecimiento exponencial de las amenazas cibernéticas, donde los incidentes de seguridad, que abarcan desde brechas de datos hasta ataques de ransomware, están en constante aumento y representan una amenaza significativa para individuos, empresas y gobiernos en todo el mundo. Estos incidentes no solo pueden resultar en la pérdida de datos confidenciales, sino que también pueden tener un impacto devastador en la reputación, la operatividad y la estabilidad financiera de las organizaciones afectadas (González, 2021).

Frente a estas crecientes amenazas es labor de los profesionales de la seguridad de la información identificar la naturaleza de los incidentes de seguridad, sus causas subyacentes y las consecuencias devastadoras que pueden tener para las víctimas y la sociedad en su conjunto. Además, comprender la importancia de implementar medidas proactivas de seguridad cibernética para mitigar el riesgo de sufrir tales incidentes y proteger activamente los activos digitales.

Por lo tanto, se debe entender y comprender que “un incidente de seguridad representa evento no contemplado que afecta directamente a la seguridad informática (integridad, confidencialidad y disponibilidad) comprometiendo el funcionamiento normal de operaciones o servicios de las aplicaciones informáticas que pone en peligro la información” (Machín, 2021, pág. 8).

Según Machín (2021), se clasifican diferentes tipos de incidentes de seguridad a los que están expuestas las aplicaciones informáticas debido a una mala gestión de seguridad que se resumen en la tabla.

Tabla 1
Tipos de incidencias de seguridad de la información

Tipo de Incidente	Descripción	Ejemplos
Accesos no autorizados	Se produce cuando alguien obtiene acceso ilegal a información confidencial o datos privados de una organización con fines delictivos.	- Intento de robo de contraseñas. - Acceso no autorizado a una base de datos protegida.
Robo de contraseñas	Implica la sustracción no autorizada de contraseñas, lo que puede comprometer la seguridad de cuentas y sistemas.	- Obtención ilegal de contraseñas de usuarios. - Uso indebido de credenciales de acceso.
Robo de información	La extracción o copia no autorizada de datos sensibles o confidenciales.	- Copia de archivos confidenciales en una unidad USB sin permiso. - Descarga no autorizada de datos de clientes.
Abuso o mal uso de servicios informáticos	Se refiere a la utilización indebida de recursos internos o externos de la organización.	- Uso excesivo de ancho de banda para actividades personales. - Acceso no autorizado a servicios en la nube de la empresa.
Alteración de la información	Implica cambios no autorizados en datos o registros.	- Modificación de registros financieros para ocultar transacciones fraudulentas. - Cambio de datos en una base de datos de inventario.

Fuente: Adaptado de incidencias de seguridad de la información (García, 2022).

La gestión efectiva de incidentes de seguridad de la información requiere una combinación de medidas técnicas, políticas claras, formación del personal y una respuesta rápida ante cualquier amenaza. La colaboración entre equipos de seguridad, TI y dirección es fundamental para proteger los activos y garantizar la continuidad del negocio.

1.1.3. Incidencias de seguridad de aplicaciones web

Las incidencias de seguridad en aplicaciones web son eventos que comprometen la integridad, confidencialidad o disponibilidad de los datos alojados en una aplicación web. Estas incidencias pueden ser causadas por una variedad de factores, que van desde vulnerabilidades en el código de la aplicación hasta errores en la configuración del servidor web.

Las causas de las incidencias de seguridad en aplicaciones web pueden ser diversas. Esto incluye vulnerabilidades conocidas como inyección de SQL, cross-site scripting (XSS), ataques de

denegación de servicio (DoS), falta de control de acceso, entre otros. Estas vulnerabilidades pueden ser explotadas por atacantes malintencionados para robar datos sensibles, comprometer la funcionalidad de la aplicación o interrumpir su funcionamiento normal (Brito, 2021). Ver tabla 2.

Tabla 2
Incidencias de seguridad de aplicaciones web

Número	Vulnerabilidad	Descripción
1	Fallos de control de acceso	Manipulación del valor de identificadores y asignación de mínimos privilegios indispensables.
2	Fallos criptográficos	Uso de algoritmos de cifrado fuertes y plenamente actualizados.
3	Inyección	Vinculación segura de los parámetros de entrada para evitar la inyección de código.
4	Diseño inseguro	La importancia de los modelos de ciclo de vida de desarrollo seguro.
5	Configuración de seguridad incorrecta	Hardening, segmentación y seguir buenas prácticas.
6	Componentes vulnerables y obsoletos	Securizar todas las fases del ciclo de vida.
7	Fallos de identificación y autentificación	Autenticación multifactor y otras medidas de seguridad.
8	Fallos en el software y en la integridad de los datos	Proteger las supply chains y ejecutar labores de comprobación.
9	Fallos en el registro y la supervisión de la seguridad	Generación de logs y sus copias de seguridad.
10	Falsificación de solicitudes del lado del servidor	Actuar sobre la capa de red y la de aplicación.

Además de las vulnerabilidades técnicas, las incidencias de seguridad en aplicaciones web también pueden ser el resultado de prácticas de desarrollo inseguras, como la falta de validación de datos, la omisión de protecciones contra ataques comunes o la no aplicación de actualizaciones de seguridad. Estos problemas pueden permitir que los atacantes encuentren y exploten vulnerabilidades en la aplicación. Las incidencias de seguridad en aplicaciones web son un problema grave que puede tener consecuencias devastadoras para las organizaciones y los usuarios finales.

La pérdida de datos confidenciales, la interrupción del servicio y el daño a la reputación pueden tener un impacto significativo en la viabilidad y la confianza en una aplicación web. Para mitigar el riesgo de incidencias de seguridad en aplicaciones web, es fundamental implementar buenas prácticas de seguridad durante todo el ciclo de vida del desarrollo de software. Esto incluye la realización de pruebas de seguridad regulares, la aplicación de parches de seguridad de forma proactiva, la capacitación del personal en concienciación sobre seguridad y la adopción de marcos de seguridad como OWASP Top 10 para identificar y abordar las vulnerabilidades comunes.

1.1.4. Técnicas de evaluación de la seguridad de aplicaciones web

La evaluación de la seguridad en sistemas es un proceso crítico que deben realizar los responsables de la infraestructura de tecnologías dentro de cualquier organización para garantizar que los sistemas informáticos, redes y demás activos informáticos estén debidamente protegidos contra los diferentes tipos de amenazas y vulnerabilidades.

Existen diferentes técnicas que se aplican para la evaluación de seguridad en sistemas, si bien cada técnica tiene varios métodos para su implementación, todas tienen como objetivo encontrar o minimizar las vulnerabilidades presentes en los sistemas (SonicWall-Inc., 2024).

Tabla 3
Técnicas de evaluación de seguridad de sistemas informáticos

Aspecto	Descripción	Importancia
Evaluación de Vulnerabilidad	Analiza y estima los riesgos de debilidades en la seguridad de los sistemas de información e informática. Identifica, prioriza y mitiga vulnerabilidades.	Esencial para prevenir ataques y minimizar amenazas a la seguridad. Debe realizarse regularmente.
Auditoría de Seguridad	Revisa políticas, procedimientos y configuraciones para garantizar el cumplimiento de las mejores prácticas de seguridad.	Ayuda a verificar el estado actual de la seguridad y a identificar áreas de mejora.
Pruebas de Penetración (Pen Testing)	Simula ataques reales para identificar vulnerabilidades.	Proporciona una visión realista de la resistencia del sistema ante ataques.
Análisis de Código Fuente	Examina el código de aplicaciones para detectar problemas de seguridad.	Ayuda a identificar vulnerabilidades en el desarrollo de software.
Evaluación de Arquitectura	Evalúa la seguridad desde una perspectiva de diseño y arquitectura.	Permite identificar riesgos en la estructura del sistema.

Fuente: Tomado de vulneraciones de aplicaciones web (SonicWall-Inc., 2024).

Las técnicas de evaluación de seguridad de sistemas, ver Tabla 2, son herramientas fundamentales en la protección de la información y la mitigación de riesgos en el entorno digital. Estas técnicas abarcan una variedad de enfoques, desde auditorías de seguridad hasta pruebas de penetración y análisis de vulnerabilidades.

1.1.5. Herramientas de evaluación de la seguridad de aplicaciones

Las herramientas de evaluación de seguridad de sistemas son componentes fundamentales para los profesionales de seguridad informática, debido a que estas herramientas están diseñadas para identificar vulnerabilidades, analizar riesgos y fortalecer la seguridad de los sistemas y redes. Si bien estas herramientas son valiosas en la evaluación de seguridad de sistemas, es importante tener en cuenta que ninguna herramienta es completamente infalible por sí sola. Además, la efectividad de estas herramientas depende en gran medida de la habilidad y experiencia del usuario en su aplicación (Pineda, 2021).

Tabla 4
Herramientas de evaluación de seguridad

Herramienta	Descripción
Burp Suite	Una suite de herramientas de prueba de seguridad diseñada específicamente para pruebas de seguridad de aplicaciones web.
OWASP ZAP	Una herramienta de código abierto para encontrar vulnerabilidades de seguridad en aplicaciones web durante el desarrollo y las pruebas.
Acunetix	Una herramienta automatizada de escaneo de vulnerabilidades diseñada para identificar y gestionar vulnerabilidades en aplicaciones web.
SonarQube	Una plataforma de análisis estático y dinámico de seguridad que ayuda a identificar y corregir vulnerabilidades de seguridad en aplicaciones durante el desarrollo.

Fuente: Tomado de (Pineda, 2021)

Por otro lado, es importante reconocer que el paisaje de amenazas está en constante evolución, lo que requiere que las herramientas de seguridad también evolucionen para hacer frente a nuevas y emergentes amenazas y a las necesidades cambiantes de los profesionales de seguridad para garantizar que las herramientas sigan siendo efectivas en la detección y mitigación de amenazas. (Pineda, 2021)

CAPÍTULO II
MARCO METODOLÓGICO

2.1.Generalidades de la investigación

El análisis forense de código y procesos es una técnica fundamental en la ciberseguridad que implica el examen detallado de los sistemas de software para detectar vulnerabilidades, errores y comportamientos maliciosos a los que puede estar sujeto, por lo tanto, este tipo de análisis es relevante en el contexto donde la integridad y seguridad de la información son aspectos prioritarios.

En este escenario, la empresa UboraTech, reconocida por brindar una serie de soluciones en el campo de la tecnología de la información, por lo que ha debido analizar la aplicación Ubidesk mediante un análisis forense de código y procesos. El análisis de código es una práctica crítica dentro del desarrollo de software de la aplicación Ubidesk que tiene por objetivo garantizar la calidad, seguridad y eficiencia de las aplicaciones informáticas. A medida que la tecnología avanza y se integra cada vez más en todos los aspectos de la vida cotidiana, la dependencia de sistemas de software fiables y seguros se ha vuelto fundamental para el desarrollo de la gestión del modelo de negocio de UboraTech.

En este contexto, el análisis de código de la aplicación Ubidesk emerge como un componente esencial en el ciclo de vida del desarrollo de software, ofreciendo múltiples beneficios que van desde la detección temprana de errores hasta la prevención de vulnerabilidades de seguridad. En un entorno digital donde los ataques cibernéticos son cada vez más sofisticados y dañinos, la capacidad de inspeccionar y mejorar el código de una aplicación no es solo una medida de precaución, sino una necesidad imperativa.

El análisis de código permite a los desarrolladores identificar problemas potenciales antes de que el software sea desplegado, reduciendo así el riesgo de fallos críticos y exposición a ciberataques que pueden comprometer datos sensibles de usuarios y empresas.

Además, en el competitivo mercado actual de UboraTech la calidad del software es un diferenciador clave que no solo garantiza la confiabilidad de las transacciones y datos, sino que permite salvaguardar datos sensibles cumpliendo con las regulaciones estatales, donde el software es confiable, eficiente y seguro no solo que satisface las necesidades de los usuarios, sino que también refuerza la reputación de la empresa como un proveedor confiable de tecnología. Por lo tanto, el análisis de código no solo es fundamental para la seguridad y funcionalidad de la aplicación Ubidesk, sino que también es crucial para mantener y mejorar la posición de la empresa en el mercado.

2.1.1. Enfoque y tipo de investigación

Para abordar el análisis de código, especialmente en contextos donde se evalúa el software Ubidesk con el objetivo de detectar vulnerabilidades y errores, el enfoque de investigación más adecuado suele ser el enfoque cuantitativo, aunque elementos del enfoque cualitativo también son relevantes partiendo de los objetivos específicos del estudio.

Se parte de la medición y análisis estadístico de los datos numéricos relacionados con el código, donde se determinó el tipo de errores específicos, la distribución de tipos de vulnerabilidades y la evaluación de la eficacia de diferentes herramientas de análisis estático de código.

Este enfoque de investigación resultó ser el más adecuado, ya que facilitó una evaluación exhaustiva de la seguridad en la infraestructura tecnológica en el entorno de pruebas de la aplicación Ubidesk. Se implementó un proceso secuencial que se basó en las prácticas recomendadas y la metodología SonarQube.

Dicho proceso comenzó con un análisis detallado del estado actual de la infraestructura, lo cual permitió detectar vulnerabilidades y desarrollar soluciones adecuadas. Estas soluciones fueron posteriormente implementadas y evaluadas nuevamente, permitiendo así obtener resultados concretos y medibles.

2.1.2. Procedimiento de investigación

El proceso de investigación se inició con la definición del entorno de pruebas de la aplicación Ubidesk. Este proceso fue crucial para establecer las condiciones en las cuales se llevó a cabo la prueba de análisis de código. Este proceso involucró la implementación de las estrategias de parametrización con respecto a la metodología de SonarQube. Algunas de las principales estrategias son: líneas de código, issues con sus tipos y severidades, reglas aplicadas, cobertura, porcentaje de código duplicado, líneas/bloques duplicados, complejidad, deuda técnica, rating SQALE y el quality gate, para evaluar la seguridad y funcionalidad de UbiDesk.

Esta metodología iterativa de análisis, medición, informes y corrección ayuda a los equipos de desarrollo a mantener un código de alta calidad, mejorar continuamente la base de código y prevenir la introducción de nuevos problemas a medida que el proyecto evoluciona, que se detalla a continuación.

Figura 3
Procesos de análisis de código definido por Sonarqube

Proceso A: Configuración inicial.

1. **Instalación y configuración de SonarQube**. Con fines del desarrollo de la tesis, se lo realizó en el entorno de pruebas de la empresa UboraTech, sobre el servidor donde está desplegado la aplicación Ubidesk. Se configuró la base de datos asegurando la comunicación entre los componentes de SonarQube para su ejecución.
    - # On Windows, se debe ejecutar
    - `C:\sonarqube\bin\windows-x86-xx\StartSonar.bat`
    - `“C:\sonar-runner” o “/etc/sonar-runner”)`

2. **Integración con el Sistema de control de versiones**: Donde se configuró SonarQube con los sistemas de control de versiones de software de UboraTech para de esta forma analizar automáticamente el código cada vez que se realiza una nueva versión del software Ubidesk.

    - `C:\sonar-"reléase UbiDesk V1.002"`
    - `“/etc/sonar-e "UbiDesk V1.002"`

Proceso B: Definición de reglas y perfil de calidad.

Personalización de reglas de seguridad:

1. **Reglas de seguridad:** Donde se configuró los parámetros para evaluar según las necesidades específicas del proyecto, tomando en cuenta tipos de errores, impacto, nivel de vulnerabilidades y prácticas de desarrollo en el código fuente.
    - `Code smell (maintainability domain)`
    - `Bug (reliability domain)`
    - `Vulnerability (security domain)`
    - `Security hotspot (security domain)`
    - `User Add/Remove tags:`
    - `"SQL injection".`
    - `"Cross-Site Scripting (XSS) vulnerability".`
    - `"Unsafe handling of sensitive data".`
    - `"Avoid unnecessarily complex loops"`
    - `"Avoid exception leakage"`
    - `"Excessive method length"`

2. **Perfiles de calidad**: Donde se configuró el perfil de administrador de calidad (QA) que permite especificar el conjunto de reglas que deben pasar las revisiones de código para ser consideradas aceptables.
    - "Quality Profiles"
    - `perfil "QA_Profile" = admnin`
    - `Ruler= "Variable naming conventions"= admnin`

Proceso C: Análisis de código

1. **Análisis continuo:** El análisis del código fuente de aplicación UbiDesk en SonarQube se lo realizó mediante el método automático *(desencadenado)* con base a las reglas especificadas del paso B.
    - `inicio de sesión: adminUbiDesck`
    - `contraseña:**********`

    - `cd C:\sonar- reléase UbiDesk V1.002\projects\languages\php\sonar-runner\php-sonar-runner-simple`
    - `C:\sonar-runner\bin\sonar-runner.bat`
2. **Revisión automática:** Al estar la aplicación desplegada en el entorno de pruebas, se configuró SonarQube para que ejecute análisis automáticos de forma desencadenada cada vez que hay cambios de código al repositorio, proporcionando feedback inmediato a los desarrolladores de UbiDesk.

    - `Configurar tu proyecto: "sonar- UbiDesk V1.002.properties"`
    - `sonar-scanner run`
    - `desc="Analizando código", unit=" UbiDesk V1.002"): time.sleep( time)`

`Analizando código", unit=" UbiDesk V1.002"): time.sleep( time)`

Proceso D: Revisión de resultados

1. **Evaluación de vulnerabilidades y bugs:** Donde se revisaron los resultados generados por SonarQube en el modo de ejecución desencadenado, donde se identificaron las vulnerabilidades críticas, bugs y malas prácticas de programación.
2. **Priorización y gestión de problemas**: Con los resultados obtenidos se realizó la priorización de los problemas basados en su severidad y el riesgo que representan para la aplicación Ubidesk más aún de las instrucciones de código que representan problemas críticos.

Tipo de alerta	Riesgo
`Inyección SQL`	Alto
`Inyección SQL – MySQL`	Alto
`Ausencia de fichas (tokens) Anti-CSRF`	Alto
`Cabecera Content Security Policy (CSP) no configurada`	Alto

`Desconfiguración de Dominio cruzado`	Alto
`Falta de cabecera Anti-Clickjacking`	Alto
`Filtrado de información en .htaccess`	Alto
`Hidden File Found (Archivo Oculto Encontrado)`	Alto
`Librería JS Vulnerable`	Alto
Cookie No HttpOnly Flag	Medio
Cookie Without Secure Flag	Medio
Cookie sin el atributo SameSite	Medio
Server Leaks Version Information vía "Server" HTTP Response Header Field	Bajo
Strict-Transport-Security Header Not Set	Bajo
X-Content-Type-Options Header Missing	Bajo
Authentication Request Identified	Informativo
Content-Type Header Missing	Informativo
Divulgación de información - Comentarios sospechosos	Informativo

Proceso E: Reporte y documentación

1. **Documentación de resultados:** Donde se documentaron los hallazgos con relación a los fallos y errores del código fuente, así como las medidas correctivas aplicadas para mantener un registro de la seguridad del proyecto Ubidesk.

2. **Reportes de cumplimiento y seguridad:** Donde se generó un reporte donde se resume el estado de la calidad y seguridad del código de la aplicación Ubidesk, que puede el Gerente Propietario de UboraTech considerarlo como un insumo para auditorías internas o externas.

Todos los procesos mencionados no solo ayudan a mantener un alto nivel de calidad en el código, sino que también aseguran que las aplicaciones desarrolladas sean seguras y estén libres de vulnerabilidades críticas.

2.1.3. Consideraciones éticas

La investigación se realizó siguiendo los principios éticos fundamentales sobre la obligación de no causar daño intencionalmente con relación a la lógica de negocio, configuraciones y modelo de servicios. Además, se contó con la autorización y supervisión técnica del programador de Ubidesk delegado de la empresa UboraTech, asegurando la confidencialidad y el anonimato en todos los procesos y actividades relacionados con la investigación.

CAPÍTULO III
RESULTADOS Y DISCUSIÓN

En este capítulo, se detalla el proceso de evaluación de seguridad aplicada a la plataforma Ubidesk mediante las herramientas de análisis de vulnerabilidades de código SonarQube. Donde se detallan los aspectos de la seguridad proporcionando una comprensión del estado actual de seguridad, además de que se han identificado áreas clave para mejorar. A continuación, se presenta una visión detallada sobre los aspectos de seguridad de software evaluados en Ubidesk.

3.1 Análisis de resultados

3.1.1. Ambiente de pruebas de la plataforma Ubidesk

Ubidesk es una plataforma de gestión del modelo de negocios de UboraTech que trabaja en línea facilitando la operación, proyectos, intercambio de documentos, servicios de tecnología, cobros y comunicación, lo que lo hace una herramienta clave para el desarrollo operacional de la empresa. El ambiente de desarrollo en el que esta implementado consta de diferentes tecnologías y herramientas que se detallan a continuación.

Tabla 5
Ambiente de desarrollo

Lenguaje de programación	Php 7.4
Framework	CodeIgniter 3.1.13
Base de datos	MariaDB 10.2
Patrón de diseño	MVC (modelo, vista, controlador)
Metodología de desarrollo	XP

El entorno de desarrollo de software Ubidesk está conformado por un conjunto de tecnologías y herramientas que permiten a los desarrolladores crear, probar y desplegar la aplicación de manera eficiente. El análisis de vulnerabilidades del código fuente se analizó utilizando la herramienta SonarQube y utilizando dos métodos diferentes.

3.1.2. Análisis de resultados de código desencadenado

El análisis de vulnerabilidades se lo realizó con la herramienta SonarQube aplicada al análisis de código fuente de la plataforma Ubidesk, donde se evaluó el software mediante una visión detallada de la calidad y seguridad del código, permitiendo identificar fallos y errores que se detallan a continuación.

Tabla 6
Método de configuración desencadenado de SonarQube

Herramienta	SonarQube
Método	Análisis de código desencadenado
Versión	10.2.1.78527

El análisis realizado con el método desencadenado se enfocó en las siguientes carpetas, abarcando varios aspectos críticos para el desarrollo de software de la aplicación Ubidesk proporcionando una visión integral de la seguridad del código. A continuación, se detallan los principales ámbitos de análisis que se evaluaron con la herramienta de análisis.

- Application/controllers
- Application/models
- Application/views
- Application/helpers

Tipo de Análisis:

- SonarPHP
- SonarHtml

El ámbito de análisis permitió detectar las vulnerabilidades según el tipo de vulnerabilidad, nivel de riesgo sobre la calidad del software.

3.1.2.1. Análisis con cuentas de riesgo de confianza

En el análisis de vulnerabilidades, se identificaron diversas alertas, clasificadas según su nivel de riesgo y confianza. La tabla 7, detalla el número de alertas para cada nivel de riesgo y confianza incluidos en el informe, el cual permitió caracterizar los riesgos y vulnerabilidades detectadas en el software Ubidesk.

Tabla 7
Análisis de vulnerabilidades Ubidesk

		Confianza				
		Confirmado por Usuario	**Alta**	**Medios de comunicación**	**Baja**	**Total**
Riesgos	**Alto**	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)	0 (0,0 %)
	Medio	0 (0,0 %)	1 (6,7 %)	3 (20,0 %)	1 (6,7 %)	5 (33,3 %)
	Bajo	0 (0,0 %)	1 (6,7 %)	5 (33,3 %)	0 (0,0 %)	6 (40,0 %)
	Informativo	0 (0,0 %)	1 (6,7 %)	2 (13,3 %)	1 (6,7 %)	4 (26,7 %)
	Total	0 (0,0 %)	3 (20,0 %)	10 (66,7 %)	2 (13,3 %)	15 (100 %)

3.1.2.2. Alertas por tipo de riesgos detectados

En el análisis de vulnerabilidades, se identificaron otros problemas de seguridad que, aunque no se clasifican como críticos o importantes, requieren atención para mejorar la seguridad general de la aplicación. La tabla 8 identifica los tipos de alerta identificados en el análisis del código de la aplicación Ubidesk, además se detalla el tipo de riesgo, la alerta y el porcentaje dentro de las líneas de código fuente en las que se encuentran.

Tabla 8
Tipos de alertas de seguridad detectadas

Tipo de alerta	Riesgo	Cantidad
Inyección SQL	Alto	2
Inyección SQL – MySQL	Alto	1
Ausencia de fichas (tokens) Anti-CSRF	Medio	4
Cabecera Content Security Policy (CSP) no configurada	Medio	12
Desconfiguración de Dominio cruzado	Medio	5
Falta de cabecera Anti-Clickjacking	Medio	6
Filtrado de información en .htaccess	Medio	1
Hidden File Found (Archivo Oculto Encontrado)	Medio	1
Librería JS Vulnerable	Medio	3
Cookie No HttpOnly Flag	Bajo	1
Cookie Without Secure Flag	Bajo	1
Cookie sin el atributo SameSite	Bajo	1
Cross-Domain JavaScript Source File Inclusion	Bajo	9
Divulgación de la marca de hora – Unix	Bajo	33
Server Leaks Version Information via "Server" HTTP Response Header Field	Bajo	53
Strict-Transport-Security Header Not Set	Bajo	75
X-Content-Type-Options Header Missing	Bajo	51
Authentication Request Identified	Informativo	2
Content-Type Header Missing	Informativo	2
Divulgación de información - Comentarios sospechosos	Informativo	60
GET para POST	Informativo	1
Modern Web Application	Informativo	2
Re-examine Cache-control Directives	Informativo	10
Retrieved from Cache	Informativo	8
Session Management Response Identified	Informativo	18
User Agent Fuzzer	Informativo	24
Total	26	

3.1.2.3. Evaluación del riesgo

El riesgo detectado mediante el análisis permite evaluar agrupando las vulnerabilidades según el tipo de riesgo que representan. Este enfoque permite una mejor comprensión y priorización de las medidas necesarias para mitigar cada tipo de amenaza identificada. La tabla 9, detalla el análisis de vulnerabilidades llevado a cabo, el cual ha revelado una serie de problemas de seguridad en la aplicación Ubidesk. A continuación, se presenta un resumen global de las vulnerabilidades

detectadas, agrupadas por el tipo de riesgo que representan y su correspondiente evaluación en términos de impacto, probabilidad y riesgo.

Tabla 9
Cálculo del riesgo por vulnerabilidad

Tipo de alerta	Factor de impacto	Calificación	Probabilidad	Calif	Riesgo
Inyección SQL	6.5	**Alto**	2.63	**Bajo**	**Medio**
Inyección SQL – MySQL					
Ausencia de fichas (tokens) Anti-CSRF	3.36	**Medio**	1.38	**Bajo**	**Medio**
Cabecera Content Security Policy (CSP) no configurada					
Desconfiguración de Dominio cruzado					
Falta de cabecera Anti-Clickjacking					
Filtrado de información en .htaccess					
Hidden File Found (Archivo Oculto Encontrado)					
Librería JS Vulnerable					
Cookie No HttpOnly Flag	2.75	**Bajo**	1.38	**Bajo**	Bajo
Cookie Without Secure Flag					
Cookie sin el atributo SameSite					
Cross-Domain JavaScript Source File Inclusion					
Divulgación de la marca de hora – Unix					
Server Leaks Version Information via "Server" HTTP Response Header Field					
Strict-Transport-Security Header Not Set					
X-Content-Type-Options Header Missing					
Authentication Request Identified	2.75	**Bajo**	0.88	**Bajo**	Bajo
Content-Type Header Missing					
Divulgación de información - Comentarios sospechosos					
GET para POST					
Modern Web Application					
Re-examine Cache-control Directives					
Retrieved from Cache					
Session Management Response Identified					
User Agent Fuzzer					

3.1.3. Análisis de resultados por método estático

El análisis de vulnerabilidades se realizó mediante el método estático sobre la lógica de programación de la aplicación Ubidesk, el cual proporcionó una visión detallada de la calidad y seguridad del código en el que se identificaron las áreas en las que el software tiene vulnerabilidades.

Tabla 10
Análisis de código por método estático

Herramienta	SonarQube
Método	Análisis de código estático
Versión	10.2.1.78527

El análisis se enfocó en archivos contenedores de la aplicación UbiDesk. El ámbito de análisis abarcó varios aspectos críticos para el desarrollo de software, proporcionando una visión integral de la seguridad del código que se detallan a continuación:

- Application/controllers
- Application/models
- Application/views
- Application/helpers

Tipo de análisis:

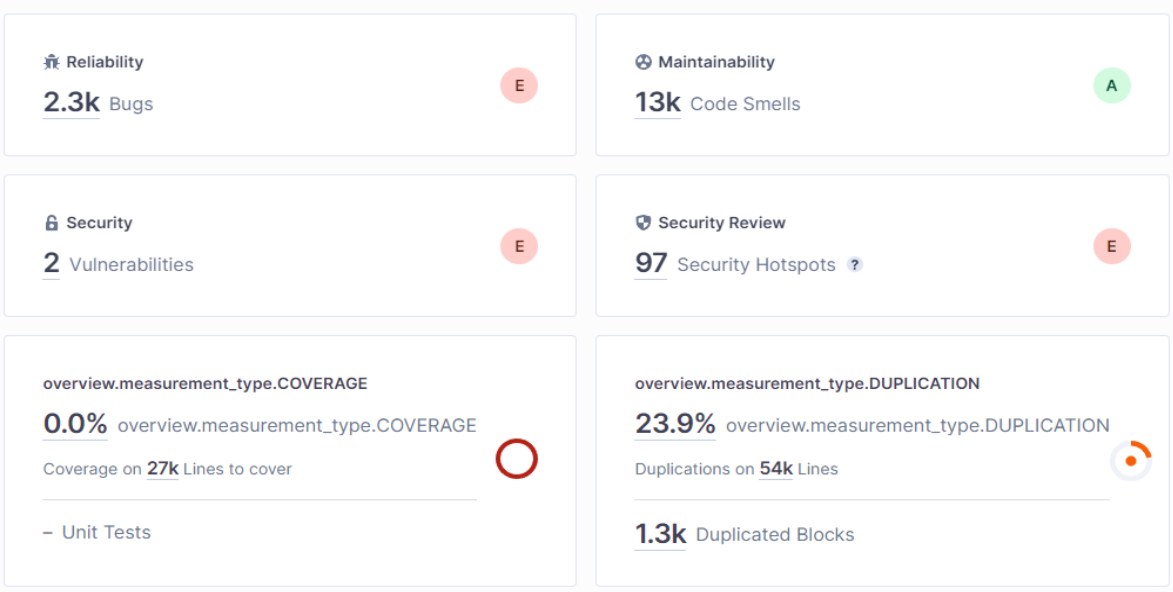
- SonarPHP
- SonarHtml

En el análisis de seguridad y calidad del código fuente de la aplicación Ubidesk, se llevó a cabo mediante la evaluación exhaustiva de las capas de la arquitectura del software, específicamente los controladores, modelos, vistas y (helpers). Este análisis se realizó utilizando dos herramientas especializadas: SonarPHP y SonarHtml.

3.1.3.1. Tipos de vulnerabilidades detectadas

Del análisis de vulnerabilidades detectadas en el análisis de vulnerabilidades de seguridad con la herramienta SonarQube, se detallan a continuación: Del análisis de vulnerabilidades de seguridad, en la figura 11, se muestra el resumen del tipo de fallo de seguridad detectado.

Figura 4
Vulnerabilidades de seguridad de SonarQube



La tabla 12 detalla los problemas de seguridad a nivel de código fuente encontrados por el método estático, donde se detalla el tipo de riesgo, a qué regla pertenece, el tipo de problema, el lenguaje de programación y el número de errores encontrados.

Tabla 11
Detalle de vulnerabilidades detectadas

Tamaño del Proyecto	
Líneas de Código	54.152
Archivos	540
Funciones	1.516
Declaraciones	27.874
Comentarios	
Líneas con comentarios	3.028
Densidad de líneas comentadas	5,3%
Duplicaciones	
Bloques de código duplicados	1.261
Líneas de código duplicadas	24.937
Densidad de líneas duplicadas	23,9%
Archivos con código duplicado	203
Problemas por gravedad	
TOTAL, PROBLEMAS	15.786

Problemas con los bloqueadores	32
Cuestiones críticas	997
Principales problemas	7.652
Problemas menores	7.093
Problemas de información	12

La tabla 13 detalla los puntos de acceso y seguridad en análisis de vulnerabilidades sobre la calidad del código, permitiendo a los desarrolladores abordar y resolver problemas antes de que se conviertan en fallos críticos o vulnerabilidades explotables. Esto no solo mejora la seguridad y funcionalidad de la aplicación, sino que también contribuye a la sostenibilidad y escalabilidad del proyecto a largo plazo.

Tabla 12
Análisis forense de código Ubidesk

Desglose de problemas				
Severidad	**Reglas**	**Tipo**	**Lenguaje**	**Problemas**
BLOCKER	Las propiedades CSS deben ser válidas	BUG	CSS	5
BLOCKER	Las credenciales no deben estar codificadas de forma rígida	VULNERABILIDAD	PHP	2
CRÍTICO	Los literales de cadena no deben duplicarse	CODE_SMELL	PHP	367
CRÍTICO	No se deben usar paréntesis innecesarios para las construcciones	CODE_SMELL	PHP	148
CRÍTICO	Las estructuras de control deben usar llaves	CODE_SMELL	PHP	139
CRÍTICO	La complejidad cognitiva de las funciones no debe ser demasiado alta	CODE_SMELL	PHP	133
CRÍTICO	Las sentencias "switch" deben tener cláusulas "default"	CODE_SMELL	PHP	71
CRÍTICO	Las referencias utilizadas en los bucles "foreach" deben ser "unset"	BUG	PHP	25
CRÍTICO	Una sola línea ejecutada condicionalmente debe denotarse con sangría	CODE_SMELL	PHP	5
CRÍTICO	Los métodos no deben estar vacíos	CODE_SMELL	PHP	4
CRÍTICO	Los condicionales deben comenzar en nuevas líneas	CODE_SMELL	PHP	1
IMPORTANTE	Las filas no deben ser demasiado largas	CODE_SMELL	PHP	3.673
IMPORTANTE	Las etiquetas "<th>" deben tener atributos "id" o "scope"	BUG	HTML	1.295

IMPORTANTE	Los atributos obsoletos en HTML5 no deben usarse	CODE_SMELL	HTML	623
IMPORTANTE	Las secciones de código no deben ser comentadas	CODE_SMELL	PHP	601
IMPORTANTE	Los métodos "privados" no utilizados deben eliminarse	CODE_SMELL	PHP	429
IMPORTANTE	Se debe preferir el uso de espacios de nombres a las funciones "include" o "require"	CODE_SMELL	PHP	250
IMPORTANTE	Las tablas deben tener encabezados	BUG	HTML	156
IMPORTANTE	El elemento "<html>" debe tener un atributo de idioma	BUG	HTML	112
IMPORTANTE	Las secciones de código no deben ser comentadas	CODE_SMELL	HTML	84
IMPORTANTE	"<! Las declaraciones DOCTYPE>" deben aparecer antes de las etiquetas "<html>"	BUG	HTML	49
IMPORTANTE	Los bloques de código anidados no deben dejarse vacíos	CODE_SMELL	PHP	41
IMPORTANTE	HTML "<table>" no debe utilizarse con fines de diseño	CODE_SMELL	HTML	37
IMPORTANTE	Las asignaciones no utilizadas deben eliminarse	CODE_SMELL	PHP	34
IMPORTANTE	Los parámetros de función no utilizados deben eliminarse	CODE_SMELL	PHP	21
IMPORTANTE	Dos bifurcaciones en una estructura condicional no deben tener exactamente la misma implementación	CODE_SMELL	PHP	19
IMPORTANTE	Las variables deben inicializarse antes de su uso	BUG	PHP	18
IMPORTANTE	Las instrucciones "if" contraíbles deben combinarse	CODE_SMELL	PHP	18
IMPORTANTE	Las propiedades no deben duplicarse	BUG	CSS	14
IMPORTANTE	Los pares redundantes de paréntesis deben eliminarse	CODE_SMELL	PHP	14
IMPORTANTE	Los selectores no deben duplicarse	CODE_SMELL	CSS	14
IMPORTANTE	Las funciones no deben tener demasiadas líneas de código	CODE_SMELL	PHP	13
IMPORTANTE	Los campos "privados" no utilizados deben eliminarse	CODE_SMELL	PHP	11
IMPORTANTE	Los bloques de varias líneas deben estar encerrados entre llaves	CODE_SMELL	PHP	11

IMPORTANTE	Los métodos no deben tener implementaciones idénticas	CODE_SMELL	PHP	10
IMPORTANTE	Las clases no deben tener demasiados métodos	CODE_SMELL	PHP	10
IMPORTANTE	Las funciones no deben contener demasiadas instrucciones return	CODE_SMELL	PHP	6
IMPORTANTE	No se debe usar la salida de funciones que no devuelven nada	BUG	PHP	4
IMPORTANTE	Las declaraciones de fuentes deben contener al menos una familia de fuentes genéricas	BUG	CSS	3
IMPORTANTE	Las variables no deben ser autoasignadas	BUG	PHP	3
IMPORTANTE	Los bloques inútiles "if(true) {...}" y "if(false){...}" deben eliminarse	BUG	PHP	3
IMPORTANTE	Los operadores ternarios no deben estar anidados	CODE_SMELL	PHP	2
IMPORTANTE	Las funciones deben usar "return" de forma coherente	CODE_SMELL	PHP	2
IMPORTANTE	Los valores de matriz no deben reemplazarse incondicionalmente	BUG	PHP	2
IMPORTANTE	Las comparaciones de recuento de matrices o objetos contables deberían tener sentido	BUG	PHP	2
IMPORTANTE	Todas las ramas de una estructura condicional no deben tener exactamente la misma implementación	BUG	PHP	1
IMPORTANTE	Los bloques vacíos deben eliminarse	CODE_SMELL	CSS	1
IMPORTANTE	Todo el código debe ser accesible	BUG	PHP	1
MENOR	Las líneas no deben terminar con espacios en blanco al final	CODE_SMELL	PHP	3.457
MENOR	Las palabras clave y constantes de PHP "verdadero", "falso", "nulo" deben estar en minúsculas	CODE_SMELL	PHP	509
MENOR	Las etiquetas "<table>" deben tener una descripción	BUG	HTML	392
MENOR	Los archivos deben contener una nueva línea vacía al final	CODE_SMELL	PHP	382
MENOR	Los nombres de función deben cumplir con una convención de nomenclatura	CODE_SMELL	PHP	340
MENOR	No se deben utilizar caracteres de tabulación	CODE_SMELL	PHP	334
MENOR	Los literales booleanos no deben ser redundantes	CODE_SMELL	PHP	265
MENOR	"empty()" debe usarse para probar el vacío	CODE_SMELL	PHP	222

MENOR	Las variables locales no utilizadas deben eliminarse	CODE_SMELL	PHP	219
MENOR	Los nombres de variables locales y parámetros de función deben cumplir con una convención de nomenclatura	CODE_SMELL	PHP	200
MENOR	El retorno de expresiones booleanas no debe incluirse en una instrucción "if-then-else"	CODE_SMELL	PHP	128
MENOR	Se debe usar "require_once" y "include_once" en lugar de "require" e "include"	BUG	PHP	126
MENOR	Los nombres de campo deben cumplir con una convención de nomenclatura	CODE_SMELL	PHP	102
MENOR	Se deben usar "&&" y "||"	CODE_SMELL	PHP	94
MENOR	Los nombres de clase deben cumplir con una convención de nomenclatura	CODE_SMELL	PHP	63
MENOR	¿La etiqueta de cierre “?>" debe omitirse en los archivos que contienen solo PHP	CODE_SMELL	PHP	49
MENOR	La imagen, el área y el botón con etiquetas de imagen deben tener un atributo "alt"	BUG	HTML	45
MENOR	La visibilidad del método debe declararse explícitamente	BUG	PHP	44
MENOR	Las etiquetas "<fieldset>" deben contener una "<leyenda>"	BUG	HTML	36
MENOR	Las sentencias "switch" deben tener al menos 3 cláusulas "case"	CODE_SMELL	PHP	27
MENOR	Las variables locales no deben declararse y luego devolverse o lanzarse inmediatamente	CODE_SMELL	PHP	12
MENOR	La palabra clave "elseif" debe usarse en lugar de las palabras clave "else if"	CODE_SMELL	PHP	6
MENOR	No se deben ignorar los valores iniciales de los parámetros de función y método	BUG	PHP	2
MENOR	Una llave cerrada debe ubicarse al principio de una línea	CODE_SMELL	PHP	2
MENOR	Las declaraciones vacías deben eliminarse	CODE_SMELL	PHP	2
MENOR	No se debe utilizar la palabra clave "var"	CODE_SMELL	PHP	2
MENOR	Las instrucciones de salto no deben ser redundantes	CODE_SMELL	PHP	1
INFO	Seguimiento de los usos de las etiquetas "TODO"	CODE_SMELL	PHP	12

En la tabla 14 se detalla la calificación general del proyecto basada en los parámetros de confiabilidad, seguridad, mantenibilidad y la deuda técnica.

Tabla 13
Puntos de acceso de seguridad

Puntos de acceso de seguridad			
Tipo	**Problema**	**Prioridad**	**Cantidad**
Auth	Se ha detectado 'contraseña' en el nombre de esta variable, revise esta credencial potencialmente codificada	Alta	2
Weak-cryptography	Asegúrese de que el uso de este generador de números pseudoaleatorios sea seguro aquí	Media	17
Log-injection	Asegúrese de que la configuración de este registrador sea segura.	Baja	15
Others	Asegúrese de que no usar la función de integridad de recursos sea seguro aquí	Baja	58
Calificaciones generales de calidad del código			
Clasificación de confiabilidad		E	
Clasificación de seguridad		E	
Clasificación de mantenibilidad		A	
Deuda Técnica		85d 2h 41min	

3.1.4. Propuesta de mejora para puntos de acceso de seguridad

La propuesta de mejora para puntos de acceso de seguridad de la aplicación Ubidesk tiene como objetivo fortalecer la protección contra las amenazas y vulnerabilidades detectadas. Se centra en proporcionar una capa adicional de seguridad sin comprometer la usabilidad y la accesibilidad para los usuarios autorizados. Tabla 15.

Tabla 14
Calificación basada en confiabilidad, seguridad, mantenibilidad y la deuda técnica.

Problema	**Las credenciales no deben estar codificadas de forma rígida**

Solución propuesta	El problema ha sido identificado en las credenciales de pusher dentro del archivo rjxtools_helper.php en las líneas 846 y 863, para solucionar este problema se debe seguir este procedimiento:
	1. Crear en la carpeta 'maincoregec\application\config' un archivo llamado pusher.php con el siguiente código:
	`<?php`
	`$config['pusher']['app_key'] = "[redacted]";`
	`$config['pusher']['app_secret'] = "[redacted]";`
	`$config['pusher']['app_id'] = "[redacted]";`
	2. Modificar 'maincoregec\application\config\autoload.php':
	Original
	`$autoload['config'] = array();`
	Modificada
	`$autoload['config'] = array('aws','pusher','firebase');`
	3. Modificar 'maincoregec\ci_core\application\helpers\rjxtools_helper.php' para llamar la configuración de claves a utilizar dentro de las llamadas de pusher:
	Original
	`$options = array(`
	`'cluster' => 'us2',`
	`'useTLS' => true`
	`);`
	`$pusher = new Pusher\Pusher(`
	`'[redacted]',`
	`'[redacted]',`
	`'[redacted]',`
	`$options`
	`);`
	Modificada

```
841         $CI = &get_instance();
842         $pusher_config = $CI->config->item( 'pusher' );
843         $options = array(
844             'cluster' => 'us2',
845             'useTLS' => true
846         );
847         $pusher = new Pusher\Pusher(
848             $pusher_config['mi_app_key'],
849             $pusher_config['mi_app_secret'],
850             $pusher_config['mi_app_id'],
851             $options
852         );
```

Beneficios	Mantener las claves de servicios como Pusher en la configuración en lugar de en el código mejora la seguridad, facilita la gestión, promueve la mantenibilidad del código y sigue las mejores prácticas de desarrollo seguro.
Recomendaciones	Cambiar las credenciales por si estuvieran comprometidas.
Problema	**Se ha detectado 'contraseña' en el nombre de esta variable, revise esta credencial potencialmente codificada.**
Solución propuesta	El problema ha sido identificado en las credenciales de Firebase dentro del archivo Firebase_model.php en la línea 20, para solucionar este problema se debe seguir este procedimiento: 1. Crear de la de carpeta 'maincoregec\application\config' un archivo llamado firebase.php con el siguiente código:

```
1   <?php
2   $config['pvd']['email'] = "                    ";
3   $config['pvd']['password'] = "                   F";
```

2. Modificar 'maincoregec\application\config\autoload.php':

Original

```
106     $autoload['config'] = array();
```

Modificada

```
106     $autoload['config'] = array('aws','pusher','firebase');
```

3. Modificar 'maincoregec\ci_core\application\models\Firebase_model.php' inicializar los valores de las propiedades en el constructor:

Original

```
19          public $pvdEmail = "                    ";
20          public $pvdPassword = "                    ";
21
22          public function __construct()
23          {
24              parent::__construct();
25              $this->load->database();
26
27              $factory = (new Factory)
```

Recomendación

```
19          public $pvdEmail = null;
20          public $pvdPassword = null;

30          public function __construct()
31          {
32              parent::__construct();
33              $this->load->database();
34              $CI = &get_instance();
35              $pvd_config = $CI->config->item( 'pvd' );
36              $this->pvdEmail = $pvd_config['email'];
37              $this->pvdPassword = $pvd_config['password'];
```

Beneficios	Mantener las claves de servicios como Firebase en la configuración en lugar de en el código mejora la seguridad, facilita la gestión, promueve la mantenibilidad del código y sigue las mejores prácticas de desarrollo seguro.
Recomendaciones	Cambiar las credenciales por si estuvieran comprometidas.
Problema	**Asegúrese de que el uso de este generador de números pseudoaleatorios sea seguro aquí**
Solución propuesta	En lugar de usar la función rand(); utilizar mt_rand().
Beneficios	La función rand() retorna los mismos valores cada determinado tiempo en caso de que sea usado para claves daría problemas de identidad por eso la opción es usar mt_rand() que tiene algoritmos de generación de números aleatorios mejorados.
Problema	**Asegúrese de que este algoritmo hash débil no se utilice en un contexto sensible aquí.**
Solución propuesta	Se recomienda usar algoritmos alternos de encriptación; en lugar de usar md5 o sha1 usar sha512 siempre y cuando sea aplicable y no afecte la funcionalidad del sistema. Original $hash = hash(‘md5’, $data);

	$hash = hash('sha1', $data); Recomendación $hash = hash('sha512', $data);
Beneficios	Previene ataques de fuerza bruta y diccionario.
Problema	**Asegúrese de que la configuración de este registrador sea segura**
Solución propuesta	Verifique en el código de producción que solo exista un solo lugar donde se establezca el registro de errores. Se han detectado 15 lugares donde se activa para mostrar los errores: 'ini_set('display_errors', 1)'; debe eliminar esas líneas.
Beneficios	Al usar la línea 'ini_set('display_errors', 1)' se podría mostrar información relevante a usuarios no registrados; al quitar esta línea de los archivos que no sean de configuración garantiza un mejor uso del despliegue de errores y evita mostrar información confidencial.
Problema	**Asegúrese de que no usar la función de integridad de recursos sea seguro aquí**
Solución propuesta	Al utilizar recurso de javascript remotos sean seguros a través de la cláusula 'integrity'. Por ejemplo: Original <script src="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote.min.js"></script>
Beneficios	Al utilizar la cláusula integrity se garantiza que los recursos JavaScript remotos se carguen de manera segura y no sean modificados por terceros malintencionados. Esto protege contra ataques de tipo "man-in-the-middle" y asegura la integridad del código JavaScript que se está incorporando en la aplicación.

3.2 Discusión de resultados

De los resultados del análisis de vulnerabilidades de SonarQube destacan los hallazgos significativos y las acciones recomendadas para actualizar la seguridad del código modificando las vulnerabilidades, errores y malas prácticas de programación en el software. La discusión de resultados se centra en comprender la gravedad de estas vulnerabilidades, priorizarlas y proponer soluciones.

Tabla 15
Análisis comparativo de vulnerabilidades entre los métodos de evaluación

Tipo de problemas	Antes	Después
Problemas con los bloqueadores	32	0

Cuestiones críticas	997	893
Principales problemas	7.652	7.559
Problemas menores	7.093	6837
Problemas de información	12	12

De los resultados obtenidos con el análisis ejecutado al código fuente de la aplicación Ubidesk, evaluado mediante los métodos estático y desencadenado (SonarPHP y SonarHtml), permitió identificar diversas categorías de problemas.

A continuación, se presenta un análisis comparativo del estado de los problemas antes y después de la intervención.

3.2.1 Análisis de problemas identificados y su evolución

El análisis muestra progresos significativos en varias áreas clave, especialmente en la eliminación completa de problemas bloqueadores y la reducción de cuestiones críticas y principales problemas. Sin embargo, aún queda trabajo por hacer para seguir mejorando la calidad del código.

Problemas con los bloqueadores

- Antes: 32
- Después: 0

Análisis: Se ha logrado eliminar completamente los problemas bloqueadores, lo cual es un avance significativo. Estos problemas eran críticos y podían impedir el funcionamiento de la aplicación. Su resolución mejora notablemente la estabilidad y la operatividad del sistema.

Cuestiones críticas

- Antes: 997
- Después: 893

Análisis: Se ha logrado una reducción del 10.4% en las cuestiones críticas. Aunque todavía existen un número considerable de problemas críticos, la disminución refleja un esfuerzo efectivo en mitigar vulnerabilidades graves que podrían comprometer la seguridad y la funcionalidad del sistema.

Principales problemas

- Antes: 7,652
- Después: 7,559

Análisis: Se ha conseguido una reducción del 1.2% en los principales problemas. Esta pequeña mejora indica que, aunque se ha trabajado en resolver estos problemas, sigue siendo necesario un enfoque más intensivo para reducir este número de manera significativa.

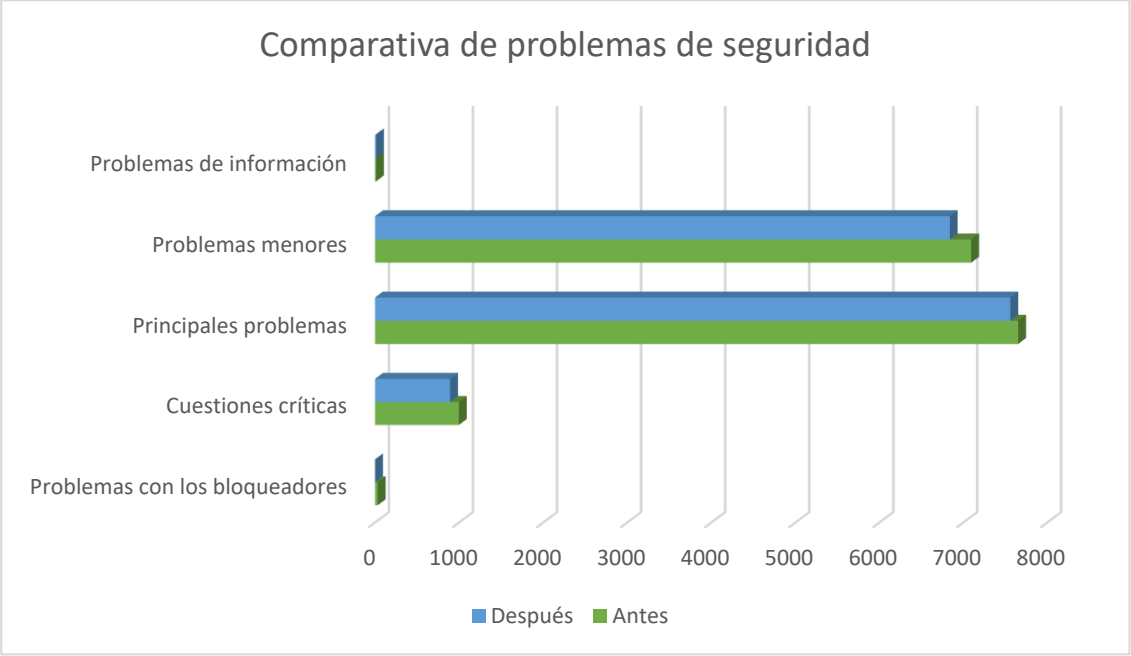
Problemas menores

- Antes: 7,093
- Después: 6,837

Análisis: Se observa una reducción del 3.6% en los problemas menores. La disminución en esta categoría es importante ya que, aunque estos problemas no son críticos, su acumulación puede afectar la mantenibilidad y eficiencia del código a largo plazo.

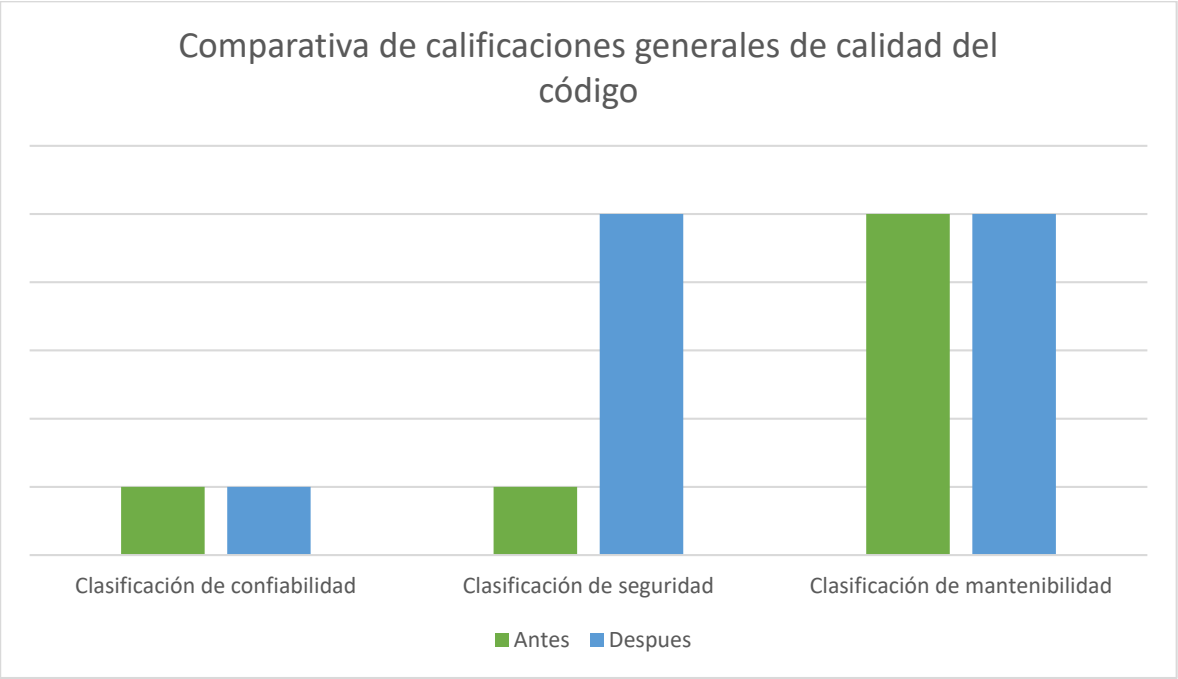
La figura 5 detalla la comparativa del análisis inicial y evaluación y al final una vez aplicadas las recomendaciones, se mejoró el parámetro de seguridad que se definió como prioridad para la mitigación y mejora de la seguridad por parte del equipo de desarrollo de UboraTech.

Figura 5
Análisis de resultados iniciales y finales



La figura 6, muestra el análisis comparativo de resultados con SonarQube. Una vez aplicado, hubo una mejora considerable en aspectos de seguridad respecto a la codificación, lo que nos indica que las recomendaciones aplicadas en los diferentes módulos de la aplicación Ubidesk mejoraron la calidad del producto de software en el ámbito de seguridad de la aplicación y en cuanto a la calidad a nivel de código.

Figura 6
Comparativa de calificaciones generales de calidad del código



La comparación de problemas de seguridad antes y después de la intervención revela mejoras significativas en las cuestiones críticas, lo que indica que las medidas aplicadas fueron eficaces en abordar los problemas más graves. Sin embargo, las reducciones marginales en los problemas menores y principales sugieren que aún hay mucho por hacer en estas áreas. Los problemas de información y con los bloqueadores mostraron una disminución mínima, destacando la necesidad de enfoques más específicos y efectivos.

Es evidente que, aunque se han logrado avances, las estrategias actuales deben revisarse y adaptarse continuamente para abordar de manera integral todos los aspectos de la seguridad. Un enfoque más diferenciado y una evaluación constante serán cruciales para lograr mejoras más sustanciales y sostenibles.

CONCLUSIONES

- En el contexto de riesgos de categoría crítica, se tomaron las acciones de mitigación sobre efectivas en los problemas más graves; sin embargo, se evidencia la persistencia en problemas menores y principales. A pesar de una mejora mínima, los "problemas menores" y "principales problemas" siguen siendo áreas problemáticas. Esto indica la necesidad de una reevaluación de las estrategias aplicadas.

- Impacto general moderado. Aunque hubo una reducción en la mayoría de las categorías de problemas de seguridad, el impacto general fue moderado. Esto sugiere que las medidas implementadas tuvieron cierto éxito, pero no fueron lo suficientemente efectivas para provocar una disminución drástica en todos los problemas.

- La categoría "Cuestiones críticas" mostró una mejora significativa, reduciéndose de aproximadamente 2000 a cerca de 1500 problemas. Esto indica que las intervenciones fueron particularmente efectivas en esta área, abordando con éxito los problemas más graves y urgentes.

- Tanto los "Principales problemas" como los "Problemas menores" experimentaron solo ligeras reducciones. Esto sugiere que las estrategias aplicadas no fueron suficientemente robustas para abordar de manera efectiva estos problemas. Es necesario revisar y posiblemente intensificar las medidas para lograr mejoras más sustanciales en estas categorías.

- La importancia de la evaluación continua. Los resultados subrayan la importancia de la evaluación continua y el ajuste de las estrategias de seguridad. Es importante monitorear constantemente los efectos de las intervenciones y estar dispuestos a adaptar las medidas según sea necesario para abordar de manera efectiva los problemas persistentes.

RECOMENDACIONES

- Establecer un proceso de revisión de código robusto que incluya revisiones por pares y automatizadas para detectar y corregir errores antes de que el código se integre en el proyecto principal utilizando herramientas como SonarQube o Codacy para evaluar la calidad del código y asegurar el cumplimiento de las mejores prácticas.

- Desarrollar y mantener una suite de pruebas automatizadas que cubra tanto pruebas unitarias como de integración. Asegurarse de que todas las funcionalidades clave y los flujos críticos de la aplicación estén cubiertos por pruebas que se ejecuten en cada commit mediante una integración continua (CI) con herramientas como Jenkins, Travis CI o GitHub Actions.

- Asegurarse de que el código siga los principios SOLID para mejorar la mantenibilidad y escalabilidad. Esto incluye garantizar que las clases y métodos sean pequeños y tengan una única responsabilidad, que los componentes sean fácilmente extensibles sin modificar el código existente, y que las dependencias estén invertidas adecuadamente.

- Mantener una documentación clara y actualizada del código, utilizando herramientas como Javadoc para Java o Sphinx para Python. Los comentarios en el código deben ser precisos y útiles, explicando por qué detrás de las decisiones de diseño y cualquier lógica compleja.

- Promover la refactorización continua del código para mejorar su estructura y eliminar deudas técnicas. Esto incluye la simplificación de métodos complejos, la eliminación de duplicaciones y la mejora de la legibilidad del código. Utilizar herramientas como IntelliJ IDEA o ReSharper para facilitar el proceso de refactorización.

- Adherirse a un conjunto de estándares de codificación específicos del lenguaje utilizado. Por ejemplo, seguir PEP 8 para Python o la guía de estilo de Google para JavaScript. Utilizar linters como ESLint para JavaScript o pylint para Python para asegurar el cumplimiento de estos estándares.

BIBLIOGRAFÍA

Brito, H. R. (2021). Riesgos de Seguridad en Pruebas de Penetración Web. *Revista Cubana de Ciencias Informáticas, vol. 15, Esp., pp. 225-243, 2021*, 12.

Cedeño Mosquera, C. J. (2023). *Sistema de ciberseguridad mediante biblioteca de código abierto en los servidores de los hospitales del IESS provincia de Manabí.*

Cloudflare. (2023). Análisis forense digital ¿Cómo se realiza? Técnicas, pasos y mejores prácticas. *Cloudflare, Inc. (www.cloudflare.com/es-la / @cloudflare)*, 8.

García, M. M. (2022). *Gestión de incidentes de ciberseguridad.* RA_MA.

Gasco. (2023). *Análisis forense informático: qué es y cómo se hace.* Obtenido de https://www.gascoabogados.es/analisis-forense-informatico/

González, A. (2021). Análisis de las causas y consecuencias de los accidentes laborales ocurridos en dos proyectos de construcción. *Rev. ing. constr. vol.31 no.1 Santiago abr. 2021*, 12.

Machín, M. N. (2021). Analysis of the causes and consequences of accidents occurring in two constructions projects. *Revista UNISCI / UNISCI Journal, Nº 42 (Octubre/October 2021)* , 15.

Mena, E. A. (2021). *Fundamentos de seguridad informática.* COMPAS.

Moya, J. G. (2023). La importancia de la seguridad informática en la educación digital: retos y soluciones. *ReciMundo*, 8.

Palacios, A. P. (2020). *Seguridad Informática.* Paraninfo.

Pineda, R. S. (2021). Methodology for the investigation of security incidents in web application. *IV Conferencia Científica Internacional UCIENCIA 2021*, 12.

SonicWall-Inc. (2024). Comprensión de los riesgos que conllevan los sitios web empresariale. *1033 McCarthy Boulevard* , 8.

Suárez, J. L. (2021). *IMPORTANCIA DE LA SEGURIDAD INFORMÁTICA Y CIBERSEGURIDAD EN EL MUNDO ACTUAL.* Colombia. Obtenido de http://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/8668/IMPORTANCIA%20DE%20LA%20SEGURIDAD%20INFORM%C3%81TICA%20Y%20CIBERSEGURIDAD%20EN%20EL%20MUNDO%20ACTUAL.pdf?sequence=1&isAllowed=y

Tejada, E. C. (2023). *Gestión de incidentes de seguridad informática.* IC Editorial.

Unad. (2020). Information technology -- Security techniques -- Information security management systems Overview and vocabulary. (ISO/IEC 27000:2018). International Organization for Standardization. *NATIONAL INSTITUTE OF STANDARD AND TECHNOLOGY.*, 12.

Vega, E. (03 de 2021). *SEGURIDAD DE LA INFORMACIÓN.* Obtenido de https://3ciencias.com/: https://3ciencias.com/wp-content/uploads/2021/03/LIBRO-SEGURIDAD-INFORMACIO%CC%81N.pdf

ANEXOS

OF.NRO.2024UB

Ibarra, 5 de julio de 2024

593 997950164
www.uboratech.net
Edificio Milano, Sánchez y Cifuentes y Rafael Larrea Andrade.

Estimado solicitante,

Presente,

En representación de la empresa UBORATECH, me dirijo a usted con gran satisfacción para informarle que hemos recibido y evaluado favorablemente los resultados del trabajo de titulación "Análisis forense de código y procesos de la aplicación Ubidesk" desarrollado por Sr. Bryan Anderson Vilañez Ordóñez con número de cédula 1004199848.

Su trabajo ha sido fundamental para fortalecer la seguridad de nuestra aplicación Ubidesk y contribuir al avance en el campo de la seguridad informática. Agradecemos profundamente su dedicación, profesionalismo y la calidad excepcional de su investigación.

Los hallazgos y recomendaciones presentadas en su tesis serán de gran utilidad para implementar mejoras significativas en la seguridad de nuestra aplicación. Su trabajo demuestra un profundo conocimiento de las técnicas de análisis forense y una comprensión clara de los desafíos que enfrenta la seguridad en el entorno digital actual.

En UBORATECH valoramos enormemente el aporte de su investigación y nos complace extenderle nuestra más sincera felicitación por este logro académico. Estamos seguros de que su talento y experiencia le permitirán alcanzar grandes éxitos en su futuro profesional.

Como muestra de nuestro agradecimiento, nos complace otorgarle a usted,

Sr. Vilañez Ordóñez.

Atentamente,

WILLIAM MIGUEL AGUINAGA VILCA

Firmado digitalmente por WILLIAM MIGUEL AGUINAGA VILCA
Fecha: 2024.07.05 12:18:49 -05'00'

Ing. William Aguinaga
CEO/UBORATECH

Code Quality Report

Ubidesk

Branch: main

Project Key: ubideskproject

Project Version: 1.0

Project Quality Gate: Sonar way

SonarQube Project Qualifier:

- Key: TRK
- Name: PROJECT
- Translated: Proyecto

Analysis date: 2024-03-30T21:27:17-0500

Analysis id: AY6SViwFWdLI_y4UHT0K

bitegarden sonarqube

This is the default generated Open Document report from the template of bitegarden Report for SonarQube™
If you want to change this template check out: https://www.bitegarden.com/sonarqube-report-documentation

Project Quality Gate Details

Quality Gate Status			OK		
Condition	**Value**	**Comparator**	**Threshold**	**Type**	**Status**
-	-	-	-	-	-

Project Size

Lines of code	26.617
Files	261
Directories	
Functions	1.128
Statements	15.828

Comments

Lines with comments	2.816
Comment lines density	9,6%

This is the default generated Open Document report from the template of bitegarden Report for SonarQube™
If you want to change this template check out: https://www.bitegarden.com/sonarqube-report-documentation

Duplications

Duplicated code blocks	615
Duplicated lines of code	9.199
Duplicated lines density	21,5%
Files with duplicated code	135

Coverage

Code coverage	0%
Line coverage	0%
Number of unit tests	0
Unit test with errors	0
Unit tests with failures	0
Unit test success density	0%
Unit test execution time	0 ms.
Unit test execution time (duration)	0
Branch coverage	0%