Tipo de artículo: Artículo original

Estudio del desarrollo seguro del software

Study of secure software development

Osiris Pérez Lastra[1*], https://orcid.org/0000-0002-0949-3214
Dainys Gainza Reyes[2], https://orcid.org/0000-0002-6087-141X
Henry Raúl González Brito[3], https://orcid.org/0000-0002-3226-9210

[1] Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados. Ave 47 No. 1802 e/ 18A y 20, Playa, La Habana. osiris@citmatel.inf.cu
[2] Universidad de las Ciencias Informáticas (Dirección de Educación de Posgrado). dgainza@uci.cu
[3] Universidad de las Ciencias Informáticas (Departamento docente de Ciberseguridad, Facultad 2). henryraul@uci.cu

*** Autor para correspondencia:** osiris@citmatel.inf.cu

Resumen

El crecimiento global de las redes y la gran demanda del uso de las aplicaciones informáticas, impulsados por la innovación tecnológica, ha propiciado la satisfacción de las necesidades de todas las esferas de la economía y la sociedad, así como el aumento del bienestar y de la calidad de vida. Sin embargo, este desarrollo tecnológico emergente ha acrecentado los ataques informáticos los cuales tienen consecuencias negativas y comprometen la integridad, confidencialidad y disponibilidad de la información, datos o servicios. Por este motivo resulta necesario garantizar la seguridad en todo el proceso de desarrollo del software. El objetivo de este trabajo por tanto se enfoca en realizar un estudio de las principales metodologías de desarrollo seguro, buenas prácticas de seguridad, principios de diseño seguro y en Scrum, como metodología ágil. Se utilizaron para la investigación, los métodos analíticos – sintético, análisis histórico – lógico, análisis documental y la encuesta. Se considera que este estudio puede ser provechoso para los especialistas involucrados en el desarrollo de software pues le brindará un estado del arte del desarrollo seguro.

Palabras clave: seguridad, metodologías de desarrollo seguro, principios de diseño seguro, Scrum

Abstract

The global growth of networks and the great demand for the use of computer applications, driven by technological innovation, has led to the satisfaction of the needs of all spheres of the economy and society, as well as the increase in well-being and quality of life. However, this emerging technological development has increased computer attacks which have negative consequences and compromise the integrity, confidentiality and availability of information, data or services. For this reason, it is necessary to guarantee security throughout the software development process. The objective of this work is therefore focused on carrying out a study of the main secure development methodologies, good security practices, secure design principles and Scrum, as an agile methodology. The analytical - synthetic methods, historical - logical analysis, documentary analysis and the survey were used for the investigation. It is considered that this study can be beneficial for specialists involved in software development as it will provide a state of the art of secure development.

Keywords: *security, secure development methodologies, secure design principles, Scrum*

Recibido: 20/09/2022
Aceptado: 10/12/2022
En línea: 30/01/2023

Introducción

Con la creciente demanda de las Tecnologías de la Información y las Comunicaciones (TIC) debido a las innumerables soluciones que existen para satisfacer las necesidades y servicios de los clientes, las empresas han tenido que evolucionar y adoptar nuevos cambios. Los cambios que se dan en este ámbito requieren agilidad y el uso de metodologías que permitan adaptar de manera rápida las condiciones de los nuevos proyectos de desarrollo, en los que la eficiencia y la flexibilidad generan menores costes de producción, y una alta calidad (García, 2019)

Los métodos ágiles han adquirido gran popularidad y es Scrum el marco más común para el desarrollo en la mayoría de los países (Dingsoeyr, Falessi, & Power, 2019; Zumba Gamboa, 2018; State of Agile, 2022). Tienen como objetivo hacer las entregas del proyecto menos complicadas, obviando un poco los detalles de la documentación y no siempre se tiene a la seguridad como una prioridad fundamental (Warden & Shore, 2021; Martin, 2002).

El uso masivo de aplicaciones informáticas unido al valor de la información que gestionan la han convertido en objetivo permanente de delitos cibernéticos (Jamil, Asif, Ashraf, Mehmood, & Mustafa, 2018; Kamuni, Asfia, Sheikh, & Patel, 2019) , por lo que la seguridad se convierte en un requisito vital pues; demanda una especial atención debido al incremento de las amenazas atribuibles a la posibilidad de "explotar vulnerabilidades" presentes en el software poniendo en riesgo a las organizaciones y al cumplimiento de su misión.

Tradicionalmente los mecanismos defensivos de las empresas para considerar la seguridad están centrados en antivirus, cortafuegos, políticas de acceso, mecanismos de cifrados y demás. Por otro lado, la seguridad en los sistemas informáticos se realiza de forma tardía y abordada cuando se termina el producto a través de pruebas de penetración. Las vulnerabilidades encontradas son solucionadas y puestas a disponibilidad del usuario mediante actualizaciones o comúnmente llamadas parches. Esta estrategia no es eficaz pues conduce a modificaciones significativas y costosas y el usuario pudiera no aplicar las actualizaciones.

Según Gary McGraw, la seguridad del software se trata de crear software seguro: diseñar software para que sea seguro, asegurarse de que el software sea seguro y educar a los desarrolladores, arquitectos y usuarios sobre cómo crear cosas seguras (Mohammed , Niazi, & Mahmood, 2017), sin comprometer la integridad, la confidencialidad y la disponibilidad de la información, y continúe funcionando incluso bajo ataques maliciosos (Abdul Karim, Albuolayan,

& Saba, 2016). Esto requiere abordar los desafíos de la seguridad durante todo el ciclo de desarrollo de software (Rizwan Ahmed, 2007).

La gran necesidad de implementar productos seguros que brinden confiabilidad y calidad a sus usuarios sin comprometer la información y privacidad se hace eminente, tal es un reporte hecho por el Observatorio Español de Delitos Informáticos de Latinoamérica (OEDI) en el año 2021, arrojó 267,011 por fraude informático; 5,342 por acceso e interceptación ilícita; 2,138 por interferencia en los datos y en el sistema; 10,476 por falsificación informática, entre otros incidentes delictivos (OEDI, 2021). De igual manera, los informes periódicos emitidos por organismos internacionales y compañías líderes en el entorno de la ciberseguridad evidencian un aumento sostenido de delitos informáticos (Positive Tecnologies, 2021; Kaspersky, 2021; INCIBE, 2021)

A partir de lo señalado anteriormente se evidencia que los datos y la información se le debe dar importancia en cualquier entorno y debe ser una premisa en las aplicaciones informáticas brindar confiabilidad y calidad sin comprometer la seguridad y privacidad de la información. En la presente investigación se exponen elementos referenciados a nivel internacional asociados a principios de diseño seguro, buenas prácticas de seguridad y metodologías de desarrollo seguro con el objetivo de brindar a los especialistas involucrados en el desarrollo de software una revisión de estado del arte sobre la temática.

Materiales y métodos

Para la obtención de información de la investigación se emplearon varios métodos científicos, a continuación, se mencionan y se explica su uso:

Métodos teóricos:

1. Analítico – Sintético: se emplea en el análisis de los elementos que brinda la bibliografía sobre la característica seguridad en el ciclo de desarrollo del software, posibilitando identificar factores comunes para obtener resultados sobre la base de los datos previamente analizados.
2. Análisis histórico – lógico: se utiliza en el estudio la trayectoria histórica sobre la característica de seguridad en el ciclo de desarrollo del software que han sido plasmadas por diferentes autores, metodologías, y principios, de forma tal que se proporcionen una base de conocimiento a partir de lo histórico e incorporar el empleo de actividades de seguridad que garanticen mitigar las vulnerabilidades del software.

Métodos empíricos:

1. Análisis documental: Se utiliza para el estudio de diferentes documentos relacionados con la seguridad en ciclo de desarrollo del software para favorecer su objetividad.
2. Encuesta: Se realizaron encuestas en la Unidad Básica Empresarial de Desarrollo de Software de la Empresa CITMATEL para obtener información y experiencias de los especialistas, sobre el uso de la seguridad en el ciclo de desarrollo del software.

Resultados y discusión

Metodologías Ágiles

Las metodologías de desarrollo de software proveen un marco de trabajo para planificar, ejecutar, y gestionar el proceso de desarrollo de sistemas de software (Vijayasarathy & Butler, 2015). Adicionalmente, constituye una guía para el equipo de desarrollo, equipo de calidad, jefe de proyecto; definiendo el ciclo de vida del software desde la obtención de los requisitos, análisis, diseño, implementación, pruebas hasta el despliegue.

Las metodologías ágiles presentan la particularidad de satisfacer a los clientes de forma continua mediante entregas frecuentes y tempranas, son flexibles a los cambios, altamente colaborativos, predomina las interacciones entre los individuos ante los procesos y herramientas, se documenta lo necesario tras finalizar las implementaciones propuestas, incrementa la productividad. La utilización de enfoque iterativo e incremental, integración continua de código, y aceptación de cambios en los requerimientos de negocio ha permitido a las metodologías ágiles remplazar las metodologías tradicionales en negocios en donde las especificaciones no son claras o van cambiando con el tiempo, dependiendo de las necesidades del cliente.

Por otra parte, State of Agile publica reportes sobre técnicas y prácticas ágiles. Estos reportes cubren una amplia gama de industrias de las comunidades de desarrollo de software y tienen en cuenta las tendencias, experiencias, desafíos, herramientas, técnicas e iniciativas DevOps en el entorno ágil. En la edición del año 2022 destacó a Scrum como la metodología más popular con una participación del 87% en mercado y combinadas con otras, ScrumBan obtuvo 27% y Scrum XP el 13% (State of Agile, 2022).

Scrum

Es un marco de trabajo para el desarrollo y mantenimiento de productos complejos empleado en el desarrollo de software, usa un enfoque iterativo e incremental para optimizar la previsibilidad y controlar el riesgo. Scrum está

fundamentado en la teoría de control de procesos empírico definiendo que el conocimiento viene de la experiencia y la toma de decisiones basada en el conocimiento (Sutherland & Schwaber, 2020).

Scrum define tres roles, los cuales son: El Scrum master, el dueño del producto o Product owner y el equipo de desarrollo o team. El scrum master es la persona que lidera el equipo asegurándose que el equipo cumpla las reglas y procesos de la metodología. El dueño del producto es el representante de los accionistas y clientes que usan el software. El equipo de desarrollo es el grupo de profesionales encargados de convertir la lista de requerimientos o también llamado Product Backlog en funcionalidades del software.

Eventos de Scrum

En Scrum el tiempo es considerado un limitante muy importante en la gestión del proyecto y para hacerle frente se propone un tiempo fijo y constante para cada evento (time-boxing). Lo anterior garantiza que los miembros del equipo empleen el tiempo necesario para una actividad en la cual se tiene claridad y por consiguiente trae como ventajas una alta velocidad de los equipos, eficiencia en el proceso de desarrollo y mejor aprovechamiento del tiempo.

- ✓ **Sprint (Ciclo):** Es el evento esencial de Scrum y contempla los demás eventos. Tiene un mes o menos de duración y aquí se lleva a cabo la creación de incrementos (producto terminado). Un nuevo ciclo comienza después de la conclusión del anterior. Durante el ciclo no se realizan cambios ya que comprometería el objetivo del ciclo.
- ✓ **Sprint Planning (Reunión de Planificación):** Tiene una duración determinada de acuerdo con el alcance del proyecto, y en este se define las funcionalidades, y los objetivos del desarrollo que se alcanzarán con este Sprint, así mismo se responde interrogantes que están relacionados a ¿qué será entregado? y ¿cómo se realizará el trabajo?
- ✓ **Daily Scrum (Scrum Diario):** Son reuniones de 15 minutos como mínimo en el que los equipos de trabajo intercambian ideas, actividades y establecen nuevos planes para ser ejecutados en las próximas 24 horas o más si llega a decidirse así. Permiten mejorar de manera significativa las comunicaciones que se deben dar entre los participantes. Ayuda a acelerar y promover el conocimiento entre los equipos de desarrolladores.
- ✓ **Sprint Review (Revisiones):** Estas se llevan a cabo al final de cada sprint, y sirven para realizar inspecciones a los productos entregados, así mismo se valida lo que se denomina la lista de producto (Product Backlog), esta se hace de manera informal y se destina a la presentación de retroalimentación de lo concluido por el equipo de trabajo.
- ✓ **Sprint Retrospective (Retrospectiva):** En este punto se crean planes de mejoras orientados a revisar, identificar y crear planes que ayuden a mejorar procesos, potencializar el equipo y ser más eficientes en el trabajo desarrollado.

Artefactos de Scrum

Son elementos o herramientas que permiten realizar la gestión en un proyecto de desarrollo ágil, los cuales se detallan a continuación:

- ✓ Product Backlog. Es el listado de tareas que engloba todo un proyecto. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio como del esfuerzo necesario para su desarrollo.
- ✓ Sprint Backlog. Es el grupo de tareas del product backlog que el equipo de desarrollo elige en el sprint planning junto con el plan para poder desarrollarlas. Debe ser conocido por todo el equipo, para asegurarse de que el foco debe estar en este grupo de tareas.
- ✓ Increment (Incremento). Es la parte de producto desarrollada en un sprint y que se encuentra en condiciones de ser entregada al cliente, o sea, terminada, probada y operativa. Su objetivo es cumplir con las medidas de calidad establecidas.

Desarrollo de software seguro

McGraw plantea que la seguridad del software se trata de crear software seguro: diseñar software para que sea seguro, asegurarse de que el software sea seguro y educar a los desarrolladores, arquitectos y usuarios sobre cómo crear cosas seguras (Mohammed , Niazi, & Mahmood, 2017) y que es la capacidad del software para resistir, tolerar y recuperarse de eventos que amenazan intencionalmente su confiabilidad (Al-Matouq, Mahmood, Alshayeb, & Niazi, 2020).

La seguridad debe estar integrada desde el surgimiento de un proyecto de desarrollo de software, resulta un error considerarla en las últimas etapas del ciclo de vida de desarrollo pues se tiene el riesgo de introducir nuevas vulnerabilidades.

Los desafíos de la implementación de la seguridad están relacionados con el triángulo de gestión de proyectos (alcance, tiempo, presupuesto), agrega complejidad a la aplicación (más compleja al uso del usuario y de implementar) y valor de retorno al negocio (demostrar que la seguridad reduce costos a la compañía)(Deane & Kraus, 2021).

A continuación, se mencionan algunas metodologías, principios y buenas prácticas de desarrollo de software seguro.

Siete Puntos de Contactos

Proporciona un conjunto de mejores prácticas agrupadas en siete puntos de contacto que pueden ser aplicadas a varios artefactos de desarrollo de software y no están limitadas para un ciclo de desarrollo en particular. Constituyen una combinación de actividades constructivas (diseño, la defensa y la funcionalidad) y destructivas (ataques, exploits).
Estos son los puntos de contacto, en el orden de efectividad (McGraw, 2004):

1. Revisión de código.
2. Análisis de riesgo arquitectónico.
3. Pruebas de penetración.
4. Pruebas de seguridad basadas en riesgos.
5. Casos de abuso.
6. Requerimientos de seguridad.
7. Operaciones de seguridad.

Principios de Seguridad

Los principios de seguridad pueden guiar el diseño e implementación de software sin fallas de seguridad. Si en el proceso de desarrollo no se considera o se está violando un principio de seguridad, esta violación es un síntoma de una potencial falla en el proceso de desarrollo por lo cual es necesario revisar cuidadosamente el proceso para que la falla sea controlada (Adelyar & Norta, 2016).

Microsoft se refiere a estos principios como SD3+C: Seguro por diseño, Seguro por defecto, Seguro en implementación y Comunicaciones. Las breves definiciones de estos principios son (Howard & Lipner, 2006):

- ✓ **Seguro por Diseño:** El software tiene que ser estructurado, diseñado e implementado para a resistir los ataques y protegerse él mismo y la información que este procesa.
- ✓ **Seguro por Defecto:** Para minimizar el daño cuando un atacante hace blanco en alguna vulnerabilidad, el estado predeterminado del software debe elegir las opciones más seguras.
- ✓ **Seguro en el Despliegue:** Se debe incluir con el software información y herramientas que ayuden a los administradores y a los usuarios a utilizar este software con seguridad. Las actualizaciones deben ser fáciles de distribuir y debe existir un sistema de respuesta relativamente rápido frente a la aparición de nuevas amenazas.
- ✓ **Seguro en las Comunicaciones:** El equipo de desarrollo debe estar preparado para detectar las vulnerabilidades de seguridad del producto y comunicarse de manera abierta y responsable con los

usuarios y los administradores para ayudarles a tomar las medidas de protección adecuadas (como la actualización o la implementación de soluciones alternativas).

Por otra parte, William Stallings y Lawrie Brown en su libro Computer Security Principles and Practices enumeran los siguientes principios de diseño seguro (Stallings & Brown, 2018):

Defensa en profundidad: Consiste en crear diferentes capas de seguridad, de tal forma que si una falla, el sistema no se vea comprometido. Requiere diseñar distintas estrategias de defensa para una misma amenaza.

Fallo seguro: Consiste en que todos los fallos lleven a un estado del sistema que se considere seguro (sin pérdida de confidencialidad, integridad y disponibilidad).

Mínimo privilegio: Cada usuario o proceso debe poseer solamente, los mínimos privilegios posibles para llevar a cabo las tareas que le son permitidas. Los privilegios se deben otorgar por el mínimo tiempo posible.

Separación de privilegios: Consecución de cualquier actividad que se considere como crítica en el sistema debe requerir la participación de dos o más entidades. Tiene como objetivo también eliminar los puntos únicos de fallo.

Economía de mecanismo o simplicidad: Recomienda emplear soluciones menos complejas pues a mayor complejidad hay más oportunidad por un adversario de explotar las debilidades.

Supervisión: Antes de ejecutar cualquier tarea se debe comprobar que el usuario o proceso esté autorizado para ello. Para evitar problemas de sincronización se recomienda no utilizar cachés de autorización.

Diseño abierto: Se evitará la aplicación de la seguridad por oscuridad, esto ayudará a crear sistemas seguros desde el diseño. Este principio asegura que la publicación o revisión del diseño no impliquen de forma directa un incidente grave de seguridad.

Mínimo en común: Este principio desaconseja la utilización de un mismo mecanismo, aunque sea común a varios procesos o usuarios, si estos tienen diferentes niveles de privilegio.

Reutilización: Es preferible la utilización de componentes ya existentes y verificados que la creación de nuevos que puedan incrementar el riesgo de vulnerabilidades y superficie de ataque.

Aceptabilidad: Los mecanismos de seguridad del sistema se deben diseñar teniendo en cuenta la aceptabilidad por parte de sus usuarios. Si los usuarios tienen dificultades en usar las características de seguridad, buscarán mecanismos para saltárselas, haciéndolas inútiles.

Punto débil: La seguridad de todo el sistema dependerá de su punto más débil.

Howard y LeBlanc en su libro *Writing Secure Code* proponen los principios de: aprender de los errores, minimizar la superficie de ataque, defensa en profundidad, asumir que los sistemas externos son inseguros, contar con un plan de fallo, no depender solo de la seguridad por oscuridad, no mezclar código y datos, así como, corregir los problemas de seguridad correctamente (Howard & LeBlanc, 2003).

Welivesecurity menciona diez consejos para lograr un diseño seguro: ningún componente es confiable hasta demostrar lo contrario, delinear mecanismos de autenticación difíciles de eludir, autorizar además de autenticar, separar datos de instrucciones de control, validar todos los datos explícitamente, utilizar criptografía correctamente, identificar datos sensibles y cómo se los debería gestionar, considerar siempre a los usuarios del sistema, tener presente que la integración de componentes cambia la superficie de ataques y considerar cambios futuros en objetos y actores. (welivesecurity, 2015)

NIST 800-64

Describe un conjunto de actividades de seguridad en cada una de sus cinco fases:

Tabla 1. Fases y actividades del NIST 800-64.

Elaboración propia. Tomado de https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-64r2.pdf

Inicio	Adquisición/Desarrollo	Implementación/Evaluación	Mantenimiento/Operaciones	Eliminación
✓Inicio de la planificación de la seguridad. ✓Categorización del sistema de información. ✓Análisis de impacto de negocio. ✓Análisis de impacto de la privacidad. ✓Aseguramiento del uso de procesos de	✓Evaluación inicial del riesgo. ✓Seleccionar y documentar los controles de seguridad. ✓Diseño de una arquitectura segura. ✓Implementación de control de seguridad en el diseño del sistema. ✓Documentación de seguridad. ✓Análisis de fortalecimiento de la seguridad.	✓Creación de un plan detallado para la certificación y acreditación del sistema. ✓Integrar seguridad en los entornos establecidos. ✓Evaluar la seguridad del sistema. ✓Acreditación de la seguridad. ✓Autorizar a la aplicación el almacenamiento,	✓Revisar la preparación operativa para manejar modificaciones no planificadas al sistema. ✓Realizar actividades de administración y control de la configuración para garantizar la consideración de posibles impactos de seguridad debido a cambios específicos en el sistema. ✓Realizar un monitoreo continuo para garantizar la efectividad de los	✓Construir y ejecutar un plan de eliminación / transición. ✓Asegurar la protección de la información (copia de seguridad) y los métodos de recuperación. ✓Requisitos legales relacionados con la retención de registros, al desechar sistemas. ✓Política de saneamiento de medios para evitar la divulgación de información no autorizada. ✓Política de eliminación de hardware y software.

desarrollo de sistemas de información seguro.	✓Documentos iniciales para la Certificación y Acreditación de Sistemas.	procesamiento y transferencia de la información.	controles de seguridad a lo largo del tiempo.	✓Política de cierre o desmontaje de sistema.

CLASP: Comprehensive Lightweight Application Security Process

CLASP es un modelo prescriptivo, basado en roles y buenas prácticas, que permite a los equipos de desarrollo implementar seguridad en cada una de las fases del SDLC en forma estructurada, medible y repetible. Estructuralmente, CLASP se descompone en 5 elementos:

A. CLASP View: 5 Perspectivas de alto nivel, que se desglosan en actividades que a su vez contienen componentes del proceso que se interconectan entre sí: Conceptos, roles, evaluación de actividades, implementación de actividades, vulnerabilidades.

B. CLASP Best Practices: Agrupación de 7 actividades de Seguridad: Programas institucionales de sensibilización, evaluar el rendimiento de las aplicaciones, obtener los requerimientos de seguridad, implementar prácticas de desarrollo seguro, construir procedimientos de solución de vulnerabilidades, definir y monitorear métricas, y publicar guías operacionales de seguridad.

C. CLASP Activities: 24 actividades diseñadas para permitir una fácil integración entre actividades de seguridad y el SDLC.

D. CLASP Resources: Ayudan a la planificación, ejecución y cumplimiento de las actividades relacionadas con la seguridad del software.

E. CLASP Taxonomy: Clasificación de alto nivel de 104 tipos de problemas o vulnerabilidades, divididos en 5 categorías de alto nivel: Errores de tipo y rangos, problemas del entorno, errores de tiempo y sincronización, errores de protocolos, y errores generales de lógica.

Microsoft SDL

El ciclo de vida de desarrollo de seguridad (SDL) es un proceso de control de seguridad orientado al desarrollo de software. Microsoft SDL introduce la seguridad y la privacidad en todas las fases del proceso de desarrollo, fases que contienen un conjunto de actividades de carácter obligatorias. La figura 1 muestra las fases y actividades.

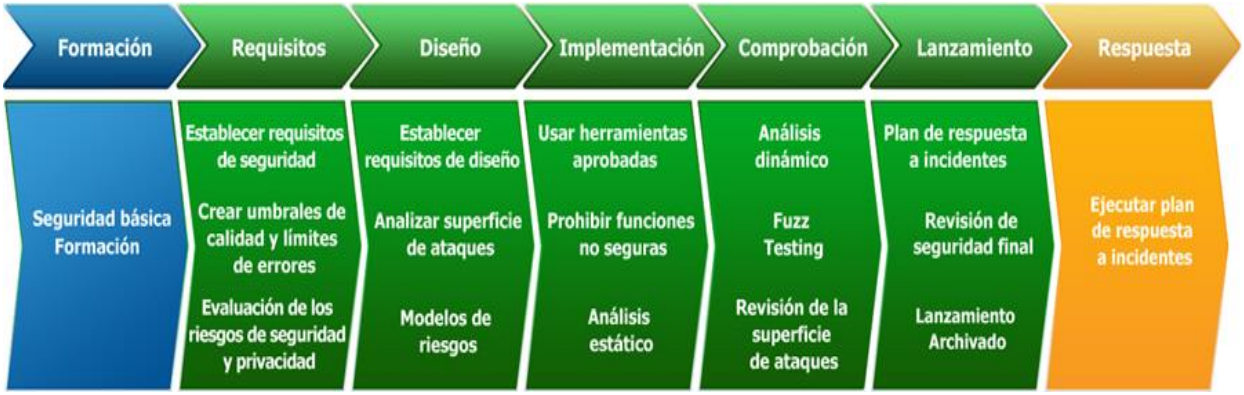

Figura 1. Fases y actividades de Microsoft SDL.

SAMM Software Assurance Maturity Model

Incluye los estándares ISO, PCI, COBIT e ISM3. Es rápido, fácil de desplegar y ayuda a mejorar el entorno de seguridad y el estado de la construcción de software. Se caracteriza por ser personalizable y adaptable en las organizaciones, según las necesidades y los niveles de seguridad que se deseen alcanzar. Proporciona los siguientes recursos:

- ✓ Evaluación de prácticas de desarrollo de software seguro.
- ✓ Construcción de un Software seguro que sea equilibrado.
- ✓ Mejoras en la aplicación que garantice la seguridad.
- ✓ Definición y medición de actividades de seguridad.

La figura 2 muestra una descripción de la metodología.

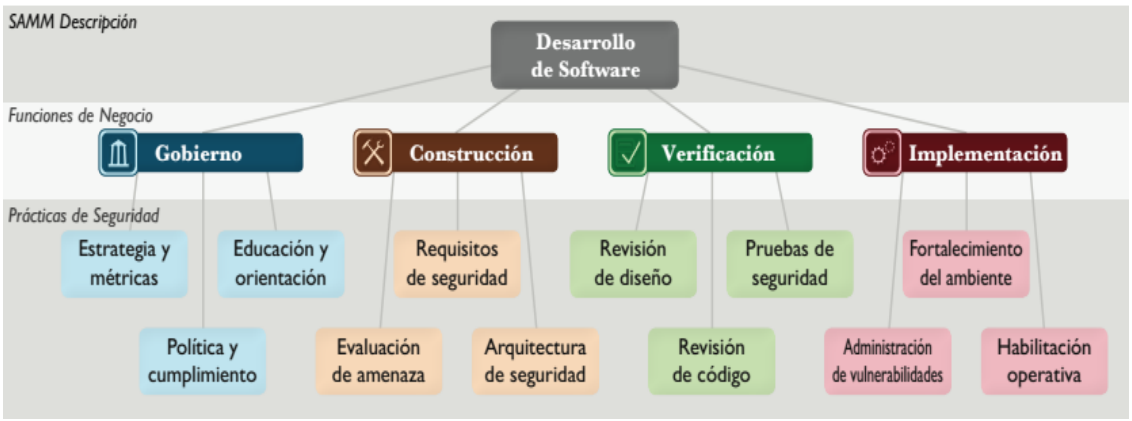

Figura 2. Descripción de la metodología SAMM.

Conclusiones

Los principios de diseño seguro, buenas prácticas de seguridad y metodologías de desarrollo seguro expuestos en este artículo se interrelacionan entre sí con un fin: en aumentar los esfuerzos en lograr una sólida seguridad del software.

La seguridad requiere de múltiples soluciones y de la aplicación de recursos durante el ciclo de vida del software.

Las metodologías ágiles deben integrar la seguridad en todos sus procesos.

Los especialistas involucrados en el desarrollo de aplicaciones informáticas no solo deben dominar los aspectos referentes al desarrollo seguro sino también deben tener conciencia para sí aplicar y compartir las mejores prácticas a lo largo del ciclo de vida de desarrollo de software que conlleva a un producto con mayor calidad, más resistente a ataques y proporciona satisfacción al cliente.

Como trabajo futuro se publicará un estudio que incorpore la seguridad en los procesos de Scrum en el entorno nacional.

Conflictos de intereses

Los autores autorizan la distribución y uso de su artículo.

Contribución de los autores

1. Conceptualización: Osiris Pérez Lastra, Dainys Gainza Reyes y Henry Raúl González Brito.
2. Obtención de Información: Osiris Pérez Lastra, Dainys Gainza Reyes y Henry Raúl González Brito.
3. Análisis formal: Osiris Pérez Lastra.
4. Investigación: Osiris Pérez Lastra.
5. Metodología: Dainys Gainza Reyes y Henry Raúl González Brito.
6. Recursos: Osiris Pérez Lastra, Dainys Gainza Reyes y Henry Raúl González Brito.
7. Supervisión: Dainys Gainza Reyes y Henry Raúl González Brito.
8. Validación: Osiris Pérez Lastra.
9. Visualización Osiris Pérez Lastra.
10. Redacción–borrador original: Osiris Pérez Lastra.
11. Redacción – revisión y edición: Dainys Gainza Reyes y Henry Raúl González Brito.

Financiamiento

La investigación no requirió fuente de financiamiento externa.

Referencias

R. Vijayasarathy, L., & W. Butler, C. (2016). Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software, 33*(5), 86-94. doi:10.1109/MS.2015.26

Abdul Karim, N. S., Albuolayan, A., & Saba, T. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security and Communication Networks, 9*(18), 5333-5345. doi:https://doi.org/10.1002/sec.1700

Adelyar, S. H., & Norta, A. (2016). Towards a secure agile software development process. *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)* (págs. 101-106). Lisbon,Portugal: IEEE. doi:10.1109/QUATIC.2016.028

Al-Matouq, H., Mahmood, S., Alshayeb, M., & Niazi, M. (2020). A Maturity Model for Secure Software Design: A Multivocal Study. *IEEE Access, 8*, 215758-215776. doi:10.1109/ACCESS.2020.3040220

CLASP. (2005). *Common Weakness Enumeration. The Clasp Application Security Process.* Obtenido de https://cwe.mitre.org/documents/sources/TheCLASPApplicationSecurityProcess.pdf

Deane, A., & Kraus, A. (2021). *The Official (ISC)2 CISSP CBK Reference.*

Dingsoeyr, T., Falessi, D., & Power, K. (2019). Agile Development at Scale: The Next Frontier. *IEEE Software, 36*(2). doi:10.1109/MS.2018.2884884

García, L. A. (9 de Agosto de 2019). *LN Creatividad y Tecnología Blog*. Obtenido de https://www.luisan.net/blog/transformacion-digital/que-son-las-metodologias-agiles

Howard, M., & LeBlanc, D. (2003). *Writing Secure Code.* Microsoft Press. Obtenido de https://books.google.com.cu/books?id=_7LEW8VHZk4C

Howard, M., & Lipner, S. (2006). *The security development lifecycle* (Vol. 8). Microsoft Press Redmond.

INCIBE. (2021). *Instituto Nacional de Ciberseguridad*. Obtenido de https://www.incibe.es/sala-prensa/notas-prensa/incibe-gestiona-mas-100000-incidentes-ciberseguridad-durante-2021

JAMIL, A., ASIF, K., ASHRAF, R., MEHMOOD, S., & MUSTAFA, G. (2018). A Comprehensive study of Cyber Attacks & Counter Measures for web systems. *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, (págs. 1-7). Obtenido de https://dl.acm.org/doi/abs/10.1145/3231053.3231116

KAMUNI, V., ASFIA, U., SHEIKH, A., & PATEL, D. (2019). Secure energy market against cyber attacks using blockchain. *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 1792-1797.

KASPERSKY. (2021). Obtenido de https://statistics.securelist.com/

MARTIN, R. C. (2002). *Agile software development: principles, patterns, and practices* (1 ed.). Prentice Hall PTR.

MCGRAW, G. (2004). Software Security. *IEEE Software & Privacy, 2*(2), 80-83. doi:10.1109/MSECP.2004.1281254

MICROSOFT SECURITY DEVELOPMENT LIFE CYCLE *(SDL)*. (s.f.). Obtenido de https://www.microsoft.com/en-us/securityengineering/sdl

MOHAMMED , N., NIAZI, M., & MAHMOOD, S. (2017). Exploring software security approaches in software development lifecycle: A systematic mapping study. *Computer Standards & Interfaces, 50*, 107-115. doi:https://doi.org/10.1016/j.csi.2016.10.001

NIST. (s.f.). *National Institute of Standards and Technology. SP 800-64 Rev. 2.Security Considerations in the System Development Life Cycle*. Obtenido de https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-64r2.pdf

OEDI. (2021). *Observatorio Español de Delitos Informáticos*. Obtenido de https://oedi.es/estadisticas/

OPEN SAMM SOFTWARE ASSURANCE MATURITY MODEL. (s.f.). Obtenido de http://www.opensamm.org/downloads/SAMM-1.0.pdf

POSITIVE TECNOLOGIES. (2021). Obtenido de https://www.ptsecurity.com/ww-en/analytics/

RIZWAN AHMED, S. (2007). Secure Software Development Identification of Security Activities and Their Integration in Software Development Lifecycle. *Master Thesis Software Engineering.* School of Engineering Blekinge Institute of Technology, Sweden. Obtenido de https://www.diva-portal.org/smash/get/diva2:833599/FULLTEXT01.pdf

STALLINGS, W., & BROWN, L. (2018). *Computer Security: Principles and Practice, Global Edition* (4 ed.). Pearson Education.

STATE OF AGILE. (2022). Obtenido de https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf

SUTHERLAND, J., & SCHWABER, K. (2020). *Scrum Guides.* Obtenido de https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

The Clasp Application Security Process. (2005).

WARDEN, S., & SHORE, J. (2021). *The Art of Agile Development.* O'Reilly Media.

WELIVESECURITY. (2015). *10 consejos de desarrollo seguro de aplicaciones*. Obtenido de https://www.welivesecurity.com/la-es/2015/03/12/10-consejos-desarrollo-seguro-de-aplicaciones/

WHITMAN, M. E., & MATTORD, H. J. (2021). *Principles of information security.* Cengage learning.

ZUMBA GAMBOA, J. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal, 3*(10), 20-33. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=6777227