Análisis de metodologías y buenas prácticas empleadas para el desarrollo seguro de aplicaciones

Yeferzon David Quisoboni Argote

Asesor

Yenny Stella Nuñez Alvarez

Universidad Nacional Abierta y a Distancia – UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Especialización en Seguridad Informática

2025

Agradecimientos

Agradezco a las directivas de la Universidad Nacional Abierta y a Distancia UNAD, quienes con su trabajo continuo nos brindan la oportunidad de estudiar y laborar, por otro lado, a cada uno de los tutores y asesores que me acompañaron en el proceso les reconozco que sin su apoyo y colaboración éste logro no hubiera sido posible.

Dedicatoria

Con amor dedico este trabajo a mi hija, que con su carisma y comprensión me acompañó en cada etapa vivida, colaborándome de manera indirecta, minimizando mis preocupaciones de padre, también lo dedico a mi esposa que con su apoyo, consagración y paciencia disminuyo mis tareas en el hogar permitiéndome una entrega más tranquila en el estudio y trabajo.

Resumen

En la actualidad el uso de las tecnologías en los diferentes procesos empresariales y del diario vivir de las personas ha disparado el desarrollo de aplicaciones o sistemas de información para facilitar los procesos. En estos sistemas se guarda información que es relevante para las empresas, por lo cual es necesario implementar buenas prácticas de desarrollo las cuales estén encaminadas a que las aplicaciones o sistemas sean seguros, pues también se ha evolucionado en la forma cómo los ciberdelincuentes vulneran los sistemas, por este motivo es necesario que estas prácticas sean tenidas en cuentas en todas las etapas del ciclo de vida del desarrollo de software (levantamiento de requerimientos, análisis, diseño, construcción y pruebas), esto con el fin de identificar las vulnerabilidades del software de manera temprana para que las aplicaciones salgan a producción con los menores riesgos de seguridad o amenazas los cuales puedan provocar problemas futuros en la seguridad de la información que afecten a las empresas y generen pérdidas económicas, de credibilidad entre otros.

Palabras clave: buenas prácticas, ciberataque, seguridad informática, software, vulnerabilidad

Abstract

Currently, the use of technology in different business processes and in people's daily lives has triggered the development of applications or information systems to facilitate processes. Information that is relevant to companies is stored in these systems, for which it is necessary to implement good development practices which are aimed at ensuring that applications or systems are secure, since the way in which cybercriminals violate systems has also evolved. For this reason, it is necessary that these practices be taken into account in all stages of the software development life cycle (gathering requirements, analysis, design, construction and testing), in order to identify software vulnerabilities. early so that the applications go into production with the least security risks or threats which may cause future problems in information security that affect companies and generate economic losses, credibility among others.

Keywords: good practices, cyberattack, computer security, software, vulnerability

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Glosario

Ciberataque: “Un ciberataque es un conjunto de acciones ofensivas contra sistemas de información”. (Bello, 2028)

Ciberseguridad: “La ciberseguridad hace referencia al conjunto de tecnologías, procesos y prácticas diseñadas para proteger redes y datos de ataques, daños o acceso no autorizado”. (Galiana, 2023)

Infraestructura: “La infraestructura de tecnología de la información, o infraestructura de TI, se refiere al conjunto de componentes necesarios para el funcionamiento y la gestión de los servicios empresariales de TI y entornos de TI” (IBM, 2022).

Ingeniería social: “La ingeniería social es la práctica de utilizar técnicas psicológicas para manipular el comportamiento. La ingeniería social se produce aprovechando el error humano y animando a las víctimas a actuar en contra de sus intereses”. (Bodnar, 2020)

Malware: “Malware es un término que abarca cualquier tipo de software malicioso diseñado para dañar o explotar cualquier dispositivo, servicio o red programable”. (McAfee, 2022)

Prueba: “La prueba de software es el proceso de evaluación y verificación de un producto o aplicación de software para saber si hace lo que se supone que debe hacer”. (IBM, 2022)

Software: “todo componente intangible (y no físico) que forma parte de dispositivos como computadoras, teléfonos móviles o tabletas y que permite su funcionamiento”. (Editorial Etecé, 2023)

Tecnología: “La tecnología es el conjunto de conocimientos y técnicas que se aplican de manera ordenada para alcanzar un determinado objetivo o resolver un problema”. (Roldán, 2017)

Red: “es aquella que sirve para permitir el intercambio digital de datos entre distintos nodos”. (European Knowledge Center for Information Technology, 2022)

Virus: “Un virus informático es un tipo de programa o código malicioso escrito para modificar el funcionamiento de un equipo”. (Norton, 2018)

DevOps: “El término DevOps, que es una combinación de los términos ingleses development (desarrollo) y operations (operaciones), designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante.” (Microsoft, 2022)

Introducción

En la presente monografía se analizan algunas de las metodologías más reconocidas para el desarrollo seguro de software, como el Security Development Lifecycle (SDL), OpenSAMM, OWASP (Open Web Application Security Project), CbyC, TSP-Secure y DevSecOps. El objetivo de este análisis es identificar las características, ventajas y desventajas de cada una de estas metodologías, a partir de investigaciones previas realizadas por diversos autores.

Además, se aborda una revisión documental sobre los principales tipos de ataques que afectan a las aplicaciones modernas, como la ingeniería social, el ransomware, la inyección de código malicioso y los ataques de denegación de servicio. Este análisis busca identificar las vulnerabilidades más comunes que permiten la efectividad de dichos ataques.

Por último, se exponen algunas de las buenas prácticas más utilizadas en el desarrollo de aplicaciones seguras, como la validación de entradas, las pruebas de penetración y el uso adecuado de sentencias SQL y contenido en URLs. El objetivo es destacar la importancia de seguir estas prácticas para mitigar riesgos y asegurar la integridad de las aplicaciones.

Planteamiento del Problema

La ciberdelincuencia se ha convertido en una de las amenazas más prevalentes a nivel global. Según un informe reciente, "en el tercer trimestre de 2022, los ataques cibernéticos aumentaron un 28% en todos los sectores económicos a nivel mundial, tendencia que continuó en 2023" (Smith, 2023). Esta escalada en los ataques responde al uso de herramientas tecnológicas avanzadas por parte de los delincuentes, quienes explotan vulnerabilidades en sistemas y aplicaciones para robar información confidencial y personal. Este panorama plantea la necesidad urgente de mitigar los riesgos asociados a las brechas de seguridad, haciendo imprescindible el uso de buenas prácticas y metodologías de desarrollo seguro de software.

A medida que la tecnología se integra más profundamente en sectores como el financiero, industrial, administrativo y de salud, también aumenta la superficie de ataque potencial. Sin embargo, muchas empresas todavía enfrentan desafíos para asegurar sus aplicaciones. Entre los problemas más comunes se encuentran la falta de alineación entre los equipos de desarrollo y pruebas con normativas, políticas y estándares internacionales de seguridad, lo que conduce a la ausencia de planificación adecuada y control durante el ciclo de vida del desarrollo del software (Fernández-Medina, López, & Piattini, 2018). Esto incrementa el riesgo de que los productos desarrollados contengan vulnerabilidades no detectadas, exponiendo a las organizaciones a pérdidas económicas y daños reputacionales.

Además, se observa que en muchas organizaciones las pruebas de seguridad no son una parte integral del proceso de desarrollo. La falta de análisis exhaustivo para la identificación de vulnerabilidades incrementa el riesgo de que las aplicaciones liberadas al mercado presenten brechas de seguridad significativas, lo que no solo pone en peligro la información de los

usuarios, sino que también compromete la adopción de dichas aplicaciones (Halfond, Viegas, & Orso, 2019).

En este contexto, es fundamental contar con un enfoque que integre buenas prácticas y metodologías de desarrollo seguro, las cuales permitan la identificación y mitigación proactiva de amenazas y vulnerabilidades a lo largo del ciclo de vida del software. Por tanto, esta monografía pretende responder a la siguiente pregunta de investigación:

¿Cómo el uso de metodologías y buenas prácticas de seguridad contribuyen a la mitigación de riesgos, amenazas y vulnerabilidades durante el ciclo de vida del desarrollo de software, garantizando así la seguridad de la información?

Justificación

La creciente transformación digital ha provocado la integración de tecnologías en múltiples aspectos de la vida cotidiana y profesional, desde procesos laborales y financieros hasta los sectores industriales y educativos. En este escenario, asegurar la protección de la información se ha vuelto una prioridad para las organizaciones, debido a la sensibilidad de los datos manejados y el impacto económico y reputacional que una brecha de seguridad puede generar. Aunque existen estándares internacionales como la familia ISO 27000, que busca la implementación efectiva de controles de seguridad en las empresas, resulta imperativo recopilar y consolidar las metodologías y técnicas exitosas empleadas en diversos proyectos. Esto permitirá proponer un conjunto actualizado de buenas prácticas y metodologías orientadas al desarrollo de aplicaciones seguras, proporcionando una referencia sólida para futuros desarrollos.

El creciente nivel de amenaza cibernética a nivel global es un factor clave en esta necesidad. Según el informe de (SonicWall, 2022), Colombia se posicionó entre los diez países más atacados del mundo en 2021, con más de 11 millones de ciberamenazas detectadas. La situación es alarmante para las empresas del país, que sufren un promedio de 2.387 ataques semanales, ubicándose como la tercera nación más afectada en América Latina (La Nota Económica, 2022). Esta alarmante frecuencia de ataques, que incluyen desde ransomware hasta phishing y explotación de vulnerabilidades en software, subraya la necesidad urgente de adoptar estrategias de seguridad proactivas durante el ciclo de vida del desarrollo de software.

En este contexto, el presente estudio se justifica en la necesidad de establecer una lista de buenas prácticas de seguridad que puedan integrarse en cada fase del desarrollo de software. El propósito es que las aplicaciones cumplan con los más altos estándares de seguridad, minimizando así las vulnerabilidades frente a ciberataques. Esta propuesta busca aprovechar las

metodologías exitosas identificadas en diversos proyectos de la industria, sirviendo como una guía sólida y basada en la experiencia para el desarrollo seguro de aplicaciones.

Objetivos

Objetivo General

Analizar las principales metodologías y buenas prácticas utilizadas para el desarrollo seguro de aplicaciones mediante una revisión bibliográfica que permita establecer cuál de ellas reducen en mayor medida las vulnerabilidades en las aplicaciones.

Objetivos Específicos

Identificar las metodologías y buenas prácticas empleadas en el desarrollo de sistemas o aplicaciones seguras mediante una revisión exhaustiva de reportes e investigaciones con el fin de establecer sus características distintivas.

Examinar los diferentes tipos de ataques que sufren las aplicaciones y que son exitosos debido a las malas prácticas utilizadas en su implantación mediante una revisión bibliográfica que permita caracterizarlos.

Definir los factores de riesgo más frecuentes en las aplicaciones o sistemas por medio de una revisión bibliográfica que permita identificar las brechas de seguridad más comunes.

Proponer un listado de buenas prácticas y metodologías para el desarrollo seguro de aplicaciones basados en los informes o investigaciones.

Marco Referencial

Antecedentes

A lo largo de los últimos años, se han desarrollado varias investigaciones que abordan las metodologías y buenas prácticas en el desarrollo de software seguro. Estas investigaciones constituyen un marco importante para comprender la relevancia de la seguridad en el ciclo de vida del desarrollo de software, y permiten contextualizar los aportes que este trabajo propone en términos de prácticas efectivas y metodologías de seguridad.

Uno de los trabajos destacados es la Política general para la aplicación de buenas prácticas de seguridad y privacidad de la información en el desarrollo de software para empresas medianas y pequeñas en el estándar OWASP, en el que se presentan diversos elementos para la implementación de buenas prácticas de seguridad basadas en las pautas de OWASP (Amaya Valencia, 2022). Este trabajo subraya la importancia de adoptar un enfoque de seguridad en cada fase del ciclo de desarrollo, alineándose con los objetivos de esta investigación.

Asimismo, la Guía de buenas prácticas de seguridad en el desarrollo de software con base en estándares reconocidos en empresas de desarrollo de software propone un enfoque basado en estándares como la ISO 27001, CMMI Nivel de Madurez 2, PSP y TSP para el desarrollo de software seguro (Fernández-Bernal, 2018). Esta investigación destaca la necesidad de incorporar un marco normativo sólido en los proyectos de desarrollo de software, tema que será central en este trabajo.

Por otro lado, la monografía Ataques cibernéticos más frecuentes en las mipymes de Colombia durante el periodo 2020 - 2021 de la pandemia COVID-19 (Martínez-Vargas, 2021) ofrece un análisis de los ciberataques más comunes sufridos por micro, pequeñas y medianas empresas en Colombia durante la pandemia. Este estudio destaca las vulnerabilidades presentes

en las organizaciones durante ese periodo y propone herramientas Open Source para mitigar los riesgos, lo que subraya la importancia de establecer medidas proactivas de seguridad desde las primeras fases del desarrollo de software.

En el artículo Firewall – Linux: Una Solución de Seguridad Informática para Pymes (Martínez, Pacheco-Meneses, & Zuñiga-Silgado, 2009), se aborda la configuración de un host con Iptables para la protección de redes. Este estudio demuestra cómo las soluciones de seguridad basadas en software libre pueden ofrecer una alternativa viable y efectiva para las pequeñas y medianas empresas que buscan mejorar su seguridad.

Finalmente, el artículo Medida del nivel de seguridad informática de las pequeñas y medianas empresas (PYMEs) en Colombia propone una herramienta para evaluar las vulnerabilidades presentes en las PYMEs (Sánchez-Sánchez, García-González, Triana, & Perez-Coronell, 2021). Este estudio destaca la necesidad de contar con procesos de evaluación y monitoreo constantes para identificar y mitigar vulnerabilidades en las organizaciones.

En resumen, estos antecedentes evidencian la importancia de adoptar metodologías y buenas prácticas en el desarrollo de software seguro, ofreciendo diversas perspectivas y enfoques que refuerzan la necesidad de proteger la información a lo largo del ciclo de vida del software. La presente investigación se construye sobre estos estudios, con el objetivo de proponer un conjunto de buenas prácticas que sirvan como referencia para el desarrollo de aplicaciones seguras.

Marco Conceptual

Ciberseguridad

La ciberseguridad se refiere a la protección de los sistemas informáticos, redes, programas y datos contra ataques, daños o acceso no autorizado (Lindemulder & Kosinski, 2024). La creciente digitalización ha ampliado el alcance de las amenazas cibernéticas, que ahora incluyen desde simples ataques de malware hasta sofisticados ataques de ransomware y denegación de servicio. Según el Instituto Nacional de Estándares y Tecnología (NIST), la ciberseguridad busca asegurar la confidencialidad**,** integridad **y** disponibilidad de la información en sistemas conectados (Stine & Barrett, 2022). En el contexto colombiano, la situación es particularmente crítica debido al aumento del número de ataques cibernéticos dirigidos a empresas locales (SonicWall, 2022), lo que subraya la urgencia de fortalecer la seguridad en el desarrollo de software.

Seguridad en el Desarrollo de Software

El desarrollo de software seguro es un enfoque que busca integrar prácticas de seguridad desde las primeras fases del ciclo de vida del desarrollo de software, con el fin de prevenir y mitigar las vulnerabilidades que puedan ser explotadas por atacantes. Esto implica la implementación de métodos de codificación seguros**,** validación de entradas y salidas, uso de bibliotecas seguras y la ejecución de pruebas de seguridad antes de desplegar el software en un entorno de producción (OWASP, 2021). La seguridad en el desarrollo de software es crítica, ya que una vulnerabilidad no detectada puede permitir accesos no autorizados a información sensible o causar fallos de funcionamiento en el sistema (ISO/IEC 27034).

Metodologías de Desarrollo Seguro

Las metodologías de desarrollo seguro como DevSecOps**,** Security Development Lifecycle (SDL) **y** OWASP (Open Web Application Security Project) promueven la integración de medidas de seguridad a lo largo de todas las fases del desarrollo de software.

DevSecOps es una evolución de la metodología DevOps que integra la seguridad en todo el proceso de desarrollo ágil, buscando automatizar las pruebas de seguridad y garantizar que se detecten vulnerabilidades lo antes posible (SENTRIO, 2021).

SDL es un enfoque que se enfoca en la seguridad desde el diseño hasta la implementación y el mantenimiento del software, con prácticas rigurosas de gestión de riesgos (Microsoft, 2022).

OWASP proporciona una guía de las vulnerabilidades más comunes y cómo mitigar los riesgos en aplicaciones web (OWASP, 2021).

Ciclo de Vida de la Vulnerabilidad

El ciclo de vida de una vulnerabilidad abarca desde su descubrimiento hasta su corrección. Incluye fases de identificación, reporte, análisis, asignación de parches, pruebas y cierre (Stine & Barrett, 2022). Un manejo adecuado de este ciclo es crucial para reducir el tiempo de exposición de un sistema vulnerable y para asegurar que las soluciones a las vulnerabilidades sean implementadas eficazmente. La gestión de parches y la respuesta a incidentes son componentes críticos para garantizar que las vulnerabilidades descubiertas no sean explotadas antes de ser corregidas.

Amenazas y Vulnerabilidades Comunes en el Desarrollo de Software

Las vulnerabilidades en software son puertas de entrada para ciberataques. Las más comunes incluyen inyección de código SQL**,** Cross-Site Scripting (XSS) **y** fallos en la validación

de entradas**.** Estas vulnerabilidades suelen explotarse para obtener acceso no autorizado a bases de datos o para ejecutar comandos maliciosos en un sistema. OWASP categoriza estas vulnerabilidades y proporciona medidas para mitigarlas, lo que es esencial para el desarrollo de aplicaciones web seguras (OWASP, 2021).

En el contexto colombiano, estudios muestran un incremento en ataques de ransomware y denegación de servicio, dirigidos particularmente a pequeñas y medianas empresas (Martínez-Vargas, 2021). Estos ataques suelen aprovecharse de vulnerabilidades no gestionadas adecuadamente, lo que resalta la importancia de aplicar pruebas de seguridad regulares y de mantener los sistemas actualizados (SonicWall, 2022).

Normas y Estándares de Seguridad

Las normas y estándares internacionales proporcionan un marco de referencia para garantizar la seguridad de la información y el desarrollo seguro de software. La ISO/IEC 27001 establece los requisitos para implementar un Sistema de Gestión de Seguridad de la Información (SGSI) (GlobalSuite Solutions, 2023), mientras que la ISO/IEC 27034 se enfoca específicamente en la seguridad de las aplicaciones (CEC, 2020). Estas normas son fundamentales para empresas que desean asegurar que sus procesos de desarrollo cumplen con las mejores prácticas globales. En particular, Colombia ha adoptado estos estándares a través de la Ley 1581 de 2012, que regula el tratamiento de datos personales y la seguridad de la información en las empresas (Superintendencia de Industria y Comercio, 2012).

Pruebas de Seguridad y Evaluación de Vulnerabilidades

Las pruebas de seguridad juegan un papel esencial en la identificación y corrección de vulnerabilidades en el software antes de que sea desplegado. Estas pruebas incluyen pruebas de penetración (pentesting), análisis de código estático y dinámico, y simulaciones de ataques

Según (Halfond, Viegas, & Orso, 2006). Las herramientas como SonarQube **y** Burp Suite permiten a los desarrolladores y equipos de seguridad identificar brechas de seguridad de manera proactiva. La falta de pruebas rigurosas es una de las principales razones por las que las aplicaciones son vulnerables a ataques una vez desplegadas (OWASP, 2021).

Marco Teórico

Seguridad en el Desarrollo de Software

El desarrollo de software seguro es esencial en un entorno digital cada vez más expuesto a ataques cibernéticos. Según Microsoft (2020), la seguridad en el desarrollo de software debe ser integrada desde las primeras fases del ciclo de vida de desarrollo de software (SDLC). La metodología conocida como Security Development Lifecycle (SDL), propuesta por Microsoft, aborda este aspecto al incorporar prácticas de seguridad en cada etapa del desarrollo (Microsoft, 2022).

De manera similar, el proyecto Open Web Application Security Project (OWASP) identifica las principales amenazas a la seguridad de las aplicaciones web y ofrece recomendaciones para mitigarlas. En particular, el OWASP Top Ten es una referencia global que lista las vulnerabilidades más comunes y críticas en aplicaciones web, como inyección SQL y Cross-Site Scripting (XSS), y proporciona pautas para reducir estas vulnerabilidades (OWASP, 2021).

Ciberseguridad y su Relevancia Actual

La ciberseguridad se define como la protección de sistemas, redes y datos de ataques malintencionados, y su importancia ha crecido exponencialmente en la era digital. Los ataques cibernéticos contra empresas, tanto grandes como pequeñas, son cada vez más comunes. Según la Ley 1581 de 2012 de Colombia, la protección de los datos personales y la privacidad de la

información son elementos clave en la regulación de las empresas colombianas (Superintendencia de Industria y Comercio, 2012). Esto subraya la necesidad de implementar estándares de seguridad robustos.

En respuesta a esta necesidad, la ISO/IEC 27001 se ha convertido en un estándar internacionalmente reconocido para la gestión de la seguridad de la información. Este estándar proporciona un marco para gestionar la seguridad de la información en cualquier tipo de organización, lo que lo convierte en una herramienta valiosa para mitigar el riesgo de ataques (ISO/IEC 27001, 2018).

Normas Internacionales para el Desarrollo Seguro de Software

El cumplimiento de estándares internacionales de seguridad es esencial para garantizar que las aplicaciones sean seguras. La ISO/IEC 27034 establece prácticas y controles que aseguran que las aplicaciones sean desarrolladas con los requisitos de seguridad adecuados en mente. Estos estándares son fundamentales para proteger la confidencialidad, integridad y disponibilidad de los datos (ISO/IEC 27034, 2011).

Metodologías de Seguridad en el Desarrollo de Software

Existen varias metodologías que permiten integrar la seguridad en el proceso de desarrollo de software. DevSecOps es una de las más relevantes actualmente, ya que combina las prácticas de desarrollo ágil (DevOps) con un enfoque de seguridad integrado. Esto permite que las pruebas de seguridad se realicen de manera continua durante el ciclo de vida del desarrollo (AWS Amazon). Además, herramientas automatizadas de análisis estático y dinámico del código son usadas para identificar vulnerabilidades lo antes posible.

Por su parte, la metodología SDL se centra en realizar análisis de amenazas, aplicar prácticas de codificación seguras y realizar pruebas de seguridad en todas las etapas del ciclo de vida del desarrollo (Microsoft, 2022).

Impacto de los Ciberataques en las PYMEs

En Colombia, las PYMEs han sido particularmente vulnerables a los ataques cibernéticos, especialmente durante la pandemia de COVID-19. Los ataques más comunes incluyen el ransomware y el phishing, que pueden causar pérdidas económicas significativas y dañar la reputación de las empresas. Un estudio de Martínez-Vargas (2021) mostró que muchas de estas empresas no cuentan con los recursos para implementar sistemas de seguridad adecuados, lo que las hace un objetivo fácil para los ciberdelincuentes (Martínez-Vargas, 2021).

Marco Legal

Normas ISO/IEC

ISO 12207 – Modelos de Ciclos de Vida del Software.

El ciclo de vida del desarrollo de software puede estar adoptar diferentes metodologías, pero hay una sola norma internacional, la cual es la ISO 12207. En la cual se establece: “un marco común para los procesos del ciclo de vida del software, con una terminología bien definida, que puede ser referenciada por la industria del software. Contiene procesos, actividades y tareas que deben aplicarse durante la adquisición de un sistema, producto o servicio de software y durante el suministro, el desarrollo, la explotación, el mantenimiento y la eliminación de los productos de software. Esto se logra mediante la participación de las partes interesadas, con el objetivo final de lograr la satisfacción del cliente. Esta Norma Internacional se aplica a la adquisición de sistemas, productos y servicios de software, al suministro, desarrollo, operación, mantenimiento y eliminación de productos de software y a la parte de software de cualquier

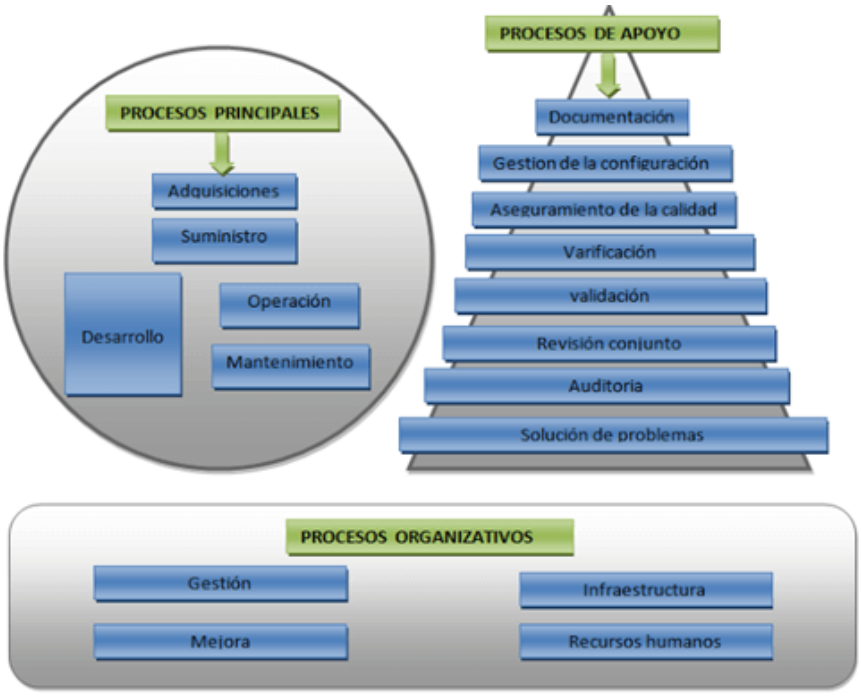
sistema, ya sea realizada interna o externamente a una organización. El software incluye la parte de software del firmware. Se incluyen aquellos aspectos de la definición del sistema necesarios para proporcionar el contexto de los productos y servicios de software. Esta Norma Internacional también proporciona procesos que pueden emplearse para definir, controlar y mejorar los procesos del ciclo de vida del software dentro de una organización o un proyecto. Los procesos, actividades y tareas de esta Norma Internacional también pueden aplicarse durante la adquisición de un sistema que contenga software" (ISO/IEC/IEEE 12207, 2017).

Esta norma está compuesta por 17 procesos, agrupados en tres categorías: procesos principales, de apoyo y de organización.

Figura 1. Procesos organizativos de la Norma ISO 12207

Figura 1

Proceso Organizativos de la Norma ISO 12207



Nota. La imagen ilustra los Procesos Organizativos de la Norma ISO 12207, que define el marco para la gestión del ciclo de vida de software. En esta estructura, los procesos organizativos

incluyen la gestión, mejora, infraestructura y recursos humanos, todos enfocados en apoyar la implementación y ejecución efectiva de los procesos principales y de apoyo. Estos procesos aseguran la correcta administración de los recursos y la optimización de las operaciones dentro de una organización, contribuyendo a la mejora continua y la gestión eficiente de proyectos de desarrollo y mantenimiento de software, alineados con los estándares internacionales de calidad. Tomada de (Arciniega, Normas y Estándares de calidad para el desarrollo de Software, 2022)

ISO/IEC 15408: “proporciona una guía muy útil que define un criterio estándar a usar como base para la evaluación de las propiedades y características de seguridad de determinado producto o Sistema IT y proporciona criterios y argumentos comprensibles para los diferentes perfiles de actores que se encuentran relacionados con las tecnologías de la seguridad” (Rafael, 2012) tales como:

- Fabricantes, desarrolladores de productos o sistemas de información.
- Los expertos de seguridad analizan y certifican a qué nivel se ajusta un producto, o sistema de información a los requisitos de seguridad.
- Los usuarios pueden expresar sus necesidades de acuerdo con el nivel de confianza y seguridad de un producto.

Las buenas prácticas (La norma 7799): “La organización ISO ha reservado recientemente su serie 27000 para temas relacionados con la seguridad de la información. Existen o están previstas las siguientes normas:

ISO 27000, vocabulario y definiciones (terminología para el resto de los estándares de la serie).

ISO 27001, Especificación del sistema de gestión de la seguridad de la información (SGSI). Esta norma será certificable bajo los esquemas nacionales de cada país.

ISO 27002, actualmente la ISO 17799. que describe el Código de buenas prácticas para la gestión de la seguridad de la información.

ISO 27003, que contiene una guía de implementación.

ISO 27004, estándar relacionado con las métricas y medidas en materia de seguridad para evaluar la efectividad del sistema de gestión de la seguridad de la información.

ISO 27005, que proporciona el estándar base para la gestión del riesgo de la seguridad en sistemas de información" (Heredero, 2006).

Metodologías y Buenas Prácticas más Utilizadas en los Proyectos para Desarrollar Software Seguro

Es importante la aplicación de metodologías y buenas prácticas a la hora de realizar desarrollo de software seguro, ya que en el caso de la metodología permite aplicar una serie de actividades y pasos para alcanzar un objetivo realizando procesos de planificación y control para tener mayor conocimiento del estado del proyecto que permitan aplicar acciones correctivas, evaluar los impactos y realizar el análisis de resultados, además la aplicación de las mejores prácticas recopiladas hasta el momento, permiten obtener mejores resultados. En el caso de desarrollo de software seguro no es diferente, pues la aplicación de buenas prácticas y metodologías permitirán obtener aplicaciones más seguras en combinación con políticas de seguridad informática y brindará mayor protección a la información de las diferentes empresas y organizaciones.

Metodologías para Desarrollo de Software Seguro

Según la investigación realizada para el desarrollo seguro de software en la actualidad existen diferentes metodologías tales como:

Security Development Lifecycle (SDL), Software Assurance Maturity Model (SAMM o OpenSAMM), Building Security in Maturity Model (BSIMM2), Comprehensive Lightweight Application Security Process (CLASP), Team Software Process (TSP), Correctness by Construction (CBYC).

Security Development Lifecycle (SDL)

El ciclo de vida de desarrollo de seguridad (SDL) es una iniciativa de Microsoft que “consiste en un conjunto de prácticas que respaldan los requisitos de cumplimiento y garantía de seguridad. SDL ayuda a los desarrolladores a crear software más seguro al reducir la cantidad y

la gravedad de las vulnerabilidades en el software, al mismo tiempo que reduce el costo de desarrollo" (Microsoft, 2022). Esta metodología ha sido compartida a todos los clientes y socios estratégicos que se dedican al desarrollo de software, esto con el objetivo de disminuir los posibles ciberataques.

Por otro lado, es importante tener claro que los procedimientos de TI no deben de ser remplazados por esta metodología, la cual aplica en procesos de desarrollo de software.

"Las prácticas que propone SDL van desde una etapa de entrenamiento sobre temas de seguridad, pasando por análisis estático, análisis dinámico, fuzz testing del código hasta tener plan de respuesta a incidentes. Una de las características principales de SDL es el modelado de amenazas que sirve a los desarrolladores para encontrar partes del código, donde probablemente exista vulnerabilidades o sean objeto de ataques" (Brito-Abundis, 2013).

Opensamm

"Es un marco de trabajo abierto para ayudar a las organizaciones a formular e implementar una estrategia de seguridad para software que se adecue a las necesidades específicas que está enfrentado la organización" (Chandra, 2013). Esta metodología proporciona recursos que permiten realizar las siguientes actividades:

- Evaluar las prácticas de seguridad en software existentes en la organización (Chandra, 2013).
- Construir un programa de seguridad en software balanceado en iteraciones bien definidas (Chandra, 2013).
- Demostrar mejoras concretas en el programa de aseguramiento de software (Chandra, 2013).

- Definir y medir las actividades relacionadas con seguridad en la organización (Chandra, 2013).

"SAMM fue definido para ser flexible de manera que pueda ser utilizado por organizaciones pequeñas, medianas o grandes que utilicen cualquier estilo de desarrollo. Además, este modelo puede ser aplicado en toda la organización, en una sola línea de negocio o incluso en un proyecto en particular" (Chandra, 2013).

Este modelo está fundamentado en las funciones de negocio centrales relacionadas con el desarrollo de software. Está constituido por tres niveles de madurez y doce prácticas de seguridad, las cuales proporcionan una serie de actividades que las organizaciones pueden implementar para incrementar el aseguramiento del software y minimizar los riesgos de seguridad.

OWASP (Open Web Application Security Project)

"Es una metodología de seguridad de auditoría web, abierta y colaborativa, orientada al análisis de seguridad de aplicaciones Web, y usada como referente en auditorías de seguridad" (Crowe, 2022) para realizar el análisis y la evaluación de los riesgos.

Esta metodología define una revisión de controles, la cual permite a los auditores garantizar que las revisiones de las plataformas se están realizando de manera correcta, es decir que han sido analizados la totalidad de vectores de ataques y se han detectado todos los fallos de seguridad, esta revisión de controles contribuye a mejorar la seguridad y protección de los sistemas de información.

OWASP ASVS (Application Security Verification Standard)

"Proporciona una base para probar los controles técnicos de seguridad de las aplicaciones web y también proporciona a los desarrolladores una lista de requisitos para un desarrollo seguro" (OWASP, 2022).

El principal propósito de la metodología es "normalizar el rango en la cobertura y el nivel de rigor disponible en el mercado cuando se trata de realizar la verificación de seguridad de aplicaciones web utilizando un estándar abierto comercialmente viable" (OWASP, 2022). Este estándar provee una base que permite realizar pruebas a los controles de seguridad técnica de la aplicación y del entorno los cuales se encargan de proteger contra vulnerabilidades.

A continuación, se describe el propósito de utilizar este estándar:

Utilizarlo como métrica con el fin de proveer criterios para medir el nivel de confianza de las aplicaciones a los propietarios y desarrolladores de estas.

Utilizarlo como guía para los desarrolladores que indique que se debe incluir en los controles de seguridad para cumplir con los requerimientos de seguridad de las aplicaciones.

Utilizarlo como una base para detallar los requerimientos de verificación de seguridad.

CbyC

Es una metodología utilizada en desarrollos de software que requieren un alto nivel de seguridad, la cual debe ser verificable. El principal objetivo de esta metodología es minimizar la tasa de defectos y ser resiliente a los cambios, esto se logra teniendo en cuentas los principios fundamentales de la metodología: "que sea muy difícil introducir errores y asegurarse que los errores sean removidos tan pronto hayan sido inyectados" (Brito-Abundis, 2013).

El uso de esta metodología no da lugar para la confusión como si lo hacen otras metodologías o herramientas que se utilizan con más frecuencia cuyos resultados son aleatorios.

Esta metrología se encuentra basada en el desarrollo incremental, en la reducción de tiempos de desarrollo y en evitar la repetición.

TSP-Secure

Esta metodología es un grupo de procesos estructurados los cuales guían las actividades a realizar en cada fase del proceso de desarrollo y muestran cómo conectarlas para construir el producto. Además, esta metodología se basa en el estándar CMMI que ayuda a resolver problemas de productividad del equipo, calidad del producto y permite optimizar tiempos y costos (1Library).

TSP-Secure se centrarse en la seguridad de las aplicaciones de software para ello tiene en cuenta los siguientes puntos (Joshi, 2013):

- Establece que debe planificarse en la implementación del software seguro.
- Está enfocado a que el producto cumpla todos los niveles de calidad en relación con la seguridad del software.
- El equipo de desarrollo debe tener un alto nivel de conocimiento para solucionar inconvenientes que se puedan presentar.

A continuación, se listan las fases del ciclo de vida de TSP (Arciniega):

- Lanzamiento
- Estrategia
- Plan
- Requisitos
- Diseño
- Implementación
- Pruebas

- Postmortem

DevSecOps

En la implementación de la metodología DevSecOps fundamentada en la cultura DevOps hay practicas establecidas que ayudan a automatizar y facilitar las pruebas de seguridad. A continuación, se describen algunas de ellas:

Analizar las vulnerabilidades de código abierto: Permite realizar un análisis de vulnerabilidades, es muy útil cuando en el proyecto se utilizan componentes externos como librerías de terceros, es decir de origen desconocido.

Prueba de seguridad de aplicaciones estáticas: Esta prueba permite lleva a cabo un análisis de código fuente en aplicaciones o producto de software, con el objetivo de identificar vulnerabilidades en el código o codificación débil. Suele emplear herramientas como SpectralOps y Klockwork para realizar este análisis.

Pruebas dinámicas de seguridad de aplicaciones: Esta herramienta permite identificar los problemas de seguridad o componentes del sistema que estén vulnerables por medio de la simulación de un ataque, se considera una prueba de caja negra ya que se realiza cuando el sistema está puesto en marcha.

Sistema de alerta: la función de esta herramienta como su nombre lo indica es generar alertas y notificaciones de problemas de seguridad, o amenazas detectadas en el sistema, los cuales serán informados al equipo de trabajo para que realice las respectivas acciones de mitigación de la manera más oportuna.

Manifiesto DevSecOps: la filosofía en la que se basa esta metodología consiste en trabajar de la mano con el cliente, por lo cual DevSecOps cuenta con un manifiesto de máximas de visión según (Bolullo, 2019):

Seguridad por diseño, en lugar de revisiones de seguridad.

Flexibilidad en la toma de decisiones en lugar de las negativas sistemáticas.

Datos y ciencia de seguridad en lugar de miedo, paranoia e incertidumbre.

Contribución y colaboración abierta en lugar de requerimientos de seguridad exclusivos.

Integrar la seguridad en el proceso de desarrollo en lugar de añadirla posteriormente.

Servicios de seguridad consumibles y transparentes, en lugar de controles de seguridad y burocracia.

Trabajar en un cambio de la cultura por encima de establecer políticas de seguridad.

Apoyarse más en equipos de desarrollo formados en seguridad que en especialistas de Seguridad.

Buenas Prácticas para Desarrollo de Software Seguro

A través del tiempo se han aplicado diferentes procesos y en el proceso de aplicación de estos se han identificado diferentes prácticas que han tenido buenos resultados las cuales se han ido consolidando y alimentando como una base que sirve de apoyo a los procesos de desarrollo seguro.

A continuación, se listan una serie de buenas prácticas las cuales apoyan los procesos de desarrollo seguro:

Pruebas de Código

Durante el ciclo de desarrollo de software y sobre todo al final de este se deben realizar pruebas que permitan encontrar errores y vulnerabilidades por eso es importante diseñar un buen plan de pruebas que se ejecute en cualquier etapa.

Código Comentado

Se trata de eliminar las fracciones de código que han sido comentadas o deshabilitadas para evitar que se generen accidentes en producción que puedan generar errores en el funcionamiento de la aplicación.

Comenta tu Código

“Para facilitar las modificaciones y mantenimiento, pero recuerda, un buen código es aquel que no necesita comentarios. Redacta sentencias simples e intenta elaborar una solución sencilla y corta. Cuanto más corta sea, menos errores se producirán y más fácil será localizarlos y solventarlos” (tiThink, 2018).

Prácticas Criptográficas

En la mayoría de las aplicaciones existentes hoy en día se hace necesario transmitir, almacenar y recuperar información confidencial, sin embargo, para mantener la confidencialidad

e integridad de la información deben implementarse soluciones criptográficas estándar. Un ejemplo donde se puede utilizar la criptografía es para el cifrado de las contraseñas, ya que de no usar un algoritmo de cifrado de contraseñas estas se transmiten en texto plano y pueden ser interceptadas o explotadas por los ciberdelincuentes.

Validación de Entradas

Una de las falencias de seguridad más comunes en las aplicaciones son las pocas validaciones de datos de entradas del cliente o del entorno. Esta falencia expone a las aplicaciones a diferentes vulnerabilidades tales como inyección SQL, ataques al sistema de archivos, ataques Locale/Unicode y desbordamientos de memoria entre otros.

"Hay que garantizar que la aplicación sea robusta contra todas las formas de ingreso de datos, ya sea obtenida del usuario, de la infraestructura, de entidades externas o de sistemas de base de datos" (Junta de Andalucia, 2022).

Estandarizar las Reglas del Desarrollo

En el proceso de desarrollo de una aplicación se debe de establecer la forma de trabajo para el caso del equipo de desarrollo deben definirse la nomenclatura de las funciones, variables, el nombre de los ficheros, atributos, etc. Esta normalización facilita el desarrollo, lectura del código y mantenibilidad.

Sentencias SQL

Se deben utilizar parámetros de vinculación de tipo seguro y evitar que las consultas sean construidas directamente con datos del usuario para evitar que los usuarios finales puedan introducir sentencias SQL las cuales puedan ejecutarse en el servidor debido a la falta de validaciones de entrada.

Para evitar los ataques por medio de inyección SQL se “requiere separar los datos de los comandos y las consultas” (OWASP, 2021).

Utilizar una API segura, que evite el uso de un intérprete por completo y proporcione una interfaz parametrizada o utilizar una herramienta de ORM.

Implemente validaciones de entradas de datos en el servidor, utilizando "listas blancas".

Para cualquier consulta dinámica restante, escape caracteres especiales utilizando la sintaxis de caracteres específica para el intérprete que se trate.

Utilice LIMIT y otros controles SQL dentro de las consultas para evitar la fuga masiva de registros en caso de inyección SQL.

Contenido URL

En este caso la URL de la aplicación no debe contener información como contraseñas, usuarios, direcciones IP, rutas del sistema de archivos para evitar que se revele información importante de la estructura de la aplicación la cual puede ser utilizada por el delincuente para vulnerar la aplicación.

Pentesting

El pentesting o prueba de penetración consiste en ejecutar una serie de ataques a una aplicación, sistema de información o red en un entorno controlado. Es decir, es un “Conjunto de ataques simulados dirigidos a un sistema informático con la finalidad de detectar posibles debilidades o vulnerabilidades” (Incibe, 2019) que puedan colocar en riesgo la seguridad de la infraestructura tecnológica y la información de una organización ya que pueden ser explotadas por un ciberdelincuente.

Para ejecutar adecuadamente las pruebas de penetración es necesario tener en cuenta los siguientes puntos o fases:

Recolección de Datos. Es sin duda la fase más importante del proceso para ejecutar adecuadamente las pruebas de penetración. Como su nombre lo indica, esta fase consiste recolectar toda la información posible del sistema que va a ser auditado para ello se emplean diferentes técnicas (escaneo de puertos, dominios, IP, versiones y servicios; obtención de obtención de metadatos, etc.) y herramientas (Nmap, Dnsmap, Dnsrecon, entre otras).

Análisis de vulnerabilidades. Consiste en analizar la información obtenida y vulnerabilidades identificadas en la fase anterior, con el propósito de definir cuáles son las estrategias de penetración que se van a utilizar, definir los objetivos y cómo se van a lograr.

Explotación**.** Consiste en aprovechar todas las vulnerabilidades y debilidades de seguridad identificadas en la fase anterior para intentar acceder al sistema de información o red. Algunos de los ataques que se realizan en esta fase son: inyección de código, carencia de métodos criptográficos, autenticación, control de errores entre otros, para lo cual se utilizan las siguientes herramientas: PowerSploit, OpenVAS, Nessus entre otras.

Otra manera es utilizar las credenciales previamente obtenidas para acceder a los sistemas.

Post-Explotación. Una vez se ha conseguido acceder a los sistemas de la organización entra en acción esta fase, con el objetivo de demostrar que consecuencias acarrea las brechas de seguridad identificadas y explotadas. Para ello en esta fase se intentará obtener el nivel máximo de privilegios, acceder a información, redes y todos los sistemas o aplicaciones con los cuales cuente la organización. A continuación, se mencionan algunas herramientas en esta fase: Empire, AutoSploit, PHPSploit, Poet, entre otras.

Informes**.** Consiste en entregar el informe con los resultados de las pruebas de penetración al cliente, en el cual se debe incluir toda la información obtenida sobre las

vulnerabilidades y debilidades encontradas, así como recomendaciones de cómo mitigarlas o solucionarlas, por último, se debe incluir también los puntos donde la seguridad esté bien implementada.

Comparación de las Metodologías de Desarrollo Seguro de Software

Tabla 1

Criterios Comunes Entre las Metodologías de Desarrollo Seguro de Software.

Atributo/Metodología	Microsoft SDL	CbyC	OWASP	SAMM	DevSecOps	TSP-S
A quieren va dirigido / Aplicación	Empresas, entornos personales y en internet	Organizaciones y personas	Entornos de internet	Organizaciones de todo tipo	Organizaciones de todo tipo	Empresas, entornos personales y en internet
Abierto	No	No	Si	Si	No	Si
Documentación	Si, a nivel de proceso	No	No	Si	Si	Si
Métodos Formalizados	No	Si	Si	Si	Si	No
Desarrollo por iteraciones	No	Si	No	No	Si	Si
Seguimiento de requerimientos	No	Si	Si	Si	Si	No
Revisión de código	No	Si	Si	Si	Si	Si
Plan de mitigar ataques	Si	No	Si	No	Si	No
Políticas pruebas	Si	No	No	Si	Si	No
Modelo de Gobierno	No	No	No	Si	No	No

Nota. La Tabla 1 presenta una comparación de los criterios comunes entre diversas metodologías de desarrollo seguro de software, permitiendo evaluar qué enfoques son más estructurados y cuáles se alinean mejor con las necesidades de cada organización.

Tabla 2

Características de Metodologías de Desarrollo de Software.

Características	Metodología SDL	Opensamm	OWASP ASVS	CbyC	TSP-Secure
	Se enfoca en la capacitación de personal para que entiendan y apliquen las buenas prácticas de seguridad	Ofrece recursos de capacitación al personal en temas de programación segura y capacita al personal sobre lineamientos específicos de desarrollo seguro para cada rol y obliga al personal a capacitarse en temas de seguridad.	Capacita al equipo de desarrollo de controles de seguridad sobre el tema para lograr satisfacer los requerimientos de seguridad del software		Realiza capacitaciones de temas sobre seguridad
	Define los requisitos de seguridad desde inicio del proyecto y los actualiza según las necesidades de seguridad.	Establece los requisitos de seguridad del software desde la etapa de levantamiento de requisitos de la aplicación.	Proporciona una base para especificar los requisitos de seguridad de la aplicación.	Considera los requerimientos funcionales, no funcionales y los de seguridad del software desde la etapa de inicio.	Define los requerimientos de seguridad
	Sugiere definir los mínimos niveles de seguridad aceptables y asignar las	Establece estrategias para realizar medición al nivel de cumplimiento de	Proporciona un criterio para evaluar el grado de confianza de las aplicaciones.		

Características	Metodología SDL	Opensamm	OWASP ASVS	CbyC	TSP-Secure
	responsabilidades de estos al equipo desde la etapa inicial del proyecto.	seguridad de las aplicaciones.			
	Sugiere plantear el modelo de amenazas el cual ayuda a establecer eficazmente las vulnerabilidades de seguridad y a mitigarlas	Permite identificar, comprender y evaluar con precisión las amenazas para mitigarlas.		Los módulos de la aplicación deben ser sencillos para permitir corregir las posibles fallas del software	Desarrollo de un plan de mitigación de incidentes
	Establece requisitos de diseño de acuerdo con las necesidades de seguridad requeridas.	Asegurar que se incluya en el diseño los lineamientos de seguridad acordes a los requerimientos y amenazas identificadas y conocidas que se puedan presentar.		Después de la definición de requerimientos se encarga de establecer las funcionalidades del software, su arquitectura y los requerimientos no funcionales de protección y seguridad.	

Nota. La Tabla 2 presenta una comparación de las principales características de diversas metodologías de desarrollo de software seguro, destacando su enfoque en capacitación, definición de requisitos de seguridad, evaluación de amenazas y estrategias de mitigación.

Ataques y Causas Más Comunes en los Sistemas o Aplicaciones

En la actualidad los ataques a los sistemas de información y aplicaciones siguen en aumento, los ciberdelincuentes emplean diferentes técnicas y métodos para aprovechar las vulnerabilidades de los sistemas. Según el estudio realizado por Verizon "2022 Data Breach Investigations Report", uno de los ataques más utilizados son los clasificados como ataques de ingeniería social, pues el 82% de los ataques tienen involucrados el factor humano, es decir que este se ha convertido en foco principal de los delincuentes a la hora de planear un ataque, por lo cual las empresas deben contar con programas de concientización de seguridad para prevenir que el personal sea víctima de los ataques de ingeniería social. Los ataques más utilizados de este tipo por los ciberdelincuentes son el phishing, smishing, vishing, whaling, entre otros.

Por otro lado, del grupo de los ataques de malware, el ransomware se ha consolidado como uno de los más utilizados en el año 2022 aumentando un 13% con respecto al 2021.

Adicionalmente, el ataque de denegación de servicios (DoS) se ha consolidado como el líder en los ataques por incidentes aprovechando las vulnerabilidades de un servicio o red.

Ataques De Ingeniería Social

Los ataques de ingeniería social son aquellos donde los ciberdelincuentes intentan persuadir al objetivo por medio de engaños y mentiras empleando diferentes técnicas y herramientas para lograr su objetivo (phishing, smishing, vishing etc.)

A continuación, se describen los ataques de ingeniería social más utilizados hasta el momento.

Phishing

El phishing es una técnica de ataque basada en ingeniería social, donde los atacantes envían mensajes fraudulentos (generalmente correos electrónicos o mensajes de texto) que

aparentan ser de fuentes confiables, como instituciones bancarias o empresas legítimas. Su objetivo principal es engañar a los usuarios para que revelen información confidencial, como contraseñas, números de tarjetas de crédito o datos personales (Proofpoint).

Las características principales del phishing son:

Es una de las formas más comunes de ataque cibernético.

El atacante utiliza un lenguaje convincente y personaliza los mensajes para que parezcan legítimos.

Generalmente incluye un enlace que dirige al usuario a un sitio web falso diseñado para robar información.

Smishing

Es una variante del phishing que utiliza mensajes de texto (SMS) en lugar de correos electrónicos como medio principal de ataque. Basado en técnicas de ingeniería social, este método busca engañar a los usuarios para que realicen acciones específicas, como hacer clic en enlaces maliciosos que redirigen a sitios web fraudulentos, descargar aplicaciones maliciosas que contienen malware (como troyanos) o proporcionar información confidencial, incluyendo credenciales de inicio de sesión, datos bancarios o información personal. Los atacantes suelen enviar mensajes persuasivos a teléfonos inteligentes, indicando a las víctimas que accedan a páginas falsas o descarguen archivos comprometidos con el propósito de obtener sus datos. Este tipo de ataque se ha vuelto más común debido a la creciente dependencia de los dispositivos móviles, lo que lo convierte en una amenaza significativa para la privacidad y la seguridad de los usuarios (Kaspersky).

Fase de Planificación. Consiste en planear la estrategia del ataque, es decir, en esta fase el atacante elegí a sus víctimas, empresa o persona a la cual va a suplantar, qué método va a

utilizar (Web falsa o archivos o app con infectados con algún tipo de malware) y qué información desea obtener, por ejemplo: Credenciales, datos personales o información de cuentas bancarias etc.

Fase de Preparación. En esta fase el atacante elabora el mensaje que servirá para persuadir o engañar a las víctimas, por esto el atacante juega un papel muy importante, ya que depende de su creatividad elaborar un mensaje convincente y que parezca legítimo teniendo en cuenta los datos que desea obtener. Algunos atacantes adquieren dominios para montar los sitios web falsos y para obtener dominios de correo para hacer que el ataque sea más convincente y difícil de detectar.

Fase de Ataque. Esta fase consiste en enviar el mensaje o la víctima o víctimas dependiendo si el ataque es masivo esperando que el mensaje logre su objetivo de persuadir a la víctima para que abra el enlace o descargue un archivo malicioso.

Fase de obtención de datos: Esta fase consiste en obtener los datos de las víctimas que cayeron en la trampa, la tasa de éxito para llegar a esta fase va a depender del método de ataque empleado, generalmente el uso de un sitio web falso tiene mayor probabilidad de éxito que la descarga de malware.

Fase de Ejecución del Fraude. En esta fase el atacante ya consiguió el objetivo, por lo cual procede a ejecutar fraude ya sea para beneficio personal o para vender la información obtenida a terceros para que estos ejecuten el fraude.

Fase de post-ataque: En esta fase el atacante se encarga de eliminar cualquier evidencia del ataque ejecutado.

Vishing

Este tipo de ataque se realiza por medio de llamadas telefónicas en la cual el atacante haciendo usos de la ingeniería social intenta persuadir a la víctima para obtener información relevante para realizar el robo de identidad o información de cuentas bancarias (Joshua, 2024).

Whaling

Este tipo de ataque está dirigido a los altos mandos de las empresas tales como CEOs y CFOs, ya que estos tienen mayor acceso a los datos confidenciales de la empresa, en este ataque se hace uso de la ingeniería social para manipular a la víctima para que autorice o realice transferencias de alto valor (Kaspersky).

Malware

"El software malicioso puede modificar el funcionamiento de un equipo informático o alterar la información que procesa, ya sea borrándose, modificándose o enviándola sin nuestro conocimiento a terceras personas" (Escrivá-Gascó, 2023).

Adware

El adware es un software que genera y muestra automáticamente anuncios publicitarios en los dispositivos de los usuarios, generalmente sin su consentimiento. Su propósito principal es generar ingresos para el creador del adware, ya sea mostrando anuncios directamente o redirigiendo al usuario a sitios web específicos. Aunque no siempre es malicioso, el adware puede convertirse en un problema de seguridad cuando se instala sin el conocimiento del usuario,

recoge datos personales o hábitos de navegación para mostrar anuncios personalizados (práctica que puede violar la privacidad) y reduce el rendimiento del dispositivo al consumir recursos del sistema.

A diferencia de otros tipos de malware, el adware no siempre tiene intenciones maliciosas, pero su comportamiento intrusivo y no deseado puede clasificarse como una amenaza de bajo nivel, especialmente cuando interfiere significativamente con la experiencia del usuario o expone datos sensibles (Latto, 2020).

Mensajes Emergentes

El adware genera anuncios emergentes (pop-ups) constantes mientras el usuario navega por Internet, tales como:

Redirección de tráfico: Puede redirigir al usuario a sitios web promocionales o llenos de publicidad.

Instalación no autorizada: A menudo se distribuye como parte de paquetes de software gratuitos o shareware.

Recopilación de datos: Puede rastrear el comportamiento del usuario en línea para mostrar anuncios dirigidos.

Ransomware

Es un tipo de malware diseñado para bloquear el acceso al sistema o cifrar los archivos de la víctima mediante el uso de algoritmos de cifrado avanzados. Su objetivo principal es extorsionar a la víctima exigiendo un pago (rescate) para restaurar el acceso o descifrar la información comprometida. Este ataque se ha convertido en una de las amenazas más comunes y devastadoras en el ámbito de la ciberseguridad (Kaspersky).

Técnicas de Propagación

Los ciberdelincuentes utilizan diversas técnicas para infectar sistemas con ransomware (Gutiérrez, 2023), entre ellas:

Phishing: Correos electrónicos fraudulentos con enlaces o adjuntos maliciosos.

Archivos infectados: Documentos de Word, Excel o PDF que contienen macros maliciosas.

Software descargado de internet: Programas no confiables o pirateados.

Dispositivos USB: Uso de unidades extraíbles infectadas para propagar el malware.

Troyanos: Combinación con otros tipos de malware que permiten la propagación masiva en redes corporativas.

Tipos de Ransomware

Existen varios tipos de ransomware algunos más peligrosos que otros, pero todos enfocados a pedir dinero a la víctima para devolver el acceso o información secuestrada (Gutiérrez, 2023).

Cifrado de Archivos. Este es el tipo más común y afecta tanto a usuarios individuales como a organizaciones. Cifra archivos clave como documentos, imágenes y videos. Ejemplo: Ryuk, WannaCry.

Bloqueo de Sistema. Bloquea el acceso completo al sistema operativo, impidiendo que el usuario acceda incluso al escritorio. Ejemplo: Reveton.

Scareware. Utiliza ventanas emergentes para engañar a la víctima, alertándola de un supuesto problema. Persuade a la víctima a instalar un "antivirus" falso, que en realidad es malware. Ejemplo: Antivirus 2009.

Consecuencias de los Ataques de Ransomware

Los ataques de ransomware pueden traer consecuencias devastadoras especialmente para las organizaciones entre ellas se destacan las siguientes (Checkpoint):

Interrupción de la actividad empresarial: Los sistemas quedan inaccesibles, deteniendo operaciones críticas y causando pérdidas económicas.

Pérdidas económicas: Los costos incluyen el pago del rescate, la contratación de expertos en ciberseguridad y la recuperación de sistemas afectados.

Pérdida de información clave: Si el ransomware causa un daño irreversible a los archivos, la información puede perderse permanentemente, incluso si se paga el rescate.

Desconfianza en la organización: Los clientes pueden perder confianza en los servicios ofrecidos debido a la exposición de datos confidenciales.

Puerta de entrada para ataques adicionales: El ransomware a menudo deja puertas traseras abiertas, permitiendo ataques posteriores como fuerza bruta o propagación en redes internas.

Recomendaciones para Prevenir Ataques de Ransomware

La primera defensa para prevenir los ciberataques es conocer conceptos básicos de seguridad, pero además es importante seguir las siguientes recomendaciones que, aunque son sencillas se deben muy riguroso en su implementación y aplicación (Checkpoint).

Mantener el software actualizado: Las actualizaciones corrigen vulnerabilidades conocidas en sistemas operativos y programas.

Filtrar y analizar archivos adjuntos: Evitar abrir archivos de remitentes desconocidos y utilizar herramientas de filtrado y análisis en servidores de correo y dispositivos.

Realizar copias de seguridad periódicas: Mantener copias de seguridad actualizadas en ubicaciones seguras y desconectadas de la red principal.

Capacitar a los usuarios: Concientizar a empleados y usuarios sobre las tácticas comunes de ransomware, como phishing, para minimizar errores humanos.

Inyección De Código Malicioso (SQL)

“La inyección de SQL es un tipo de ciberataque encubierto en el cual un hacker inserta código propio en un sitio web con el fin de quebrantar las medidas de seguridad y acceder a

datos protegidos. Una vez dentro, puede controlar la base de datos del sitio web y secuestrar la información de los usuarios" (Belcic, 2020).

Ataque de Denegación de Servicio o Ataque DOS

Un ataque de Denegación de Servicio (DoS) es un tipo de ataque cibernético diseñado para interrumpir la disponibilidad de un recurso o servicio. Se logra sobrecargando un servidor, dispositivo o red, haciendo que los usuarios legítimos no puedan acceder temporal o permanentemente al recurso (Cloudflare).

- Características principales

Degradación del servicio: Se provoca al inundar el sistema objetivo con solicitudes masivas o procesos maliciosos, consumiendo todos los recursos disponibles (Ciberinseguro, 2023).

Interrupción del acceso: El objetivo principal es impedir que los usuarios legítimos accedan al recurso afectado (Ciberinseguro, 2023).

- Clasificación de ataques DoS

Ataques internos: Llevados a cabo por empleados o usuarios de la misma organización. Estos pueden ser intencionados (venganza, robo de información) o accidentales (errores humanos que provocan sobrecargas de sistemas) (Axur).

Ataques externos: Realizados por actores externos que aprovechan vulnerabilidades en los sistemas para interrumpir el servicio. Estos pueden incluir el uso de botnets para aumentar la escala del ataque (Axur).

Causas de que un Ataque Cibernético sea Exitoso

Vulnerabilidad de los sistemas informáticos: Son todas las debilidades de un sistema, las cuales puede utilizar un atacante para para comprometer los 3 principios de la seguridad de la

información (Confidencialidad, integridad y disponibilidad) como también la infraestructura del sistema, estas vulnerabilidades muchas veces son causadas por errores o fallos en el diseño e implementación del proyecto, como por ejemplo falta de inclusión de pruebas de penetración (Pentesting) en proyecto que debido a su forma lo requiere (Ambit-bst, 2020).

https://www.piranirisk.com/es/blog/vulnerabilidades-en-seguridad-de-la-informacion

Ingeniería social: Se trata de un método que utilizan los delincuentes para engañar a víctimas potenciales y obtener información por medio de la creación de lazos de confianza, los datos obtenidos van desde datos personales (teléfonos, direcciones de correo), de cuentas (correos, rede sociales), financieros (número de cuentas bancarias, etc.) (Kaspersky).

Pérdida y robos de dispositivos electrónicos: Se considera este una causa de los ataques cibernéticos, ya que, los objetos perdidos o robados pueden caer en manos de delincuentes y la mayoría de las veces estos dispositivos como portátiles, tabletas, celulares y USB tienen información valiosa tales como usuarios, clientes, proveedores, proyectos, información de cuentas bancarias entre otros (Kaspersky). Esta información que puede ser utilizada por los ciberdelincuentes para obtener beneficios monetarios, ya que por medio de ella pueden realizar diferentes tipos de ataques como phishing, smishing entre otros.

Brechas de seguridad en terceros: Las organizaciones que dependen de proveedores o socios externos con controles de seguridad deficientes están en riesgo de ser afectadas indirectamente por ataques dirigidos contra estos terceros (Arnal, 2024).

Empleados con malas intenciones: Trabajadores descontentos o con acceso privilegiado pueden intencionalmente comprometer la seguridad del sistema, filtrando datos o degradando servicios (Proofpoint).

Falta de controles de acceso y privilegios: La falta de controles adecuados en el acceso y la gestión de privilegios representa un riesgo significativo para la seguridad de las organizaciones. En muchos casos, no se realiza un control exhaustivo de los usuarios, roles y permisos asignados, lo que puede llevar a que un atacante interno, con acceso a una cuenta mal configurada o con privilegios indebidos, provoque fugas masivas de información o realice acciones maliciosas dentro del sistema. Además, la ausencia de una política clara para desactivar usuarios inactivos o eliminar cuentas de empleados que ya no forman parte de la entidad puede permitir que personas no autorizadas sigan accediendo a los sistemas, comprometiendo la integridad y confidencialidad de la información organizacional (VPN Unlimited).

Negligencia por parte de los usuarios a la hora de instalar y mantener actualizados el antivirus y programa anti-spyware residentes (Zapata, 2022).

Divulgación accidental. La divulgación accidental puede facilitar ataques cibernéticos adicionales, ya que la información expuesta puede ser utilizada por actores maliciosos para explotar vulnerabilidades existentes en los sistemas o para llevar a cabo técnicas de ingeniería social, ya que más de un 32% de las incidencias tienen que ver con la divulgación no intencionada de información por parte de los empleados (Sevillano, 2017).

Definir los Factores de Riesgo más Frecuentes en las Aplicaciones o Sistemas por Medio de una Revisión Bibliográfica que Permita Identificar las Brechas de Seguridad más Comunes.

El objetivo de este capítulo es definir los factores de riesgo más frecuentes en las aplicaciones o sistemas mediante una revisión bibliográfica exhaustiva, con el fin de identificar las brechas de seguridad más comunes. En el contexto del desarrollo de software, la identificación y mitigación de riesgos de seguridad es fundamental para prevenir ataques y proteger la integridad, confidencialidad y disponibilidad de los datos.

Las aplicaciones modernas son cada vez más complejas y están expuestas a una variedad de amenazas debido a las diversas tecnologías y plataformas en las que se desarrollan e implementan. Factores como la autenticación inadecuada, las vulnerabilidades de inyección, la exposición de datos sensibles, la configuración insegura, la validación insuficiente de entradas y el uso de componentes con vulnerabilidades conocidas, se destacan como algunos de los riesgos más críticos en la seguridad del software.

Este capítulo se basa en la revisión de literatura académica y reportes de la industria para proporcionar una visión detallada de estos factores de riesgo. Al identificar y analizar las brechas de seguridad más comunes, se pretende ofrecer una base sólida que permita a los desarrolladores y profesionales de seguridad implementar estrategias efectivas para mitigar estos riesgos y mejorar la seguridad general de las aplicaciones.

Mediante el análisis de estudios recientes y prácticas recomendadas, este capítulo no solo destaca las vulnerabilidades más relevantes, sino que también enfatiza la importancia de una postura de seguridad proactiva y multifacética. La implementación de prácticas robustas de seguridad desde las etapas iniciales del desarrollo hasta el mantenimiento continuo es esencial

para reducir los riesgos y garantizar la resiliencia de las aplicaciones frente a amenazas emergentes.

Fallos en la Autenticación y la Gestión de Sesiones

Uno de los factores de riesgo más frecuentes en las aplicaciones es la implementación deficiente de los mecanismos de autenticación y gestión de sesiones. Según (OWASP, 2021), las vulnerabilidades en esta área pueden permitir que atacantes comprometan credenciales de usuario, asuman identidades de usuarios válidos y mantengan sesiones activas de manera indebida. Un estudio de Fett, Kerschbaum y Schwenk (2019) resalta que la falta de prácticas seguras en la gestión de tokens de sesión y contraseñas es una de las principales causas de estas vulnerabilidades.

La autenticación y la gestión de sesiones son componentes críticos de la seguridad de las aplicaciones. Fallos en estos procesos pueden comprometer gravemente la seguridad de una aplicación, permitiendo a atacantes eludir controles de acceso, robar identidades de usuarios y mantener acceso no autorizado a recursos sensibles. Este apartado analiza las causas comunes de estos fallos y proporciona ejemplos y recomendaciones basadas en la literatura existente.

Causas Comunes de Fallos en la Autenticación

Contraseñas Débiles y Reutilización de Contraseñas. Una de las causas más frecuentes de fallos en la autenticación es el uso de contraseñas débiles o la reutilización de contraseñas en múltiples servicios. Según un estudio de (Bonneau, 2012), los usuarios tienden a elegir contraseñas que son fáciles de recordar, pero también fáciles de adivinar, y a menudo reutilizan las mismas contraseñas en diferentes sitios web, aumentando el riesgo de compromisos masivos.

Almacenamiento Inseguro de Credenciales. Muchas aplicaciones fallan al no proteger adecuadamente las credenciales de los usuarios. Según (OWASP, 2021), el almacenamiento de

contraseñas en texto plano o el uso de algoritmos de hash inseguros son prácticas comunes que pueden ser fácilmente explotadas por atacantes.

Implementación Deficiente de Mecanismos de Autenticación Multifactor (MFA). Aunque MFA es una medida de seguridad eficaz, su implementación incorrecta puede crear una falsa sensación de seguridad. Los errores en la implementación de MFA, como la falta de protección contra el robo de tokens, pueden dejar a las aplicaciones vulnerables a ataques. (Mantovani, 2019)

Causas Comunes de Fallos en la Gestión de Sesiones

Tokens de Sesión Inseguros. Los tokens de sesión que no se generan o gestionan correctamente pueden ser interceptados y utilizados por atacantes para suplantar a usuarios legítimos. (OWASP, 2021) destaca que la falta de cifrado de tokens de sesión y la reutilización de tokens antiguos son prácticas inseguras comunes.

Duración de Sesión Inadecuada. Configurar una duración de sesión demasiado larga puede aumentar el riesgo de que un token de sesión sea utilizado por un atacante. Por otro lado, sesiones demasiado cortas pueden afectar negativamente la experiencia del usuario. Un equilibrio adecuado es esencial para la seguridad y la usabilidad.

Falta de Mecanismos de Revocación de Sesión. La ausencia de mecanismos para invalidar tokens de sesión activos cuando se detecta actividad sospechosa o cuando un usuario cierra sesión puede permitir a los atacantes mantener acceso no autorizado. La revocación de sesión es crucial para limitar el tiempo de exposición a ataques. (Jiménez & César , s.f.)

Ejemplos de Ataques Comunes

Ataques de Fuerza Bruta y Ataques de Diccionario. Los atacantes intentan adivinar contraseñas mediante la prueba sistemática de combinaciones posibles. El uso de contraseñas débiles facilita estos ataques.

Secuestro de Sesión (Session Hijacking). Este ataque ocurre cuando un atacante intercepta un token de sesión válido y lo utiliza para acceder a la cuenta de un usuario. El uso de cifrado y mecanismos seguros de manejo de tokens puede mitigar este riesgo.

Fijación de Sesión (Session Fixation). En este tipo de ataque, el atacante establece una sesión conocida que el usuario legítimo luego utiliza, permitiendo al atacante secuestrar la sesión. La regeneración de tokens de sesión después de la autenticación puede prevenir este tipo de ataque (OWASP, 2021).

Recomendaciones

Implementación de Contraseñas Fuertes y Políticas de MFA. Las aplicaciones deben exigir contraseñas complejas y únicas y fomentar el uso de autenticación multifactor. Según (Bonneau, 2012) las políticas de contraseña robustas y el uso de MFA pueden reducir significativamente el riesgo de compromisos de autenticación.

Protección de Credenciales de Usuario: Almacenar contraseñas utilizando algoritmos de hash seguros, como bcrypt, y proteger tokens de sesión con cifrado adecuado son prácticas recomendadas para asegurar las credenciales de los usuarios (OWASP, 2021).

Gestión segura de Sesiones. Utilizar tokens de sesión generados de manera segura, implementar límites de duración de sesión razonables y mecanismos de revocación de sesión robustos son esenciales para la seguridad de las sesiones.

Inyecciones

Las inyecciones representan una de las vulnerabilidades más críticas y comunes en el desarrollo de software. Estas vulnerabilidades permiten a los atacantes insertar código malicioso en las aplicaciones a través de entradas no validadas, comprometiendo la seguridad y funcionamiento del sistema. En este apartado, se examinan las causas, ejemplos y contramedidas de las inyecciones, con énfasis en inyecciones SQL y de comandos, proporcionando una base sólida para comprender y mitigar estos riesgos.

Causas Comunes de Inyecciones

Falta de Validación y Sanitización de Entradas. La principal causa de las inyecciones es la insuficiente validación y sanitización de los datos de entrada. Según (Halfond, Viegas, & Orso, 2006), muchas aplicaciones no verifican adecuadamente las entradas del usuario, permitiendo que datos maliciosos se procesen sin restricción.

Construcción Dinámica de Consultas SQL. Las consultas SQL construidas dinámicamente mediante la concatenación de entradas del usuario son especialmente vulnerables a inyecciones. Cuando las entradas no se escapan correctamente, los atacantes pueden manipular la consulta para ejecutar comandos no autorizados (Halfond, Viegas, & Orso, 2006).

Falta de Uso de Parámetros en Consultas. No utilizar parámetros o declaraciones preparadas en las consultas SQL aumenta el riesgo de inyecciones. Según (Wassermann & Su), el uso de consultas parametrizadas puede prevenir la ejecución de código no autorizado.

Ejemplos de Inyecciones

Inyección SQL (SQL Injection): Este tipo de ataque permite a los atacantes manipular consultas SQL para ejecutar comandos no deseados. Por ejemplo, una entrada maliciosa como '

OR '1'='1 en un campo de login podría convertir una consulta en SELECT * FROM users WHERE username = '' OR '1'='1', devolviendo todos los registros de la base de datos.

Inyección de comandos (Command Injection): Los atacantes insertan comandos maliciosos en un sistema operativo a través de una aplicación vulnerable. Por ejemplo, en un script de Shell que utiliza entradas de usuario sin validación adecuada, un atacante podría incluir: rm -rf / para eliminar archivos del sistema.

Contramedidas para Prevenir Inyecciones

Validación y Sanitización de Entradas. Es fundamental validar y sanitizar todas las entradas del usuario para asegurar que solo se acepten datos esperados y seguros. (Halfond, Viegas, & Orso, 2006) sugieren implementar filtros estrictos y reglas de validación para cada campo de entrada.

Uso de Consultas Parametrizadas. Las consultas parametrizadas y las declaraciones preparadas son efectivas para prevenir inyecciones SQL. (Wassermann & Su) recomiendan el uso de este enfoque para todas las consultas SQL, asegurando que los datos del usuario se traten como parámetros y no como parte del código de consulta.

Utilización de ORM (Object-Relational Mapping). Los ORM ayudan a abstraer el acceso a la base de datos y pueden mitigar el riesgo de inyecciones al manejar automáticamente la parametrización de consultas. Los ORM pueden proporcionar una capa adicional de seguridad al gestionar las interacciones con la base de datos de manera más segura. (AltexSoft Inc, 2024)

Implementación de WAF (Web Application Firewall). Un WAF puede ayudar a detectar y bloquear intentos de inyección al analizar las solicitudes entrantes y aplicar reglas de seguridad. (OWASP, 2021) recomienda el uso de WAF como una medida adicional para proteger las aplicaciones web de ataques.

Exposición de Datos Sensibles

La exposición de datos sensibles se refiere a la divulgación no autorizada de información personal, financiera o confidencial que puede llevar a consecuencias graves, como el robo de identidad, fraude financiero y pérdida de confianza de los clientes. Este tipo de vulnerabilidad ocurre cuando una aplicación no maneja adecuadamente la protección de datos críticos. En este apartado, se analizarán las causas comunes, ejemplos y medidas preventivas para evitar la exposición de datos sensibles, basándonos en la literatura y prácticas recomendadas.

Causas Comunes de la Exposición de Datos Sensibles

Almacenamiento inseguro de datos: Una causa frecuente es el almacenamiento de datos sensibles sin cifrado o utilizando métodos de cifrado inseguros. Según (OWASP, 2021), muchas aplicaciones almacenan datos críticos como contraseñas, números de tarjetas de crédito y datos personales en texto plano o utilizando algoritmos de cifrado obsoletos.

Transmisión insegura de datos**:** La falta de cifrado durante la transmisión de datos entre el cliente y el servidor puede permitir a los atacantes interceptar y acceder a información sensible. El uso de protocolos inseguros como HTTP en lugar de HTTPS es una práctica común que expone los datos a riesgos significativos (Ristic, 2014).

Falta de control de acceso: La implementación deficiente de controles de acceso puede permitir que usuarios no autorizados accedan a datos sensibles. Según (Parra, 2023), las fallas en la gestión de permisos y roles pueden resultar en la exposición involuntaria de información crítica.

Ejemplos de Exposición de Datos Sensibles

Filtración de datos personales: Un ejemplo clásico es la filtración de datos personales a través de brechas de seguridad en aplicaciones web. Incidentes como la violación de datos de

Equifax en 2017, donde se expusieron nombres, fechas de nacimiento y números de seguro social de millones de personas, destacan la gravedad de este problema (Fruhlinger, 2020).

Interceptación de datos de pago: La transmisión de información de tarjetas de crédito sin cifrar puede ser interceptada por atacantes. Los certificados SSL/TLS son esenciales para proteger las aplicaciones, pero muchas aplicaciones no implementan adecuadamente el cifrado SSL/TLS, exponiendo datos financieros sensibles. (Bocetta, 2020)

Medidas Preventivas para Evitar la Exposición de Datos Sensibles

Cifrado de Datos en Reposo y en Tránsito. Es fundamental cifrar los datos sensibles tanto en reposo como durante su transmisión. (OWASP, 2021) recomienda el uso de algoritmos de cifrado modernos y seguros, como AES para el cifrado de datos en reposo y TLS para la protección de datos en tránsito.

Implementación de Controles de Acceso Rigurosos. Las aplicaciones deben contar con controles de acceso bien definidos y aplicados estrictamente para asegurar que solo los usuarios autorizados puedan acceder a datos sensibles. (Yoris, 2023) sugiere la utilización de modelos de control de acceso basados en roles (RBAC) para gestionar los permisos de manera eficiente.

Revisión y Monitoreo de Seguridad. Realizar auditorías de seguridad periódicas y monitorear continuamente la infraestructura de TI para detectar y responder a posibles vulnerabilidades es crucial. De acuerdo con (Moes, 2023), las auditorías regulares y el monitoreo proactivo pueden identificar y mitigar riesgos antes de que resulten en brechas de datos.

Seguridad en el desarrollo y ciclo de vida del software (SDLC). Integrar prácticas de seguridad desde el inicio del ciclo de vida del desarrollo del software, como el uso de pruebas de seguridad automatizadas y revisiones de código, puede prevenir la exposición de datos sensibles.

(OWASP, 2021) enfatiza la importancia de adoptar un enfoque de desarrollo seguro, incorporando la seguridad en cada fase del SDLC.

Configuración Insegura

La configuración insegura se refiere a la mala gestión de los ajustes de seguridad de las aplicaciones, servidores, bases de datos y otras infraestructuras. Esto puede resultar en vulnerabilidades explotables por atacantes. Este apartado analiza las causas, ejemplos y estrategias para prevenir configuraciones inseguras, basándose en literatura y mejores prácticas.

Causas Comunes de Configuración Insegura

Configuraciones predeterminadas inseguras: Muchas aplicaciones y sistemas se instalan con configuraciones predeterminadas que no son seguras. Según (OWASP, 2021), los valores predeterminados suelen priorizar la facilidad de uso sobre la seguridad, dejando puertas abiertas para ataques.

Falta de actualizaciones y parches**:** No mantener el software y sistemas actualizados con los últimos parches de seguridad es una causa frecuente de configuraciones inseguras. (IBM, 2021) sugiere que muchas brechas de seguridad ocurren debido a la explotación de vulnerabilidades conocidas que no han sido parcheadas.

Configuración inadecuada de permisos: La asignación incorrecta de permisos y roles puede dar lugar a configuraciones inseguras. La falta de controles de acceso adecuados y la sobrecarga de privilegios son problemas comunes que resultan en vulnerabilidades.

Ejemplos de Configuración Insegura

Aplicaciones web sin seguridad adecuada: Un ejemplo típico es la falta de configuración segura en servidores web. Los servidores Apache o Nginx, por ejemplo, pueden venir con

módulos y características habilitadas que no son necesarias y que pueden ser explotadas si no se desactivan (Ristic, 2014).

Bases de datos expuestas: Bases de datos configuradas sin restricciones adecuadas pueden ser accesibles desde cualquier lugar en la red, permitiendo a los atacantes acceder y exfiltrar datos. Un ejemplo notable es la brecha de datos de MongoDB en 2017, donde miles de bases de datos fueron expuestas debido a configuraciones predeterminadas inseguras (Fruhlinger, 2020).

Servicios en la nube mal configurados: Configuraciones incorrectas en servicios en la nube pueden exponer datos y sistemas a ataques. Según (Chaloupka, 2024) las configuraciones de seguridad inadecuadas en servicios de almacenamiento en la nube son una causa común de violaciones de datos.

Estrategias para Prevenir Configuraciones Inseguras

Revisión y ajuste de configuraciones predeterminadas: Revisar y ajustar las configuraciones predeterminadas de todas las aplicaciones y sistemas para alinearlas con las políticas de seguridad de la organización. (OWASP, 2021) recomienda desactivar módulos y características innecesarias y configurar adecuadamente los permisos y accesos.

Actualización y aplicación de parches regulares: Mantener todos los sistemas y aplicaciones actualizados con los últimos parches de seguridad. (Spiros Alexiou, Ph.D., CISA, CSX-F, & CIA, 2019) enfatiza la importancia de un programa de gestión de parches robusto para mitigar las vulnerabilidades conocidas.

Implementación de configuraciones seguras por defecto: Configurar los sistemas para que sean seguros desde el inicio, aplicando principios de mínima exposición y privilegio. La

implementación de configuraciones seguras por defecto puede prevenir muchas vulnerabilidades relacionadas con configuraciones inseguras.

Auditorías de configuración y escaneo de vulnerabilidades: Realizar auditorías regulares de las configuraciones y escaneos de vulnerabilidades para identificar y corregir configuraciones inseguras. (Lozano, 2024) sugiere el uso de herramientas automatizadas para detectar configuraciones inseguras en tiempo real y tomar acciones correctivas.

Capacitación en seguridad: Asegurar que los administradores y desarrolladores estén bien capacitados en las mejores prácticas de configuración de seguridad. (OWASP, 2021) destaca la importancia de la capacitación continua para mantener la conciencia y competencia en seguridad entre el personal técnico.

Fallos en la Validación y Sanitización de Entradas

Los fallos en la validación y sanitización de entradas representan una de las vulnerabilidades más críticas en el desarrollo de software. Estos fallos ocurren cuando las aplicaciones no verifican o limpian adecuadamente los datos proporcionados por los usuarios antes de procesarlos, lo que puede dar lugar a diversos tipos de ataques, como inyecciones SQL, cross-site scripting (XSS) y otros. Este apartado explora las causas, ejemplos y estrategias para prevenir estos fallos, basándose en literatura y mejores prácticas.

Causas Comunes de Fallos en la Validación y Sanitización de Entradas

Falta de validación en el lado del servidor: Confiar únicamente en la validación del lado del cliente es un error común, ya que los atacantes pueden eludir fácilmente estas validaciones. Según (OWASP, 2021), la validación debe realizarse en el servidor para garantizar su efectividad.

Ausencia de sanitización de entradas: No limpiar adecuadamente los datos de entrada puede permitir que datos maliciosos se procesen y ejecuten en el sistema. (Rubin, 2024) señala que la falta de sanitización de entradas es una causa primaria de ataques como inyecciones SQL y XSS.

Uso de métodos de validación y sanitización inadecuados: Implementar métodos de validación y sanitización que no cubren todos los casos posibles puede dejar brechas de seguridad. Los desarrolladores a menudo subestiman la complejidad de la sanitización adecuada, lo que resulta en fallos de seguridad.

Ejemplos de Fallos en la Validación y Sanitización de Entradas

Inyecciones SQL: Una aplicación que no valida ni sanitiza adecuadamente las entradas del usuario puede ser vulnerable a inyecciones SQL, donde un atacante inserta comandos SQL maliciosos para manipular la base de datos. La brecha de datos de 2019 en el sitio web de British Airways fue atribuida a una vulnerabilidad de inyección (CNN Chile, 2019).

Cross-Site Scripting (XSS): Cuando las entradas del usuario no son sanitizadas, los atacantes pueden inyectar scripts maliciosos en las páginas web, que luego se ejecutan en los navegadores de otros usuarios. XSS es una de las vulnerabilidades más comunes y peligrosas en las aplicaciones web.

Inyección de comandos: La falta de validación y sanitización puede permitir que los atacantes inyecten comandos arbitrarios en aplicaciones que ejecutan comandos del sistema. Este tipo de ataque es especialmente peligroso en aplicaciones que interactúan con el sistema operativo subyacente (Peikar & Vahidnia, 2021).

Estrategias para Prevenir Fallos en la Validación y Sanitización de Entradas

Validación del lado del servidor: Implementar validaciones robustas del lado del servidor para asegurar que los datos de entrada cumplan con los requisitos esperados. (OWASP, 2021) recomienda la validación estricta de todos los datos recibidos del usuario, incluyendo tipo de datos, formato y longitud.

Sanitización de entradas: Limpiar todas las entradas de usuario para eliminar o neutralizar cualquier dato potencialmente malicioso.

Principio de lista blanca (Whitelist): Adoptar el enfoque de lista blanca, donde solo se permiten datos que coincidan con un conjunto específico de criterios aceptables. Este enfoque es más seguro que la lista negra, que intenta bloquear datos conocidos como maliciosos pero puede fallar ante nuevos tipos de ataques. (Grupo Atico34, s.f.)

Capacitación de desarrolladores: Capacitar a los desarrolladores en prácticas seguras de codificación, enfatizando la importancia de la validación y sanitización de entradas. (OWASP, 2021) destaca la formación continua como una medida crucial para reducir los errores de validación y sanitización en el desarrollo de software.

Uso de frameworks y bibliotecas seguras: Utilizar frameworks y bibliotecas de desarrollo que incorporen prácticas de seguridad robustas para la validación y sanitización de entradas.

Uso de Componentes con Vulnerabilidades Conocidas

El uso de componentes con vulnerabilidades conocidas es una de las amenazas más significativas en el desarrollo de software moderno. Estos componentes incluyen bibliotecas, frameworks y otros módulos que pueden tener fallos de seguridad previamente identificados y documentados. La dependencia excesiva de terceros sin una adecuada evaluación de seguridad puede exponer a las aplicaciones a riesgos graves. Este apartado examina las causas, ejemplos y

estrategias de mitigación para manejar el uso de componentes vulnerables, apoyándose en literatura académica y mejores prácticas de la industria.

Causas Comunes del Uso de Componentes con Vulnerabilidades Conocidas

Falta de monitoreo y actualización: Las organizaciones a menudo no mantienen un seguimiento adecuado de los componentes utilizados en sus aplicaciones ni los actualizan regularmente.

Dependencia de repositorios públicos: Utilizar repositorios públicos para obtener componentes puede ser arriesgado, ya que algunos pueden contener vulnerabilidades.

Falta de pruebas de seguridad: La omisión de pruebas de seguridad específicas para componentes de terceros puede permitir que vulnerabilidades conocidas permanezcan sin detectar.

Ejemplos de Fallos por Uso de Componentes con Vulnerabilidades Conocidas

Struts2 vulnerabilidad (CVE-2017-5638): La vulnerabilidad en Apache Struts2 permitió la ejecución remota de código. Equifax sufrió una brecha de datos masiva en 2017 debido a la explotación de esta vulnerabilidad, afectando a aproximadamente 147 millones de consumidores (Fruhlinger, 2020).

Heartbleed (CVE-2014-0160): Esta vulnerabilidad en la biblioteca de criptografía OpenSSL permitió la extracción de información sensible de la memoria de los servidores. Heartbleed afectó a millones de sitios web y expuso datos críticos durante varios años antes de ser descubierta (NVD, 2014)

Deserialización de Java (CVE-2015-4852): Esta vulnerabilidad permitió la ejecución de código malicioso mediante la deserialización de objetos Java. Varias aplicaciones empresariales,

incluidos servidores de aplicaciones como WebLogic, fueron afectados, causando importantes riesgos de seguridad (NVD, 2015)

Estrategias para Prevenir el Uso de Componentes con Vulnerabilidades Conocidas

Inventario y monitoreo continuo: Mantener un inventario actualizado de todos los componentes utilizados y monitorear continuamente las vulnerabilidades reportadas. (OWASP, 2021) recomienda el uso de herramientas de análisis de composición de software (SCA) para automatizar este proceso.

Aplicación de parches y actualizaciones: Implementar un proceso riguroso para aplicar parches y actualizaciones tan pronto como estén disponibles.

Evaluación de seguridad de componentes: Realizar evaluaciones de seguridad exhaustivas de los componentes antes de su integración.

Preferencia por componentes con soporte activo: Seleccionar componentes que tengan un soporte activo y una comunidad comprometida, lo que garantiza que las vulnerabilidades se aborden rápidamente.

Capacitación en seguridad para desarrolladores: Educar a los desarrolladores sobre los riesgos asociados con el uso de componentes de terceros y las mejores prácticas para manejarlos. (OWASP, 2021) enfatiza la necesidad de una formación continua para asegurar que los desarrolladores sean conscientes de los riesgos y sepan cómo mitigarlos.

Conclusión

La identificación de estos factores de riesgo más frecuentes en aplicaciones y sistemas revela la necesidad de adoptar una postura proactiva y multifacética en la seguridad del desarrollo de software. Las organizaciones deben implementar prácticas de seguridad robustas y mantenerse actualizadas con las últimas amenazas y vulnerabilidades para mitigar los riesgos

identificados. La integración de metodologías como DevSecOps, SDL y OpenSAMM puede ser crucial para lograr una reducción significativa de vulnerabilidades en el software.

Listado de Buenas Prácticas y Metodologías para el Desarrollo Seguro de Aplicaciones

En los capítulos anteriores, se han identificado y evaluado diversas metodologías y prácticas de seguridad, resaltando sus ventajas, desventajas y contextos de uso. Además, se han examinado los tipos de ataques comunes y las brechas de seguridad más frecuentes que resultan de malas prácticas en el desarrollo de software. En base a estos conocimientos se hace posible proponer un conjunto de recomendaciones que aborden de manera efectiva las vulnerabilidades identificadas y mejoren el nivel de seguridad de las aplicaciones desarrolladas.

A continuación, se relaciona la propuesta de buenas prácticas y metodologías se centra en varios principios clave:

Buenas Prácticas

Integración Temprana y Continua de la Seguridad

La integración temprana y continua de la seguridad, también conocida como "Security by Design", es un principio fundamental en el desarrollo seguro de aplicaciones. Este enfoque sostiene que la seguridad no debe ser una consideración posterior ni un complemento añadido en las etapas finales del desarrollo, sino un componente integral desde la concepción del proyecto hasta su despliegue y mantenimiento. Incorporar la seguridad desde el inicio permite identificar y mitigar vulnerabilidades potenciales en las primeras fases, reduciendo tanto el riesgo como los costos asociados con la corrección de problemas de seguridad en etapas posteriores.

Planificación y Requisitos de Seguridad.

Identificar requisitos de seguridad**:** Es de suma importancia realizar la identificación y documentación de los requisitos de seguridad de la aplicación incluyendo la evaluación de los riesgos y amenazas potenciales, como también el establecer los controles de seguridad necesarios.

Establecer políticas de seguridad: Se debe establecer políticas de seguridad claras y aplicables que estén acordes a los estándares de la industria y de las mejores prácticas y que sean una guía para el desarrollo de software.

Diseño Seguro.

Modelado de amenazas: con base en el diseño de la aplicación se debe hacer un análisis de modelado de amenazas que permita identificar las vulnerabilidades y de este y sus posibles vectores de ataque

Principio de menor privilegio: Realizar el diseño de los sistemas y aplicaciones utilizando el principio de menor privilegio con el propósito de garantizar que cada uno de los componentes y usuarios solo cuente con los permisos relacionados a sus funciones.

Codificación Segura.

Guías de codificación segura: Implementar guías y estándares de codificación segura para los desarrolladores que incluyan prácticas como la validación y sanitización de entradas, el manejo seguro de errores, y la gestión adecuada de la memoria. (Owasp, s.f.)

Revisiones de código: Definir el proceso que se debe realizar para las revisiones de código sistemáticas, el cual debe verificar que el código cumpla y este conforme a los estándares, prácticas y directrices establecidas para asegurar que el desarrollado sea seguro.

Pruebas de Seguridad.

Pruebas de penetración: Ejecutar pruebas de penetración con el objetivo de identificar y explotar las posibles vulnerabilidades que puede encontrar y aprovechar un atacante.

Análisis estático y dinámico: Hacer uso de herramientas de análisis estático y dinámico con el objetivo de detectar vulnerabilidades en el código fuente y en la aplicación en ejecución.

Despliegue y Operaciones Seguras.

Revisión de seguridad pre-despliegue: Se sugiere realizar una revisión de seguridad exhaustiva antes del despliegue de la aplicación con el fin de asegurar que las vulnerabilidades han sido mitigadas.

Monitorización continua: Implementar monitorización continua de la seguridad de la aplicación en producción para detectar y responder a incidentes de seguridad en tiempo real.

Mantenimiento y Actualizaciones.

Gestión de parches y actualizaciones: Establecer un proceso detallado para aplicar parches y actualizaciones de seguridad que incluya la gestión de dependencias y librerías externa, en este se debe indicar la periodicidad de realización y asegurarse que el proceso se ejecute correctamente.

Auditorías de seguridad regulares: Realizar auditorías de seguridad periódicamente que permitan evaluar la efectividad de los controles implementados hasta el momento y realizar ajustes en caso de ser necesario.

Automatización de Procesos de Seguridad

La automatización de los procesos de seguridad en el desarrollo de software es un enfoque esencial para identificar y mitigar vulnerabilidades de manera eficiente y oportuna. En un entorno de desarrollo ágil y DevOps, donde la velocidad y la frecuencia de las entregas de software son altas, confiar exclusivamente en procesos manuales para la seguridad es impráctico y propenso a errores. La implementación de herramientas y técnicas automatizadas permite a las organizaciones integrar la seguridad en cada etapa del ciclo de vida del desarrollo de software (SDLC), garantizando que las aplicaciones sean seguras desde su concepción hasta su despliegue

Implementación de la Automatización de Procesos de Seguridad.

Análisis estático de seguridad (SAST): Es una técnica de prueba de seguridad que analiza el código fuente de una aplicación en busca de vulnerabilidades sin ejecutarla. Este análisis se realiza en las primeras etapas del desarrollo, permitiendo la identificación temprana de problemas. (Mallón, Pruebas de seguridad de aplicaciones estáticas (SAST), 2024)

Herramientas comunes: SonarQube, Checkmarx, Fortify Static Code Analyzer.

Análisis dinámico de seguridad (DAST): Es una técnica que prueba la aplicación mientras se ejecuta, simulando ataques desde una perspectiva externa para identificar vulnerabilidades en tiempo de ejecución. (Mallón, 2024)

Herramientas Comunes: OWASP ZAP, Burp Suite, Acunetix.

Análisis de composición de software (SCA): Se utiliza para identificar componentes de software de terceros y librerías abiertas utilizadas en una aplicación, evaluando su seguridad y licencias.

Herramientas comunes: WhiteSource, Black Duck, Snyk.

Integración de seguridad en CI/CD: Implementar CI/CD práctica de integración continua que permite la integrar código fuente a un repositorio de forma automática y periódica y también la realización de pruebas automatizadas que contribuyen a garantizar que las integraciones de código sean confiables durante todo el ciclo de vida del desarrollo. (RedHat, 2022).

Herramientas comunes: Jenkins, GitLab CI, CircleCI con plugins de seguridad.

Pruebas de penetración automatizadas: Realizar pruebas de penetración automatizadas con el fin de ayudar a los equipos de seguridad a descubrir vulnerabilidades de seguridad críticas y establecer planes de acción para mitigarlas. (IBM, n.d.)

Herramientas comunes: Metasploit, Astra, Core Impact.

Escaneo de Vulnerabilidades: Implementar herramientas de escaneo de vulnerabilidades para ayudar a los equipos de seguridad a identificar brechas, puntos débiles o vulnerabilidades en cualquier parte del sistema, la red o las aplicaciones web, incluyendo componentes como firewalls, impresoras, máquinas de fax, routers, servidores web, sistemas operativos, servicios en la nube, herramientas de código abierto y durante el testeo de seguridad de aplicaciones. (Fortra, 2022)

Herramientas Comunes: Nessus, Qualys, OpenVAS.

Gestión de parches automatizada: Implementar un sistema automatizado de detección y aplicación de parche en sistemas y aplicaciones.

Herramientas Comunes: WSUS, Patch My PC, Automox.

Capacitación y Conciencia en Seguridad

Mantener al equipo de desarrollo actualizado y capacitado en las últimas prácticas y amenazas de seguridad.

Validación y Sanitización de Datos

La validación y sanitización de datos son prácticas críticas en el desarrollo seguro de aplicaciones, diseñadas para proteger las aplicaciones contra una variedad de ataques basados en la manipulación de entradas. Estas prácticas aseguran que los datos proporcionados por los usuarios sean adecuados, seguros y no puedan ser utilizados maliciosamente para comprometer la seguridad del sistema. La falta de validación y sanitización adecuadas es una de las causas más comunes de vulnerabilidades como las inyecciones SQL, cross-site scripting (XSS) y muchos otros ataques.

Implementación de la Validación y Sanitización de Datos.

- **Validación de datos**

Validación en el Lado del Cliente: Implementar validaciones básicas de datos del lado del cliente con el fin de mejorar la experiencia del usuario ya que se proporciona retroalimentación inmediata. Algunas de las validaciones que se pueden implementar son las siguientes:

- Validación de Formatos de Campo
- Validación de Longitud de Texto
- Validación Numérica
- Validación de Fechas
- Validación de Campos Obligatorios

Validación en el lado del servidor: Implementar validaciones de datos robustas en el servidor para asegurar que todos los datos recibidos cumplan con los requisitos de seguridad y formato.

Tipos de validación:

- **Validación de tipo**: Asegurarse de que los datos son del tipo esperado (por ejemplo, números, cadenas, fechas).
- **Validación de formato**: Verificar que los datos siguen un formato específico (por ejemplo, números de teléfono, direcciones de correo electrónico).
- **Validación de rango**: Confirmar que los datos caen dentro de un rango permitido (por ejemplo, valores numéricos mínimos y máximos).
- **Validación de longitud**: Comprobar que los datos tienen una longitud apropiada (por ejemplo, longitud mínima y máxima de contraseñas).

- **Sanitización de datos**

Escapado de Caracteres: Escapar caracteres especiales que pueden ser interpretados de manera peligrosa por el sistema (por ejemplo, el uso de comillas simples en SQL).

Eliminación de caracteres peligrosos: Remover o neutralizar caracteres que no sean necesarios y que puedan ser peligrosos (por ejemplo, etiquetas de HTML en campos de texto).

Uso de librerías de sanitización: Utilizar librerías y funciones de sanitización proporcionadas por frameworks de desarrollo que están diseñadas específicamente para manejar datos de manera segura.

- **Prácticas Recomendadas**

Principio de mínimo privilegio: Solo procesar y aceptar los datos que son absolutamente necesarios para la funcionalidad de la aplicación.

Listas blancas (Whitelisting): Ver ítem “Estrategias para Prevenir Fallos en la Validación y Sanitización de Entradas” en el punto “Principio de lista blanca (Whitelist)”.

Defensa en profundidad: Implementar múltiples capas de validación y sanitización tanto en el lado del cliente como en el servidor para asegurar una defensa robusta.

Revisión y actualización continua: Revisar y actualizar regularmente las prácticas de validación y sanitización para enfrentar nuevas amenazas y mantener la seguridad del sistema.

Gestión de Configuración Segura

La gestión de configuración segura es un componente crítico en el desarrollo y mantenimiento de aplicaciones y sistemas seguros. Involucra la implementación de prácticas y procedimientos para manejar de manera segura las configuraciones del software, asegurando que los parámetros y opciones de configuración no se conviertan en vectores de ataque. La gestión de configuración segura abarca desde la protección de archivos de configuración y la gestión de claves hasta la aplicación de configuraciones seguras por defecto.

Importancia de la Gestión de Configuración Segura.

Prevención de vulnerabilidades: Configuraciones incorrectas o inseguras pueden exponer aplicaciones a una amplia gama de vulnerabilidades, incluyendo la exposición de datos sensibles, escalamiento de privilegios, y acceso no autorizado.

Consistencia y estabilidad: Una gestión adecuada asegura que las configuraciones sean consistentes en todos los entornos (desarrollo, prueba, producción), reduciendo la probabilidad de errores y problemas de configuración.

Cumplimiento normativo: Muchas regulaciones y estándares de la industria requieren la implementación de controles de configuración segura para proteger la integridad y confidencialidad de los datos.

Prácticas de Gestión de Configuración Segura.

- **Protección de Archivos de Configuración**

Permisos y acceso: Asegurar que solo usuarios y procesos autorizados tengan acceso a archivos de configuración críticos. Utilizar permisos mínimos necesarios para estos archivos.

Cifrado de configuraciones Sensibles: Cifrar la información sensible dentro de archivos de configuración, como contraseñas y claves API.

- **Gestión de claves y secretos**

Almacenamiento seguro: Implementar herramientas especializadas para almacenar y gestionar secretos y claves, como HashiCorp Vault, AWS Secrets Manager, o Azure Key Vault. Por ejemplo, la configuración de un sistema de gestión de secretos para rotar automáticamente las claves y contraseñas utilizadas por la aplicación.

Rotación de claves: Implementar políticas de rotación regular de claves y contraseñas para minimizar el riesgo de compromiso. Por ejemplo, automatizar la rotación de claves de acceso a bases de datos y servicios externos.

- **Configuraciones seguras por defecto**

Valores por defecto seguros: Verificar que las configuraciones por defecto que vienen en las aplicaciones y sistemas sean seguras y de no ser así modificarlas para que lo sean. Por ejemplo, deshabilitar por defecto servicios y puertos no necesarios en la configuración del servidor.

Desactivación de funcionalidades no utilizadas: Garantizar que las características y servicios que no son necesarios sean desactivados o eliminados para reducir la superficie de ataque.

- **Auditoría y monitoreo de configuración**

Registro y seguimiento de cambios: Los cambios que se realicen a las configuraciones deben ser registrados detalladamente y monitoreados en tiempo real. Por ejemplo, la utilización de sistemas de control de versiones como Git para rastrear cambios en archivos de configuración y herramientas de monitoreo como Splunk para alertas en tiempo real.

Revisión y auditoría regular: Se recomienda realizar auditorías y revisiones periódicas a las configuraciones con el fin de establecer si deben ajustarse en caso de que estén desactualizadas o sean inseguras.

- **Automatización y Gestión Centralizada**

Herramientas de Gestión de Configuración: Utilizar herramientas de gestión de configuración como Ansible, Puppet, o Chef para automatizar la aplicación y el mantenimiento de configuraciones seguras.

Consistencia en entornos: Mantener seguimiento y control del estado de la configuración para asegurar que sean consistentes y seguras en todos los entornos (desarrollo, pruebas, producción). Por ejemplo, la implementación de pipelines de CI/CD que validen y apliquen configuraciones seguras de manera automática.

Auditorías y Revisiones de Seguridad

Las auditorías y revisiones de seguridad son procesos esenciales en el desarrollo seguro de aplicaciones. Estas prácticas se enfocan en identificar, analizar y corregir posibles vulnerabilidades de manera proactiva antes de que puedan ser explotadas por actores malintencionados. La implementación de revisiones de código y auditorías periódicas asegura que las aplicaciones mantengan un alto nivel de seguridad y conformidad con las mejores prácticas y normativas vigentes.

Importancia de las Auditorías y Revisiones de Seguridad.

Permiten identificar y corregir vulnerabilidades antes de que sean explotadas.

Ayudan a asegurar que las aplicaciones cumplan con las regulaciones y estándares de seguridad aplicables.

Fomentan una cultura de seguridad y mejora continua en el desarrollo de software.

Disminuyen la probabilidad de incidentes de seguridad y sus posibles impactos.

Tipos de Auditorías y Revisiones de Seguridad.

- **Revisiones de Código (Code Reviews)**

Revisiones manuales: Involucran la revisión detallada del código fuente por parte de desarrolladores experimentados para identificar vulnerabilidades y malas prácticas de seguridad. Por ejemplo, revisar manualmente el manejo de la entrada del usuario para detectar posibles inyecciones SQL o XSS.

Revisiones automatizadas: Utilizan herramientas automatizadas para escanear el código en busca de vulnerabilidades comunes y patrones de código inseguros. Por ejemplo, para realizar análisis de código estático se pueden usar herramientas como SonarQube, Checkmarx, o Fortify

- **Auditorías de Seguridad**

Auditorías Internas: Realizadas por el equipo de seguridad interno de la organización para evaluar y mejorar continuamente las prácticas de seguridad.

Auditorías Externas: Realizadas por entidades externas o consultores de seguridad para proporcionar una evaluación imparcial y exhaustiva de la seguridad de la aplicación.

- **Pruebas de Penetración (Pentesting)**

Pruebas Internas: Son simulaciones de ataques realizados por el equipo de seguridad interno para evaluar la resistencia de la aplicación ante ataques y establecer sus vulnerabilidades.

Pruebas Externas: Son simulaciones de ataques realizados por equipos externos para evaluar la resistencia de la aplicación ante ataques y establecer sus vulnerabilidades y generar un plan de acción.

Proceso de Auditorías y Revisiones de Seguridad.

- **Planificación**

Definir el alcance: En esta fase se establecen las áreas y componentes de la aplicación que van a ser revisadas y auditadas

Establecer criterios: Establecer los criterios y estándares de seguridad que se tendrán en cuenta para evaluar la aplicación

- **Ejecución**

Revisión de código manual y automatizada: Realizar revisiones de código tanto manuales como automatizadas para identificar vulnerabilidades.

Pruebas de Penetración: Llevar a cabo pruebas de penetración para identificar vulnerabilidades desde la perspectiva de un atacante.

- **Análisis y reporte**

Análisis de resultados: Realizar el análisis de los hallazgos generados en la revisión de código y auditorias con el fin de establecer los niveles de impacto de las vulnerabilidades y su gravedad (crítica, alta, media, baja.)

Generación de reportes: Generar reportes con el detalle de las vulnerabilidades, su impacto potencial y las recomendaciones para su mitigación.

- **Corrección y seguimiento**

Implementación de soluciones: Implementar la solución de los errores para corregir las vulnerabilidades identificadas siguiendo las recomendaciones del reporte de auditoría.

Revisión Posterior: Se debe realizar una revisión de seguimiento con el fin de asegurar que las correcciones se han implementado correctamente y no han introducido nuevas vulnerabilidades.

Cifrado de Datos Sensibles

El cifrado de datos sensibles es una medida esencial para garantizar la confidencialidad e integridad de la información. Consiste en convertir datos legibles en un formato codificado que solo puede ser descifrado por individuos autorizados. Esta práctica protege los datos tanto en tránsito como en reposo, previniendo accesos no autorizados y reduciendo el riesgo de violaciones de seguridad.

Importancia del Cifrado de Datos Sensibles.

Protege la información sensible asegurando que solo las personas autorizadas puedan acceder a ella.

Asegura que los datos no han sido alterados o manipulados durante el tránsito o almacenamiento.

Ayuda a cumplir con regulaciones y estándares de seguridad que requieren la protección de datos sensibles mediante el cifrado.

Minimiza el impacto de posibles violaciones de datos, ya que los datos cifrados no se pueden leer sin la clave de descifrado correcta.

Tipos de Cifrado.

- **Cifrado en Tránsito**

Protocolo HTTPS: Utiliza SSL/TLS para cifrar la comunicación entre el cliente y el servidor utilizando certificados SSL/TLS válidos.

VPNs (Virtual Private Networks): Cifran todo el tráfico de red entre dispositivos, creando un túnel seguro (Fortinet, s.f.).

Cifrado de correo electrónico: Utilizar protocolos como S/MIME o PGP para cifrar correos electrónicos.

- **Cifrado en Reposo**

Cifrado de Base de Datos: Utilizar mecanismos de cifrados nativos de bases de datos para proteger datos almacenados. Por ejemplo, en las bases de datos Oracle, IBM y Microsoft se puede configurar Transparent Data Encryption (TDE) para proteger los datos en reposo. (Kaspersky, s.f.)

Cifrado de Archivos y Almacenamiento: Implementar cifrado a nivel de sistema de archivos o unidades de disco. Por ejemplo, utilizar BitLocker en Windows o FileVault en macOS para cifrar discos duros.

Cifrado en Nube: Utilizar servicios de cifrado ofrecidos por proveedores de nube para proteger datos almacenados. Por ejemplo, configurar AWS KMS (Key Management Service) para cifrar datos almacenados en Amazon S3.

Prácticas de Cifrado de Datos Sensibles.

- **Selección de algoritmos de cifrado seguros**

Estándares de la Industria: Hacer uso de algoritmos de cifrado recomendados y ampliamente reconocidos, como AES (Advanced Encryption Standard) con claves de 256 bits.

Evitar algoritmos obsoletos: A la hora de implementar algoritmos se recomienda asegurar que los algoritmos de cifrado no sean obsoletos o inseguros, como DES o RC4.

- **Gestión segura de claves de cifrado**

Almacenamiento seguro de claves: Hacer uso de sistemas de gestión de claves para administrar las claves de cifrado de una forma segura.

Rotación regular de claves: Establecer políticas que indiquen el cambio regular de las claves con el propósito de disminuir el riesgo de compromiso.

- **Implementación de cifrado integral**

Cifrado extremo a extremo (E2EE): Hacer uso de claves de cifrado con el objetivo de mezclar los datos o la información para que los datos estén protegidos de extremo a extremo y solo las personas que tienen la clave puedan acceder a la información.

Integración en el ciclo de vida del desarrollo: Las prácticas de cifrado deben incorporarse en cada una de las fases del ciclo de vida del desarrollo para contribuir en la seguridad de los datos. Por ejemplo, incluir pruebas de cifrado en la fase de integración continua (CI) y despliegue continuo (CD).

- **Auditorías y monitoreo**

Auditorías regulares: La implementación de revisiones de prácticas de cifrado y auditorías periódicas aseguran que el cifrado se mantenga seguro y efectivo.

Monitoreo de actividades de cifrado: Hacer usos de sistemas de monitoreo que permitan detectar y alertar sobre actividades sospechosas relacionadas con el cifrado.

Conclusiones

Al aplicar estas buenas prácticas y metodologías, las organizaciones pueden desarrollar aplicaciones más seguras y resilientes, reduciendo significativamente el riesgo de explotación de vulnerabilidades. La siguiente sección detallará estas prácticas y metodologías propuestas, proporcionando una guía comprensiva para su implementación en diversos contextos de desarrollo de software.

Metodologías

Finalmente, se realiza una evaluación de las siguientes metodologías o marco de trabajos para el desarrollo seguro de software revisados en el capítulo 1 de este documento: SDL (Security Development Lifecycle), OpenSAMM (Software Assurance Maturity Model), BSIMM (Building Security In Maturity Model), CLASP (Comprehensive, Lightweight Application Security Process), TSP-Secure (Team Software Process for Secure Software Development), CbyC (Checklists by Code), DevSecOps, OWASP (Open Web Application Security Project), y OWASP ASVS (Application Security Verification Standard). Como resultado del análisis se indica cual podría ser la metodología más acertada en base a sus características distintivas, su efectividad para reducir vulnerabilidades, su aplicabilidad en diversos entornos de desarrollo, y su aceptación y uso en la industria.

En primer lugar, se realizar una descripción de las ventajas, desventajas y usos comunes de cada metodología

Security Development Lifecycle (SDL)

Ventajas

Proporciona un enfoque estructurado y sistemático para integrar la seguridad en cada fase del ciclo de vida del desarrollo de software, ya que se compone por una serie de fases definidas cada una con actividades claras orientadas a la seguridad, también es una metodología basada en principios de seguridad como el análisis de amenazas, codificación segura y pruebas de seguridad, además, implementa la mejora continua y el monitoreo por medio de la capacitación continua y el monitoreo y respuesta a nuevas amenazas.

Enfatiza la formación y concienciación de los desarrolladores sobre las prácticas de seguridad. Capacita continuamente a los desarrolladores sobre amenazas y técnicas de ataque actuales, mejorando sus prácticas de codificación. Les enseña a adoptar prácticas de codificación segura, realizar revisiones y análisis de código para identificar vulnerabilidades, y reconocer errores comunes que puedan generar riesgos. Además, incluye formación en el uso de herramientas de análisis de código estático y dinámico, integrando pruebas de seguridad automatizadas en los procesos de CI/CD. Además, Fomenta una cultura de seguridad promoviendo la responsabilidad compartida de la seguridad del software entre todo el personal de desarrollo y asegurando un ambiente de colaboración efectivo entre los equipos de seguridad y otros departamentos. También garantiza que los desarrolladores cumplan con las normas y estándares de seguridad, manteniendo la documentación adecuada para auditorías.

Incluye actividades específicas como análisis de amenazas, revisión de código, y pruebas de penetración, actividades que son fundamentales para identificar, mitigar y prevenir vulnerabilidades de seguridad en cada fase del desarrollo del software.

Desventajas

Puede ser costoso y requerir un tiempo significativo para su implementación completa debido a factores relacionados con los recursos necesarios (personal especializado, herramientas que incluyen mantenimiento y actualización), la formación, y la adaptación de procesos.

Puede ser visto como rígido en entornos de desarrollo ágil debido a su estructura secuencial y formal, que contrasta con la naturaleza iterativa, flexible y adaptativa de Agile

Usos Comunes

Utilizado por grandes organizaciones que desarrollan software crítico donde la seguridad es una prioridad alta.

OpenSAMM (Software Assurance Maturity Model)

Ventajas

Proporciona un marco de trabajo flexible y escalable para evaluar y mejorar la seguridad del software porque permite a las organizaciones adaptar las prácticas de seguridad a sus necesidades específicas, implementar mejoras de manera incremental, y escalar estas prácticas según el tamaño y complejidad de la organización.

Permite a las organizaciones personalizar el modelo según sus necesidades específicas debido a su diseño flexible y modular.

Facilita la medición del progreso en la madurez de la seguridad del software al proporcionar una estructura clara y objetiva para evaluar la madurez de seguridad, alinear las evaluaciones con un estándar común, proporcionar orientación sobre cómo mejorar y permitir un seguimiento y ajuste continuo de las estrategias de seguridad del software.

Desventajas

Requiere un compromiso continuo y recursos para la implementación y mejora, ya que esto implica no solo la asignación de tiempo y esfuerzo, sino también la inversión en formación,

tecnología y cambios organizacionales para garantizar que se logren los beneficios deseados en términos de una mejor postura de seguridad del software.

Puede ser complejo para organizaciones pequeñas sin experiencia en seguridad debido a la necesidad de conocimientos especializados, recursos limitados, complejidad de los procesos, falta de experiencia en evaluaciones de seguridad y la posible necesidad de asistencia externa. Sin embargo, con el apoyo adecuado y un enfoque gradual, las organizaciones pequeñas pueden superar estos desafíos y beneficiarse de la implementación de prácticas de seguridad del software basadas en OpenSAMM.

Usos Comunes

Adecuado para organizaciones que buscan un enfoque gradual y medible para mejorar la seguridad del software.

Building Security In Maturity Model (BSIMM)

Ventajas

Basado en la observación de prácticas de seguridad efectivas en diversas organizaciones lo que facilita que brinde una guía basada en datos reales sobre las prácticas de seguridad, es decir se basa en un modelo de datos empírico que refleja las prácticas comunes y exitosas de seguridad del software en el mundo real.

Facilita la comparación y evaluación de las prácticas de seguridad frente a otras organizaciones ya que proporciona un marco estructurado y estándar.

Desventajas

No prescribe prácticas específicas, lo que puede requerir interpretaciones adicionales para aplicarlo y esfuerzos de personalización por parte de las organizaciones, lo que puede resultar en una curva de aprendizaje mayor y una mayor complejidad en su aplicación.

Enfocado principalmente en organizaciones grandes con programas de seguridad bien establecidos por lo que puede limitar su aplicabilidad y relevancia para organizaciones más pequeñas o en desarrollo.

Usos Comunes

Utilizado por organizaciones que desean basar sus prácticas de seguridad en datos empíricos y benchmarking.

Comprehensive, Lightweight Application Security Process (CLASP)

Ventajas

Proporciona una serie de actividades prácticas y específicas para incorporar la seguridad en el desarrollo de software pues hace que la seguridad sea más accesible, facilita la implementación, fomenta la colaboración y la comunicación, y permite una adaptación flexible a diferentes contextos organizacionales. Esto ayuda a garantizar que la seguridad del software sea una parte integral y efectiva del proceso de desarrollo.

Enfocado en integrar la seguridad desde el inicio del proyecto lo que contribuye en la reducción de costos hasta la mejora de la resiliencia del software y el cumplimiento de requisitos de seguridad

Es adaptable y no intrusivo en el proceso de desarrollo existente lo que se constituye en ventaja ya que permite realizar el proceso de integración sin fricciones, flexibilidad en la implementación, adopción gradual, minimización de la resistencia al cambio, alineación con metodologías ágiles y promoción de la colaboración y la comunicación

Desventajas

Puede carecer de la profundidad necesaria para organizaciones grandes con requisitos de seguridad complejos por lo que las organizaciones pueden necesitar complementar CLASP con

otros marcos y enfoques de seguridad más especializados para satisfacer completamente sus necesidades de seguridad

Menos conocido y adoptado que otros modelos por lo que puede requerir una mayor conciencia y promoción para aumentar su adopción y uso en la comunidad de seguridad del software.

Usos Comunes

Adecuado para pequeñas y medianas empresas que buscan una integración rápida y efectiva de prácticas de seguridad.

Team Software Process for Secure Software Development (TSP-Secure)

Ventajas

Combina prácticas de desarrollo de software de alta calidad con seguridad, porque contribuye a reducir vulnerabilidades, mejorar la calidad del software, reducir costos y garantizar la alineación con estándares de calidad y seguridad reconocidos.

Promueve la formación y concienciación de todos los miembros del equipo en prácticas de seguridad porque contribuye a mejorar la cultura de seguridad, aumentar la responsabilidad y participación, facilitar la identificación temprana de problemas de seguridad, mejorar la eficiencia y efectividad en la implementación, y fomentar la mejora continua y adaptabilidad en materia de seguridad del software.

Integra la seguridad en el flujo de trabajo ágil y colaborativo porque garantiza la coherencia con metodologías ágiles, facilita el feedback temprano y frecuente, promueve la colaboración interdisciplinaria, permite la priorización basada en riesgos y ofrece adaptabilidad y flexibilidad para responder a los cambios en el proceso de desarrollo de software.

Desventajas

Puede requerir una inversión considerable en capacitación y cambios en el proceso debido a los costos, el tiempo y los esfuerzos requeridos para su adopción. Sin embargo, estos desafíos pueden ser superados con una planificación cuidadosa, una comunicación efectiva y un compromiso firme por parte de la organización para mejorar la seguridad del software.

Menos flexible para entornos que no siguen estrictamente el TSP ya que puede requerir ajustes significativos para su implementación en entornos que no están alineados con este marco específico

Usos Comunes

Ideal para equipos que ya utilizan TSP y desean integrar prácticas de seguridad sin interrumpir su flujo de trabajo.

Checklists by Code (CbyC)

Ventajas

Proporciona listas de verificación específicas y prácticas para asegurar el código esto lo hace valioso para desarrolladores y equipos de desarrollo que buscan mejorar la seguridad de sus aplicaciones de manera efectiva y práctica.

Fácil de implementar y utilizar por desarrolladores individuales debido a su enfoque práctico, sencillez de uso, adaptabilidad y apoyo de la comunidad

Puede integrarse en herramientas de revisión de código automatizadas por lo que ofrece beneficios, como la estandarización de verificaciones, la integración en el flujo de desarrollo, el ahorro de tiempo y recursos, y la posibilidad de mejorar continuamente la seguridad del código.

Desventajas

Puede no cubrir todos los aspectos de la seguridad del software debido a su enfoque específico en la seguridad del código, sus limitaciones en el alcance, la necesidad de

actualización continua frente a nuevas amenazas, y la omisión de prácticas de seguridad operacional, integración y pruebas que son críticas para una seguridad completa del software.

Menos enfoque en la estrategia general y más en tácticas específicas porque se centra en proporcionar acciones prácticas y detalladas para asegurar el código, sin abarcar necesariamente las directrices estratégicas y organizativas necesarias para una gestión integral de la seguridad del software.

Usos Comunes

Utilizado por equipos de desarrollo que buscan mejorar la calidad del código mediante listas de verificación de seguridad.

DevSecOps

Ventajas

Integra la seguridad en el flujo de trabajo de DevOps, promoviendo la colaboración entre desarrolladores, operaciones y seguridad y garantizando que la seguridad sea una responsabilidad compartida por todo el equipo.

Facilita la implementación continua de pruebas y controles de seguridad ya que está enfocado en automatizar las tareas, integrarlas en el pipeline de CI/CD, proporcionar detección temprana de vulnerabilidades, ofrecer feedback rápido, estandarizar los controles de seguridad, promover una cultura de mejora continua y adaptarse rápidamente a nuevas amenazas para mantener la agilidad del desarrollo.

Desventajas

Puede requerir una significativa reestructuración organizativa y cultural porque implica cambios fundamentales en la forma en que se gestionan y priorizan las prácticas de seguridad, así como en la colaboración y comunicación entre diferentes equipos. Esta transición puede ser

compleja y demandar una reevaluación de roles, procesos y herramientas, junto con una inversión en capacitación y un esfuerzo concertado para superar la resistencia al cambio.

Dependiente de herramientas y prácticas de automatización avanzadas porque estas herramientas permiten la integración y el despliegue continuo, automatizan las pruebas y el monitoreo de seguridad, gestionan la configuración y la infraestructura de manera segura, coordinan las actividades del pipeline, y aseguran la gobernanza y el cumplimiento de manera eficiente y escalable.

Usos Comunes

Ideal para organizaciones que ya practican DevOps y desean integrar la seguridad sin sacrificar la agilidad y la velocidad.

Open Web Application Security Project (OWASP)

Ventajas

Ofrece una amplia gama de recursos, herramientas y guías para mejorar la seguridad de las aplicaciones web porque proporciona guías de seguridad reconocidas, herramientas de seguridad de código abierto, documentación y recursos educativos extensos, metodologías estructuradas, fomenta la colaboración comunitaria, organiza eventos educativos, ofrece hojas de referencia rápida y mantiene sus recursos actualizados para abordar las amenazas emergentes.

Conocido y ampliamente adoptado en la industria.

Proporciona listas de las principales vulnerabilidades y prácticas para mitigarlas (como OWASP Top Ten) porque identifica las vulnerabilidades de seguridad más críticas, ofrece recomendaciones específicas para mitigarlas, es ampliamente reconocida y utilizada en la industria, ayuda a educar y concienciar sobre seguridad, sirve como base para políticas y normativas, y se complementa con otros recursos detallados

Desventajas

Enfocado principalmente en aplicaciones web, puede no cubrir todas las necesidades de otros tipos de software.

Requiere que las organizaciones seleccionen y adapten los recursos y herramientas adecuados para sus necesidades. Esto implica evaluar las guías y herramientas disponibles y personalizarlas para que se ajusten a los requisitos de seguridad de la organización.

Usos Comunes

Utilizado por desarrolladores web y organizaciones que buscan mejorar la seguridad de sus aplicaciones web.

OWASP Application Security Verification Standard (ASVS)

Ventajas:

Proporciona un estándar detallado y específico para verificar la seguridad de las aplicaciones, debido a su estructura organizada, enfoque gradual, detalle y especificidad de los requisitos, basado en las mejores prácticas de la industria, y su amplia adopción global como estándar de referencia en seguridad de aplicaciones

Facilita la evaluación de la seguridad a diferentes niveles de detalle al proporcionar una estructura de niveles graduales, flexibilidad en la implementación, requisitos detallados y específicos, y adaptabilidad a diferentes contextos y necesidades de seguridad. Esto permite que las organizaciones realicen evaluaciones de seguridad que sean precisas, relevantes y proporcionales al riesgo y la criticidad de sus aplicaciones.

Basado en una comunidad amplia y activa de profesionales de seguridad que colaboran para desarrollar, revisar y mantener un estándar de verificación de seguridad de aplicaciones de

alta calidad. Esta comunidad proporciona la experiencia, la diversidad y la retroalimentación necesarias para garantizar que el estándar sea relevante, preciso y efectivo en la práctica.

Desventajas

Puede ser percibido como complejo y extenso para organizaciones pequeñas debido a la cantidad de requisitos, los niveles de seguridad graduales, la necesidad de interpretación y adaptación, los recursos limitados disponibles y el enfoque en aplicaciones complejas. Esto puede dificultar su adopción y aplicación efectiva en entornos con limitaciones de recursos y experiencia en seguridad de aplicaciones.

Requiere una implementación rigurosa y detallada debido a la complejidad de los requisitos, los niveles graduales de seguridad, la adaptación a contextos específicos, la verificación y pruebas necesarias, y el mantenimiento continúo requerido para asegurar la efectividad de las medidas de seguridad implementadas.

Usos Comunes

Utilizado por organizaciones que desean implementar un estándar robusto y completo para la verificación de la seguridad de aplicaciones.

Mejor Metodología para Reducción de Vulnerabilidades

DevSecOps. Destaca como una poderosa metodología para la reducción de vulnerabilidades gracias a su enfoque integral en la integración de la seguridad en todas las etapas del ciclo de vida del desarrollo de software. Al promover una cultura de colaboración entre los equipos de desarrollo, operaciones y seguridad, DevSecOps garantiza que la seguridad sea una prioridad desde el inicio del proceso de desarrollo hasta el despliegue y operación en producción. Además, la automatización de pruebas y controles de seguridad permite una detección temprana de posibles vulnerabilidades, lo que facilita su corrección antes de que se

conviertan en problemas significativos. Esta metodología se adapta especialmente bien a organizaciones que buscan mantener un equilibrio entre agilidad y seguridad, permitiéndoles avanzar rápidamente en sus proyectos de desarrollo mientras garantizan un alto nivel de protección contra amenazas cibernéticas

OWASP y OWASP ASVS. La Organización de la Seguridad en Aplicaciones Web (OWASP) ofrece una invaluable colección de recursos y herramientas prácticas diseñadas para mejorar la seguridad de las aplicaciones web. Desde listas de verificación de seguridad hasta guías de buenas prácticas y herramientas de análisis de vulnerabilidades, OWASP proporciona una amplia gama de recursos que pueden utilizarse para fortalecer la seguridad en el desarrollo de software. Por otro lado, el OWASP Application Security Verification Standard (ASVS) representa un estándar detallado y específico para verificar la seguridad de las aplicaciones. Al definir una serie de requisitos y controles de seguridad organizados en diferentes niveles de complejidad, el ASVS brinda a las organizaciones una guía clara para evaluar y mejorar la seguridad de sus aplicaciones. Ambos, OWASP y OWASP ASVS, son ampliamente adoptados y reconocidos en la industria de la seguridad informática, lo que facilita su implementación y alineación con las mejores prácticas de seguridad en el desarrollo de software.

SDL. El Ciclo de Vida del Desarrollo Seguro (SDL) destaca como una sólida metodología para reducir vulnerabilidades al ofrecer un enfoque estructurado y sistemático para integrar la seguridad en todas las fases del ciclo de vida del desarrollo de software. SDL enfatiza la importancia de considerar la seguridad desde las etapas iniciales del proceso de desarrollo y mantenerla como una prioridad continua a lo largo de todo el ciclo de vida del software. Al incluir actividades específicas como análisis de amenazas, revisiones de código y pruebas de penetración, SDL proporciona un marco robusto para identificar y mitigar vulnerabilidades de

manera proactiva. Además, SDL enfatiza la formación y concienciación de los desarrolladores sobre prácticas de seguridad, lo que contribuye a una cultura organizacional centrada en la seguridad. Esta metodología es especialmente adecuada para organizaciones que buscan implementar prácticas de seguridad sólidas y efectivas en su proceso de desarrollo de software, lo que conduce a una reducción significativa de las vulnerabilidades y una mayor resistencia frente a las amenazas cibernéticas.

OpenSAMM. El Modelo de Madurez de Seguridad de Aplicaciones Abiertas (OpenSAMM) se destaca como una metodología robusta para la reducción de vulnerabilidades al proporcionar un marco de trabajo flexible y escalable para evaluar y mejorar la seguridad del software. OpenSAMM permite a las organizaciones personalizar el modelo según sus necesidades específicas, lo que les permite adaptar las prácticas de seguridad a su contexto único. Además, OpenSAMM facilita la medición del progreso en la madurez de la seguridad del software, lo que permite a las organizaciones realizar un seguimiento de su evolución en términos de seguridad. Aunque requiere un compromiso continuo y recursos para la implementación y mejora, OpenSAMM ofrece una guía basada en datos reales sobre las prácticas de seguridad que funcionan, lo que permite a las organizaciones tomar decisiones informadas y efectivas para reducir las vulnerabilidades en sus aplicaciones.

Conclusión

La elección de la metodología para el desarrollo seguro de software depende de las necesidades y características específicas de la organización. DevSecOps, OWASP/OWASP ASVS, SDL, y OpenSAMM son destacadas por su efectividad, flexibilidad y capacidad para reducir vulnerabilidades de manera significativa. La combinación de estas metodologías puede proporcionar un enfoque integral y robusto para el desarrollo seguro de software, asegurando que

las aplicaciones sean protegidas contra amenazas y vulnerabilidades de manera proactiva y eficiente.

Conclusiones

En conclusión podemos decir que las metodologías de desarrollo seguro de software deben incluir temas como: levantamiento requerimientos de seguridad en la fase de levantamiento de requisitos, y se debe establecer posteriormente la estructura de las funcionalidades y de su arquitectura incluyendo los requerimientos de seguridad especificados y los riesgos identificados en otras aplicaciones, además se debe capacitar el personal sobre temas relacionados a la seguridad y lineamientos específicos de desarrollo seguro, también se debe garantizar el entendimiento de las tareas asignadas al equipo y establecer métricas para medir el nivel de cumplimiento de seguridad de las aplicaciones y realizar constantes actualizaciones que mejoren los niveles de seguridad, establecer planes de mitigación de riesgos, por tal motivo podemos decir que estas metodologías deben implementarse para ejecutar los proyectos de manera organizada, mejorar el rendimiento, eficiencia de estos y poder cumplir el objetivo, además mejoran el nivel de seguridad de las aplicaciones y disminuyen los costos debido a la detección temprana de fallos.

Las buenas prácticas de desarrollo seguro son fundamentales en la creación de software, ya que contribuyen a reducir el tiempo y los costos del proceso, minimizando errores y mejorando la calidad del producto final. Esto permite obtener sistemas más confiables y alineados con las necesidades actuales del negocio. Además, estas prácticas facilitan el aprendizaje y brindan a los desarrolladores una metodología de programación más estructurada, precisa, rigurosa y segura.

Después de hacer la revisión bibliográfica para identificar cuáles son los tipos de ataques que se presentan, se puede decir que los ataques de ingeniería social son uno de los más comunes y este tipo de ataque se puede combinar con otros como el ransomware que aprovechan las

brechas de seguridad de los sistemas para llevar a cabo el ataque. Además, el ransomware es uno de los ataques que representan mayor peligro para individuos, infraestructura, organizaciones y empresas de todos los tamaños, podemos ver que este tipo de ataque fue utilizado con una tasa alta de éxito en los últimos años, por ejemplo, para febrero de 2022 en el norte de Alemania se realizó un ataque a dos empresas que afectó el sistema de pago de gasolineras. Después en el mes de marzo se presentó un ataque en Grecia el cual afectó la entrega de correos y el proceso de las transacciones bancarias. Luego en el mes de mayo se presentó un ataque en los organismos de gobierno de Costa Rica lo que obligó a declarar una emergencia nacional, los hospitales se cerraron y fueron interrumpidas las actividades de las aduanas y la recaudación de impuestos.

Como podemos ver en la actualidad los ataques cibernéticos son una de las grandes problemáticas mundiales, la cual merece la atención de personas, empresas, organizaciones, ya que si tenemos acceso a la tecnología en cualquier momento podemos ser víctimas de estos. Según el informe Microsoft Digital Defense Report 2022, la cantidad de ataques de contraseña por segundo está en un estimado de 921 ataques, lo cual indica que en un año ha aumentado en un 74%.

Por otro lado, según datos recopilados por el laboratorio de inteligencia de amenazas de Fortinet, “Brasil sufrió 31.500 millones de intentos de ciberataques de enero a junio de este año, un aumento del 94 % con respecto al mismo período del año pasado (con 16.200 millones), siendo el segundo país más atacado de América Latina, detrás de México, con 85.000 millones, y seguido por Colombia (con 6.300 millones) y Perú (con 5.200 millones). En total, la región de América Latina y el Caribe ha sufrido 137 mil millones de intentos de ciberataques” (Bnamericas, 2022).

De acuerdo con lo anterior invertir en ciberseguridad es clave la para la seguridad y estabilidad de las personas, empresas, organizaciones y estados, ya que de lo contrario se verán más expuestos a diferentes ciberataques con consecuencias que van desde interrupción de las actividades hasta daños en la propiedad física y la vida humana.

Después de explorar una variedad de metodologías y buenas prácticas para el desarrollo seguro de aplicaciones en el Capítulo 3, es evidente que existe un conjunto diverso de enfoques disponibles para mitigar las vulnerabilidades en el desarrollo de software. Desde el enfoque estructurado y sistemático del Ciclo de Vida del Desarrollo Seguro (SDL) hasta la flexibilidad y escalabilidad del Modelo de Madurez de Seguridad de Aplicaciones Abiertas (OpenSAMM), cada metodología y práctica ofrece sus propias ventajas y desafíos.

Es crucial reconocer que no existe una solución única para garantizar la seguridad del software. Más bien, la combinación adecuada de metodologías y prácticas dependerá de las necesidades específicas y las características de cada organización. Sin embargo, queda claro que la integración temprana y continua de la seguridad, la automatización de procesos, la gestión de configuración segura y otras prácticas identificadas en este capítulo juegan un papel fundamental en la reducción de vulnerabilidades y en la construcción de aplicaciones más seguras y resistentes.

Al adoptar un enfoque proactivo y multifacético que incorpore las mejores prácticas y metodologías disponibles, las organizaciones pueden fortalecer su postura de seguridad y mitigar los riesgos asociados con el desarrollo de software. En última instancia, la implementación exitosa de estas prácticas y metodologías no solo protegerá los activos digitales de una organización, sino que también contribuirá a la confianza del usuario, la reputación de la marca y el éxito a largo plazo en un entorno cada vez más digital y conectado.

Referencias

Belcic, I. (22 de septiembre de 2020). *¿Qué es la inyección de SQL y cómo funciona?* Obtenido de www.avast.com: https://www.avast.com/es-es/c-sql-injection

Sawka, M., & Niemiec, M. (20 de septiembre de 2022). A Sponge-Based Key Expansion Scheme for Modern Block Ciphers. *Energies*, *15*(19). doi:10.3390/en15196864

1Library. (s.f.). *La metodología TSP y la calidad de software.* Recuperado el 28 de octubre de 2024, de https://1library.co/article/la-metodolog%C3%ADa-tsp-y-la-calidad-de-software.8yddg3ey

27000, I. (2018). Obtenido de https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en

AltexSoft Inc. (7 de marzo de 2024). *Object-Relational Mapping Tools: Pros, Cons, and When to Use*. Obtenido de altexsoft.medium.com: https://altexsoft.medium.com/object-relational-mapping-tools-pros-cons-and-when-to-use-c1283b3c1164

Amaya Valencia, Y. A. (2022). *Política general para la aplicación de buenas prácticas de seguridad y privacidad de la información en el desarrollo de software para empresas medianas y pequeñas en el estándar OWASP*. Obtenido de repository.ucatolica.edu.co: https://repository.ucatolica.edu.co/handle/10983/27649

Ambit-bst. (10 de noviembre de 2020). *Tipos de Vulnerabilidades y Amenazas informáticas.* Obtenido de https://www.ambit-bst.com/blog/tipos-de-vulnerabilidades-y-amenazas-inform%C3%A1ticas

Arciniega, F. (s.f.). *¿Qué es el TSP – Team Software Process?* Recuperado el 28 de octubre de 2024, de https://fernandoarciniega.com/tsp-team-software-process/

Arciniega, F. (20 de septiembre de 2022). *Normas y Estándares de calidad para el desarrollo de Software*. Obtenido de https://fernandoarciniega.com/normas-y-estandares-de-calidad-para-el-desarrollo-de-software/

Arnal, C. (24 de julio de 2024). *Las brechas de seguridad en la cadena de suministro suben un 68%*. Obtenido de https://www.watchguard.com/es/wgrd-news/blog/las-brechas-de-seguridad-en-la-cadena-de-suministro-suben-un-68

AWS Amazon. (s.f.). *¿Qué es DevSecOps?* Obtenido de aws.amazon.com: https://aws.amazon.com/es/what-is/devsecops/

Axur. (s.f.). *Amenazas externas, internas y riesgo digital: ¿Cuál es la diferencia?* Recuperado el 22 de octubre de 2024, de https://blog.axur.com/es-es/amenazas-externas-internas-y-riesgo-digital-cu%C3%A1l-es-la-diferencia

Azure. (12 de septiembre de 2022). *¿Qué es DevOps?* Obtenido de azure.microsoft.com: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/

Baldassarre, M. T., Barletta, V. S., Caivano, D., & Scalera, M. (28 de febrero de 2020). Integrating security and privacy in software development. *Software Quality Journal, 28*(3), 987–1018. doi:https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/s11219-020-09501-6

Bello, E. (8 de noviembre de 2028). *Ciberseguridad: Tipos de ataques y en qué consisten.* Obtenido de www.iebschool.com: https://www.iebschool.com/blog/ciberseguridad-ataques-tecnologia/

Bnamericas. (2022). *Brasil es el segundo país que más ciberataques sufre en América Latina*. Obtenido de www.bnamericas.com: https://www.bnamericas.com/es/noticias/brasil-es-el-segundo-pais-que-mas-ciberataques-sufre-en-america-

latina#:~:text=Brasil%20sufri%C3%B3%2031.500%20millones%20de,con%206.300%20millones)%20y%20Per%C3%BA

Bocetta, S. (17 de Diciembre de 2020). *Los certificados SSL/TLS, esenciales para las aplicaciones móviles*. Obtenido de GlobalSign Blog: https://www.globalsign.com/es/blog/understanding-why-ssltls-certificates-are-essential-securing-mobile-applications

Bodnar, D. (29 de octubre de 2020). *¿Qué es la ingeniería social?* Obtenido de www.avast.com: https://www.avast.com/es-es/c-social-engineering#:~:text=La%20ingenier%C3%ADa%20social%20es%20la%20pr%C3%A1ctica%20de%20utilizar,v%C3%ADctimas%20a%20actuar%20en%20contra%20de%20sus%20intereses.

Bolullo, A. (30 de mayo de 2019). *DevSecOps: la evolución de la seguridad del software*. Obtenido de www.paradigmadigital.com: https://www.paradigmadigital.com/dev/devsecops-evolucion-seguridad-software/

Bonneau, J. H. (2012). The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. *2012 IEEE Symposium on Security and Privacy*, 553-567. doi:10.1109/SP.2012.44.

Brito-Abundis, C. J. (3 de diciembre de 2013). Metodologías para desarrollar software. *repositorio.uta.edu.ec*, pág. https://core.ac.uk/download/pdf/322549836.pdf.

Cassottana, B., Roomi, M. M., Mashima, D., & Sansavini, G. (16 de enero de 2023). Resilience analysis of cyber-physical systems: A review of models and methods. *43*(15), 2359-2379. Obtenido de https://doi.org/10.1111/risa.14089

CEC. (20 de marzo de 2020). *¿Qué es la ISO 27034?* Obtenido de Confederación de Empresarios de La Coruña: https://www.cec.es/isoiec-27034-estandar-internacional-para-la-seguridad-de-las-aplicaciones/

Chaloupka, P. T. (11 de abril de 2024). *Seguridad de datos en la nube: definiciones, riesgos y 7 mejores prácticas para la protección de datos en la nube.* Obtenido de www.safetica.com: https://www.safetica.com/es/blog/seguridad-de-datos-en-la-nube-definiciones-riesgos-y-7-mejores-practicas-para-la-proteccion-de-datos-en-la-nube

Chandra, P. (2013). *Software Assurance Maturity Model, Una guía para integrar seguridad en el desarrollo de software.* Obtenido de https://opensamm.org/downloads/SAMM-1.0-es_MX.pdf

Checkpoint. (s.f.). *¿Qué es el ransomware?* Recuperado el 17 de octubre de 2024, de https://www.checkpoint.com/cyber-hub/threat-prevention/ransomware/

Ciberinseguro. (14 de febrero de 2023). *Denegación de Servicio (DoS y DDoS): cómo funciona y cómo protegerse.* Obtenido de https://ciberinseguro.com/denegacion-de-servicio-dos-y-ddos-como-funciona-y-como-protegerse/

Ciccariello, P. (2022). *MALWARE: Los más peligrosos y cómo enfrentarlos* (Vol. Volumen 204 de Informes USERS). (RedUSERS, Ed.)

Cloudflare. (s.f.). *¿Qué es un ataque de denegación de servicio (DoS)?* Recuperado el 18 de octubre de 2024, de https://www.cloudflare.com/es-es/learning/ddos/glossary/denial-of-service

CNN Chile. (8 de julio de 2019). *British Airways enfrenta millonaria multa tras vulneración de datos de sus clientes.* Obtenido de www.cnnchile.com:

https://www.cnnchile.com/mundo/british-airways-millonaria-multa-vulneracion-datos-clientes_20190708/

Coppola, M. (08 de mayo de 2023). *Seguridad informática: qué es, tipos y características*. Obtenido de blog.hubspot.es: https://blog.hubspot.es/website/que-es-seguridad-informatica

Crowe. (septiembre de 2022). *Auditoría de Seguridad Web OWASP*. Obtenido de www.crowe.com: https://www.crowe.com/uy/services/ciberseguridad/owasp

Das, A., Borisov, N., & Caesar, M. C. (2014). Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*.

Diaz, S. M. (2014). *Pruebas de seguridad en aplicaciones web como imperativo en la calidad de desarrollo del software*. Obtenido de repository.unab.edu.co: https://repository.unab.edu.co/bitstream/handle/20.500.12749/12308/2014_CIINATIC_capitulo7.pdf?sequence=1&isAllowed=y

Diéguez, M. &. (2012). *De la Gestión de Seguridad en el Ciclo de Vida del Software*. Obtenido de https://www.researchgate.net/publication/236011343_De_la_Gestion_de_Seguridad_en_el_Ciclo_de_Vida_del_Software

Editorial Etecé. (12 de agosto de 2022). *Tecnología*. Obtenido de concepto.de: https://concepto.de/tecnologia/#ixzz7gUGomufu

Editorial Etecé. (19 de noviembre de 2023). *Software*. Obtenido de Concepto.de: https://concepto.de/software/

Escrivá-Gascó, G. (2023). *Seguridad informática.* (S. Macmillan Iberia, Ed.) Madrid. Obtenido de https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/43260?page=114.

European Knowledge Center for Information Technology. (14 de marzo de 2022). *Tecnología de redes*. Obtenido de TIC Portal: https://www.ticportal.es/glosario-tic/tecnologia-redes

Fernández-Bernal, C. (29 de septiembre de 2018). *Guía de buenas prácticas de seguridad en el desarrollo de software con base en estándares reconocidos en empresas de desarrollo de software.* Obtenido de Repositorio Institucional UNAD: https://repository.unad.edu.co/handle/10596/20449

Ferraris, J. A. (2010). *Fundamentos de informática y programación en C* (1 Ed ed.). (S. Ediciones Paraninfo, Ed.) Madrid. Obtenido de https://books.google.com.co/books?id=FfEfCB-hXCgC&printsec=frontcover&dq=programaci%C3%B3n+informatica&hl=es&sa=X&redir_esc=y#v=onepage&q=programaci%C3%B3n%20informatica&f=false

Fortinet. (s.f.). *Túnel dividido VPN*. Recuperado el 1 de junio de 2024, de www.fortinet.com: https://www.fortinet.com/lat/resources/cyberglossary/vpn-split-tunneling#:~:text=Una%20VPN%20proporciona%20a%20los,ubicaci%C3%B3n%2C%20si%20as%C3%AD%20se%20desea.

Fortra. (25 de junio de 2022). *Qué es el escaneo de vulnerabilidades y cómo funciona*. Obtenido de www.fortra.com: https://www.fortra.com/es/blog/escaneo-vulnerabilidades

Fruhlinger, J. (12 de febrero de 2020). *Equifax data breach FAQ: What happened, who was affected, what was the impact?* Obtenido de www.csoonline.com: https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html

Fuentes, B. A., Michelle, C., & Jenkins, M. (2021). Application of Process Metrics for Software Testing: A Case Study. *2021 IEEE V Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI)* (págs. 1-6). San José, Costa Rica: IEEE. doi:https://ieeexplore-ieee-org.bibliotecavirtual.unad.edu.co/stamp/stamp.jsp?tp=&arnumber=9794340

Galiana, P. (noviembre de 15 de 2023). *Qué es la ciberseguridad, por qué es importante y cómo convertirte en experto.* Obtenido de www.iebschool.com: https://www.iebschool.com/blog/que-es-ciberseguridad-tecnologia/

GlobalSuite Solutions. (22 de septiembre de 2023). *¿Qué es la norma ISO 27001 y para qué sirve?* Obtenido de https://www.globalsuitesolutions.com/es/que-es-la-norma-iso-27001-y-para-que-sirve/?gad_source=1&gclid=Cj0KCQjwvpy5BhDTARIsAHSilykZQoaJ_YB_XY18Wwpgot-XJsS3NrJ9-7vpHsMNdmfc0_FUzDbdmYEaAtQiEALw_wcB

Grupo Atico34. (s.f.). *Lista blanca, gris o negra. Qué son y diferencias*. Recuperado el 2 de junio de 2024, de protecciondatos-lopd.com: https://protecciondatos-lopd.com/empresas/lista-blanca-gris-negra/

Grzegorz, S., Mazurczyk, W., & Karpiński, A. (13 de junio de 2019). Security Assurance in DevOps Methodologies and Related Environments. *International Journal of Electronics and Telecommunications, 65*(2). doi:10.24425/ijet.2019.126303

Gutiérrez, I. (8 de septiembre de 2023). *Ransomware: Aprende cómo funciona y cuida tu negocio.* Obtenido de https://a3sec.com/blog/que-es-un-ransomware

Halfond, W. G., Viegas, J., & Orso, A. (2006). A Classification of SQL Injection Attacks and Countermeasures. *IEEE International Symposium on Secure Software Engineering*.

Heredero, C. d. (2006). *Dirección y gestión de los sistemas de información en la empresa: Una visión integradora* (Segunda Ed ed.). (E. Editorial, Ed.) Madrid. Obtenido de https://books.google.com.co/books?id=OqlSVYn0fI0C&pg=PA181&dq=iso+27000&hl=es&sa=X&ved=2ahUKEwjW0uac5Pr6AhW1mYQIHbVyCskQ6wF6BAgNEAE#v=onepage&q=iso%2027000&f=false

Iberdrola. (25 de septiembre de 2022). *Ciberataques: Ataques cibernéticos: ¿cuáles son los principales y cómo protegerse de ellos?* Obtenido de www.iberdrola.com: https://www.iberdrola.com/innovacion/ciberataques

IBM. (s.f.). *¿Qué son las pruebas de penetración?* Recuperado el 1 de junio de 2024, de www.ibm.com: https://www.ibm.com/mx-es/topics/penetration-testing

IBM. (2021). *www.ibm.com.* Obtenido de www.mainstream-tech.com: https://www.ibm.com/downloads/cas/EOPJ7R4K

IBM. (2022). *¿Qué es infraestructura de TI?* . Obtenido de www.ibm.com: https://www.ibm.com/mx-es/topics/infrastructure

IBM. (10 de agosto de 2022). *¿Qué es una prueba de software?* Obtenido de www.ibm.com: https://www.ibm.com/mx-es/topics/software-testing

Incibe. (4 de julio de 2019). *¿Qué es el pentesting? Auditando la seguridad de tus sistemas*. Obtenido de www.incibe.es: https://www.incibe.es/protege-tu-empresa/blog/el-pentesting-auditando-seguridad-tus-sistemas

ISO/IEC 25010. (2011). *Calidad de Sotware y Datos*. Obtenido de iso25000.com: https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO/IEC 27001. (2018). *NORMA ISO 27001*. Obtenido de https://www.normaiso27001.es/

ISO/IEC 27034. (2011). *www.iso.org*. Obtenido de Plataforma de navegación en línea (OBP): https://www.iso.org/obp/ui/es/#iso:std:iso-iec:27034:-1:ed-1:v1:en

ISO/IEC/IEEE 12207. (2017).

Jiménez, M., & César , S. (s.f.). *¿Cómo Evitar Ataques por Cantidad? Medidas de seguridad*. Recuperado el 2 de junio de 2024, de todoforti.net: https://todoforti.net/ciberseguridad-tecnica/como-evitar-ataques-por-cantidad-medidas-de-seguridad/

Joshi, J. (8 de enero de 2013). *Secure Software Development Models/Methods.* Obtenido de https://www.sis.pitt.edu/jjoshi/courses/IS2620/Spring13/Lecture1.pdf

Joshua, C. (1 de agosto de 2024). *¿Qué es el vishing? Definición, métodos de ataque y prevención.* Obtenido de Avast: https://www.avast.com/es-es/c-what-is-vishing

Junta de Andalucia. (2022). *Codificación y Validación de entrada/salida*. Recuperado el septiembre de 2022, de http://www.juntadeandalucia.es: http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/codificacion-y-validacion-entradasalida

Kaspersky. (s.f.). *¿Qué es el cifrado de datos? Definición y explicación*. Recuperado el 1 de junio de 2024, de atam.kaspersky.com: https://latam.kaspersky.com/resource-center/definitions/encryption

Kaspersky. (s.f.). *¿Qué es el robo de datos y cómo evitarlo?* Recuperado el 26 de octubre de 2024, de https://latam.kaspersky.com/resource-center/threats/data-theft?srsltid=AfmBOooPgKCVgjPJKA8-SvTgP5D9QtZtNNauCWQPJ6wVeI71-3OEFa9S

Kaspersky. (s.f.). *¿Qué es el smishing y cómo puedes protegerte de esta amenaza?* Recuperado el 10 de octubre de 2024, de https://latam.kaspersky.com/resource-center/threats/what-is-smishing-and-how-to-defend-against-it

Kaspersky. (s.f.). *¿Qué es la ingeniería social?* Recuperado el 26 de octubre de 2024, de https://latam.kaspersky.com/resource-center/definitions/what-is-social-engineering?srsltid=AfmBOopzGHixLTDnsZzquZoOVjhthYvGS8A4pBpAnvQSHyHB-KA64Q2C

Kaspersky. (s.f.). *¿Qué es un ataque de whaling?* Recuperado el 15 de octubre de 2024, de https://latam.kaspersky.com/resource-center/definitions/what-is-a-whaling-attack?srsltid=AfmBOor8i4IccqgejY6lJmGQKgWZiz3SG_IbeThKj5tm258HC0j2aXmQ

Kaspersky. (27 de septiembre de 2022). *¿Qué son los ataques DDoS?* Obtenido de latam.kaspersky.com: https://latam.kaspersky.com/resource-center/threats/ddos-attacks

Kaspersky. (s.f.). *Identificación de ransomware: en qué se diferencian los troyanos de cifrado.* Recuperado el 15 de octubre de 2024, de https://latam.kaspersky.com/resource-center/threats/ransomware-attacks-and-types?srsltid=AfmBOoqFdTAMrvoOWPV7fONAo-LnqKwt8BMUWz2xUeGrYwqRajrqs9m5

La Nota Económica. (27 de mayo de 2022). *En Colombia las empresas están siendo atacadas en promedio 2.387 veces por semana*. Obtenido de lanotaeconomica.com.co: https://lanotaeconomica.com.co/movidas-empresarial/en-colombia-las-empresas-estan-siendo-atacadas-en-promedio-2-387-veces-por-semana/

Latto, N. (12 de febrero de 2020). *¿Qué es el adware y cómo puede prevenirlo?* Obtenido de Avast: https://www.avast.com/es-es/c-adware

Lindemulder, G., & Kosinski, M. (12 de agosto de 2024). *¿Qué es la ciberseguridad?* Obtenido de IBM: https://www.ibm.com/es-es/topics/cybersecurity

López, Á. (30 de mayo de 2022). *Desarrollo seguro de software*. Obtenido de itcl.es: https://itcl.es/blog/desarrollo-seguro-de-software/

Lozano, P. A. (9 de abril de 2024). *Escaneo de vulnerabilidades: Herramientas y técnicas.* Obtenido de openwebinars.net: https://openwebinars.net/blog/escaneo-de-vulnerabilidades/

Mallón, X. (24 de abril de 2024). *Pruebas de seguridad de aplicaciones dinámicas (DAST)*. Recuperado el 1 de junio de 2024, de keepcoding.io: https://keepcoding.io/blog/pruebas-de-seguridad-de-aplicaciones-dinamicas/

Mallón, X. (9 de mayo de 2024). *Pruebas de seguridad de aplicaciones estáticas (SAST)*. Recuperado el 1 de junio de 2024, de keepcoding.io: https://keepcoding.io/blog/pruebas-de-seguridad-de-aplicaciones-estaticas/

Mantovani, V. (2019). Autenticación de Múltiples factores (MFA). Obtenido de http://bibliotecadigital.econ.uba.ar/download/tpos/1502-1524_MantovaniV.pdf

Martínez, K. J., Pacheco-Meneses, J., & Zuñiga-Silgado, I. (2009). *Firewall-linux: Una solución de seguridad informática para pymes (pequeñas y medianas empresas).* Obtenido de www.redalyc.org: https://www.redalyc.org/pdf/5537/553756879003.pdf

Martínez-Vargas, D. P. (25 de noviembre de 2021). *Ataques cibernéticos más frecuentes en las mipymes de Colombia durante el periodo 2020 - 2021 de la pandemia covid-19.* Obtenido de Repositorio Institucional UNAD: https://repository.unad.edu.co/handle/10596/51471

McAfee. (10 de agosto de 2022). *¿Qué es malware?* Obtenido de www.mcafee.com: https://www.mcafee.com/es-mx/antivirus/malware.html

Microsoft. (11 de agosto de 2022). *¿Qué es DevOps?* Obtenido de azure.microsoft.com: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/

Microsoft. (septiembre de 2022). Cuáles son las prácticas de Microsoft SDL? *www.microsoft.com*, págs. https://www.microsoft.com/en-us/securityengineering/sdl.

Moes, T. (julio de 2023). *¿Qué es una auditoría de seguridad? Todo sobre ello*. Obtenido de softwarelab.org: https://softwarelab.org/es/blog/que-es-una-auditoria-de-seguridad/

Morales Molina , C. D., Hernandez Suarez, A., Sanchez Perez, G., Toscano Medina, L. K., Perez Meana , H., Mercado , J. O., . . . Garcia Villalba, L. J. (1 de mayo de 2021). A Dense Neural Network Approach for Detecting Clone ID Attacks on the RPL Protocol of the IoT. *21*(9). doi:10.3390/s21093173

Moreno Marín, J. E., & Coronado Sánchez, P. C. (2 de octubre de 2020). Modelo base de conocimiento para auditorías de seguridad en servicios web con inyección SQL. *Ingeniería*, *25*(3), 264–283. doi:https://doi.org/10.14483/23448393.15740

NeoAttack. (27 de agosto de 2022). *Software*. Obtenido de neoattack.com: https://neoattack.com/neowiki/software

Nivia, R. M., Cortés, P. E., & Rojas, A. E. (4 de julio de 2018). Implementation Phase Methodology for the Development of Safe Code in the Information Systems of the Ministry of Housing, City, and Territory. En C. Springer (Ed.), *Computational Science and Its Applications – ICCSA 2018, 10961*, págs. 34-49. doi:https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-3-319-95165-2_3

Norton. (8 de agosto de 2018). *¿Qué es un virus informático?* Obtenido de mx.norton.com: https://mx.norton.com/blog/malware/what-is-a-computer-virus

NVD. (4 de julio de 2014). *CVE-2014-0160.* Obtenido de nvd.nist.gov: https://nvd.nist.gov/vuln/detail/cve-2014-0160

NVD. (18 de noviembre de 2015). *CVE-2015-4852.* Obtenido de nvd.nist.gov: https://nvd.nist.gov/vuln/detail/CVE-2015-4852

OWASP. (2021). *OWASP Top 10:2021. A03:2021 – Inyección*. Obtenido de owasp.org: https://owasp.org/Top10/es/A03_2021-Injection/

OWASP. (2021). *OWASP Top Ten 2021. Open Web Application Security Project*. Obtenido de owasp.org: https://owasp.org/www-project-top-ten/.

OWASP. (Septiembre de 2022). *OWASP Application Security Verification Standard*. Obtenido de owasp.org: https://owasp.org/www-project-application-security-verification-standard/

Owasp. (s.f.). *Prácticas De Codificación Segura - Guía Rápida De Referencia*. Recuperado el 31 de mayo de 2024, de https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-es/01-introduction/05-introduction

Parra, J. (19 de diciembre de 2023). *Evita consecuencias indeseables: Gestión de permisos y roles para evitar riesgos*. Obtenido de amisando.es: https://amisando.es/evita-consecuencias-indeseables-gestion-de-permisos-y-roles-para-evitar-riesgos/?expand_article=1

Proofpoint. (s.f.). *¿Qué es phishing?* Recuperado el 10 de octubre de 2024, de https://www.proofpoint.com/es/threat-reference/phishing

Proofpoint. (s.f.). *4 errores que aumentan la vulnerabilidad en seguridad de las empresas a las amenazas internas.* Recuperado el 26 de octubre de 2024, de

https://www.proofpoint.com/es/blog/insider-threat-management/4-security-missteps-make-employees-and-companies-vulnerable-insider

Rafael, R. L. (23 de julio de 2012). *Norma ISO/IEC 15408, Common Criteria.* Obtenido de peritoit.com: https://peritoit.com/2012/07/23/norma-isoiec-15408-common-criteria/

RedHat. (11 de mayo de 2022). *La integración y la distribución continuas (CI/CD).* Obtenido de www.redhat.com: https://www.redhat.com/es/topics/devops/what-is-ci-cd

Redhat. (15 de marzo de 2023). *¿Qué es DevSecOps?* Obtenido de www.redhat.com: https://www.redhat.com/es/topics/devops/what-is-devsecops

Riesco, S. (7 de mayo de 2021). *¿Para qué sirve la norma ISO 27001 de Seguridad de la Información?* Obtenido de www.formazion.com: https://www.formazion.com/noticias_formacion/para-que-sirve-la-norma-iso-27001-de-seguridad-de-la-informacion-org-6592.html

Ristic, I. (2014). *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications* (ilustrada, revisada ed.). Obtenido de https://books.google.com.co/books?id=fQOLBAAAQBAJ&hl=es&source=gbs_navlinks_s

Roldán, P. N. (21 de Agosto de 2017). *Tecnología: Qué es, usos y ejemplos*. Obtenido de Economipedia.com: https://economipedia.com/definiciones/tecnologia.html

Romero Castro Martha Irene, F. M. (2018). *Introducción a la seguridad informática y el análisis de vulnerabilidades* (1 ed. ed.). España, Alicante. Obtenido de https://d1wqtxts1xzle7.cloudfront.net/45233653/De_la_Gestin_de_Seguridad_en_el_Ciclo_de20160430-21010-q9u3k3-with-cover-page-

v2.pdf?Expires=1666680985&Signature=JemvnM6GrKtzZxtha0Rd4a8DXlqfbpUcUVjiyuRqU3TkP7xbA1OeOnFYLHUFfkN30cSrTof~IQlrWIrwx19FzDIeFNRzTJ

Rubin, C. C. (8 de abril de 2024). *nyecciones SQL, CSRF y XSS.* Obtenido de ciberseguridad.comillas.edu: https://ciberseguridad.comillas.edu/inyecciones-sql-csrf-y-xss/

Sánchez-Sánchez, P. A., García-González, J. R., Triana, A., & Perez-Coronell, L. (2021). Medida del nivel de seguridad informática de las pequeñas y medianas empresas (PYMEs) en Colombia. *Información tecnológica*, págs. 121-128. Obtenido de http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642021000500121&lng=es&nrm=iso

Santos Chavez, J. J. (26 de agosto de 2022). *¿Qué es una vulnerabilidad informática y cómo protegerse?* Obtenido de www.deltaprotect.com: https://www.deltaprotect.com/blog/vulnerabilidad-informatica

SENTRIO. (10 de 11 de 2021). *¿Qué es DevSecOps? Integra la seguridad en DevOps*. Obtenido de https://sentrio.io/blog/que-es-devsecops-vs-devops/

Sevillano, F. (2 de octubre de 2017). *Las 7 causas más habituales de un ciberataque a empresas*. Obtenido de https://willistowerswatsonupdate.es/ciberseguridad/infografia-las-7-causas-mas-habituales-de-los-ciberataques-a-empresas/

SonicWall. (2022). *Informe de amenazas cibernéticas*. Obtenido de www.sonicwall.com: https://www.sonicwall.com/medialibrary/es/infographic/infographic-2022-sonicwall-cyber-threat-report-latam.pdf

Spiros Alexiou, Ph.D., CISA, CSX-F, & CIA. (1 de mayo de 2019). *Practical Patch Management and Mitigation.* Obtenido de www.isaca.org:

https://www.isaca.org/resources/isaca-journal/issues/2019/volume-3/practical-patch-management-and-mitigation

Stine, K., & Barrett, M. (2022). *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1 (Spanish translation), NIST Cybersecurity Framework International*. doi:https://doi.org/10.6028/NIST.CSWP.04162018es

Superintendencia de Industria y Comercio. (2012). *Ley 1581 de 2012*. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

tiThink. (13 de junio de 2018). *12 buenas prácticas para el desarrollo de software*. Obtenido de www.tithink.com: https://www.tithink.com/es/2018/06/13/12-buenas-practicas-para-el-desarrollo-software/

Ungoti. (5 de septiembre de 2022). *Ciclo de vida del desarrollo de software*. Obtenido de ungoti.com: https://ungoti.com/es/soluciones/desarrollo-de-software/sdlc/#:~:text=El%20ciclo%20de%20vida%20del,definici%C3%B3n%20de%20los%20requisitos%20hasta

UNIR. (18 de octubre de 2022). *Principios de la seguridad informática: consejos para la mejora de la ciberseguridad*. Obtenido de mexico.unir.net: https://mexico.unir.net/ingenieria/noticias/principios-seguridad-informatica/#:~:text=Proteger%20la%20informaci%C3%B3n%20significa%20garantizar,la%20disponibilidad%20de%20la%20informaci%C3%B3n.

Urbina, G. B. (2016). *Introducción a la seguridad informática.* (1 ed ed.). México. Obtenido de https://books.google.com.co/books?id=IhUhDgAAQBAJ&printsec=frontcover&dq=seguridad+informatica&hl=es-419&sa=X&redir_esc=y#v=onepage&q=seguridad%20informatica&f=false

VPN Unlimited. (s.f.). *Control de acceso fallido.* Recuperado el 27 de octubre de 2024, de https://www.vpnunlimited.com/es/help/cybersecurity/broken-access-control?srsltid=AfmBOoof0Xu_zN1_LwySLL4fj4k2i7kzXBIBLk6Rgr-Rs3v5JNVlkvMd

Wassermann, G., & Su, Z. (s.f.). Static Detection of Cross-Site Scripting Vulnerabilities. *30th International Conference on Software Engineering (ICSE).*, 171-180.

xelere. (27 de julio de 2022). *Etapas de un ataque de ransomware y cómo prevenirlo*. Obtenido de xelere.com/: https://xelere.com/2022/07/27/etapas-de-un-ataque-de-ransomware-y-como-prevenirlo/

Xu, Y., Fang, Y., Liu, Z., & Zhang, Q. (7 de noviembre de 2023). PWAGAT: Potential Web attacker detection based on graph attention network. *557*. Obtenido de https://www.sciencedirect.com/science/article/pii/S0925231223008482

Yoris, L. (8 de marzo de 2023). *Control de Acceso Basado en Roles (RBAC)*. Obtenido de blog.tecnetone.com: https://blog.tecnetone.com/control-de-acceso-basado-en-roles-rbac

Zapata, J. (octubre de 2022). *Causas de los ataques informáticos.* Obtenido de jzseguridadweb.blogspot.com: https://jzseguridadweb.blogspot.com/p/fishing.html

Zavala, Á., & Alvarado, I. (2018). The importance of good practices in the development of secure web applications and their repercussions for not implementing them. *2018 IEEE 38th Central America and Panama Convention (CONCAPAN XXXVIII)* (págs. 1-5). San Salvador: IEEE. doi:10.1109/CONCAPAN.2018.8596528.