INTEGRACIÓN DE LA GUÍA OWASP AL FRAMEWORK SCRUM EMPRESA DE DESARROLLO DE SOFTWARE SAC

ROBINSON TRIANA BECOCHE

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA - UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
PALMIRA
2024

INTEGRACIÓN DE LA GUÍA OWASP AL FRAMEWORK SCRUM EMPRESA DE DESARROLLO DE SOFTWARE SAC

ROBINSON TRIANA BECOCHE
rtrianab@unadvirtual.edu.co

Proyecto de grado – proyecto aplicado presentado para optar por el título de ESPECIALISTA EN SEGURIDAD INFORMÁTICA

Directora
ESP. YENNY ESTELLA NUÑEZ ALVAREZ
yenny.nunez@unad.edu.co

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
PALMIRA
2024

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

_____

Firma del presidente de Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Ciudad., Fecha sustentación

DEDICATORIA

Dedico este proyecto:

A Dios que con su bálsamo de sanación me ha dado tranquilidad, salud y luz en momentos difíciles, permitido enfocarme en mi propósito y estar en armonía en todos los aspectos de mi vida.

A mi madre Ana Cecilia Becoche, que con su dedicación y sacrificio me ha apoyado incondicionalmente para llegar a ser quien soy, porque nunca ha dejado de creer en mí y me ha enseñado que la dedicación y perseverancia son fundamentales para alcanzar nuestros objetivos.

A mi hija Valery Sofia Triana, mi amor eterno, mi más grande motivación en cada uno de los pasos que me llevan a cumplir mis metas, a quien quiero orientar con el ejemplo de honestidad, perseverancia, nobleza y empatía.

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 este logro no hubiera sido posible.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

pág.

GLOSARIO

APLICACIONES WEB: tipo de software al cual se accede a través de un navegador de internet, sus datos están alojados en un servidor web (en la nube).

CIBERDELINCUENTE: delincuente que busca explotar una vulnerabilidad de un sistema informático con fines maliciosos, ya sea por beneficios económicos, ideologías o venganza.

CIBERSEGURIDAD: es el proceso continuo de proteger equipos, servicios, redes, aplicaciones de software y datos de posibles ataques informáticos.

CÓDIGO SEGURO: buenas prácticas, técnica o metodologías que se aplican en el ciclo de vida del desarrollo de software, teniendo en cuenta a las personas, procesos y la arquitectura utilizada para endurecer la seguridad informática de una aplicación, reduciendo sus vulnerabilidades.

DESARROLLO DE SOFTWARE: **"**el desarrollo de software se refiere a un conjunto de actividades informáticas dedicadas al proceso de creación, diseño, despliegue y compatibilidad de software**"** [1].

METODOLOGÍA ÁGIL: **"**una forma de pensar en la colaboración y los flujos de trabajo, y define un conjunto de valores que guían nuestras decisiones con respecto a lo que hacemos y a la manera en que lo hacemos**"**[2], con un enfoque iterativo en la gestión de un proyecto.

OWASP: **"**el Open Web Application Security Project ®, es una fundación sin ánimo de lucro que trabaja para mejorar la seguridad del software; a través de proyectos de software de código abierto liderados por su comunidad"[3].

PENTEST: pruebas de penetración realizadas a los sistemas informáticos de una organización, realizadas por un Hacker ético que aplica una serie de técnicas y herramientas para identificar y explotar las vulnerabilidades que puedan tener estos sistemas.

SCRUM: es "una forma de hacer el trabajo en equipo en pequeñas partes a la vez, con experimentación continua y ciclos de retroalimentación en el camino para aprender y mejorar a medida que avanza. SCRUM ayuda a las personas y los equipos a generar valor de forma incremental de forma colaborativa"[4].

---

[1] IBM. ¿Qué es el desarrollo de software?. [En línea]. Bogotá. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://www.ibm.com/co-es/topics/software-development

[2] RedHat. ¿Qué es la metodología ágil?. [En línea]. Bogotá. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://www.redhat.com/es/devops/what-is-agile-methodology

[3] OWASP. Who is the OWASP® Foundation?. [En línea]. Maryland. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://owasp.org/

[4] Scrum.Org. What is Scrum?. [En línea]. Andalucía. La entidad. [consultado el 22 de diciembre de 2022]. Disponible en: https://www.scrum.org/resources/what-is-scrum

VULNERABILIDAD: "una vulnerabilidad es un fallo técnico o deficiencia de un programa que puede permitir que un usuario no legítimo acceda a la información o lleve a cabo operaciones no permitidas de manera remota**"**[5].

[5] INCIBE. Vulnerabilidad. [En línea]. Madrid. La entidad. [consultado el 22 de diciembre de 2022]. Disponible en: https://www.incibe.es/aprendeciberseguridad/vulnerabilidad

RESUMEN

El aumento vertiginoso del desarrollo de software ha exigido a esta industria integrar al ciclo de vida de desarrollo de software metodologías ágiles como SCRUM, generando entregas iterativas y funcionales en cortos periodos de tiempo, reduciendo costos y permitiéndoles soportar el alto volumen de trabajo.

Este enfoque de entregas rápidas y funcionales ha generado que se deje de lado la seguridad informática y las buenas prácticas de desarrollo seguro, considerando que aplicarlas no permite cumplir con los tiempos de entrega de los desarrollos. La seguridad informática debe considerarse uno de los componentes más importantes en la calidad del software y de esta misma manera debe incluirse en todo el ciclo de vida de desarrollo de software.

Se formula la Integración al framework SCRUM de la metodología suministrada por OWASP con 3 de sus proyectos principales, el modelo **SAMM (**SOFTWARE ASSURANCE MATURITY MODEL v2.0.3**)**, el estándar **ASVS** (APPLICATION SECURITY VERIFICATION STANDARD v4.0.3)y la guía **WSTG** (Web Security Testing Guide v4.2), analizando dichas técnicas para generar un procedimiento que facilite su aplicación en el ciclo de vida de desarrollo de software ágil y seguro.

ABSTRACT

The rapid increase in software development has required this industry to integrate agile methodologies such as SCRUM into the software development life cycle, generating iterative and functional deliveries in short periods of time, reducing costs and allowing them to support the high volume of work.

This approach of rapid and functional deliveries has caused computer security and good secure development practices to be left aside, considering that applying them does not allow development delivery times to be met. Computer security should be considered one of the most important components in software quality and in this same way should be included in the entire software development life cycle.

The Integration to the SCRUM framework of the methodology provided by OWASP with 3 of its main projects is formulated, the SAMM model (SOFTWARE ASSURANCE MATURITY MODEL v2.0.3), the ASVS standard (APPLICATION SECURITY VERIFICATION STANDARD v4.0.3) and the WSTG guide (Web Security Testing Guide v4.2), analyzing these techniques to generate a procedure that facilitates their application in the agile and secure software development life cycle.

1. DEFINICIÓN DEL PROBLEMA.

1.1 FORMULACIÓN DEL PROBLEMA

El crecimiento acelerado en el registro, procesamiento e intercambio de grandes volúmenes de información, así como la evolución continua y masificación de las tecnologías, hacen que para suplir o soportar estas necesidades se deba desarrollar software; generando un crecimiento exponencial en este sector. Así mismo las casas de desarrollo de software enfrentan los retos que supone un gran volumen de trabajo, entrega de software de calidad, la entrega funcional en cortos periodos de tiempo, la flexibilidad los cambios frecuentes por parte de los clientes, la carencia de prácticas de código seguro, con lo cual se ve afecta la calidad del software, generando brechas de seguridad; aunque la aplicación sea funcional para el cliente, en consecuencia se ve afectado la reputación de la casa de desarrollo, por su puesto sus clientes e interesados.

Por otra parte, las organizaciones no incorporan la seguridad desde el inicio del SDLC, dejando estas actividades para el final del ciclo; de acuerdo con Vanegas[6], integrar la seguridad en todas las fases del SDLC debería ser tan importante como garantizar que el software sea de calidad; en la actualidad el problema es que las empresas solo se preocupan por la seguridad cuando sus aplicaciones ya se han implementado. Si el objetivo es tener un software funcional y de calidad, de igual manera se debe unificar estos objetivos con la seguridad y sus bases, confidencialidad, integridad, disponibilidad, trazabilidad y autenticidad, según López “si estos objetivos no son acatados correctamente por los ingenieros de software, sus sistemas no podrán continuar su desarrollo del ciclo de vida, ya que la seguridad introduce no solo características de calidad, sino también restricciones bajo las cuales el sistema debe operar”[7].

Se han realizado estudios donde se demuestra que aplicar técnicas de seguridad desde las primeras etapas del SDLC minimiza el costo de corregir las vulnerabilidades.

---

6 VANEGAS, BARRERO, Alejandro. SEGURIDAD PARA MINIMIZAR RIESGOS EN EL DESARROLLO DEL SOFTWARE. Universidad ECOTEC. Bogotá D.C.:[En línea]. [consultado el 20 de enero de 2023]. Disponible en: http://polux.unipiloto.edu.co:8080/00003022.pdf

7LÓPEZ ALVAREZ, Diana Maria. Método para el desarrollo de software seguro basado en la ingeniería de software y ciberseguridad. Universidad Piloto de Colombia. Bogotá D.C.:[En línea]. [consultado el 20 de enero de 2023]. Disponible en: http://polux.unipiloto.edu.co:8080/00003022.pdf

El National Institute of Standards and Technology (NIST) en su reporte The Economic Impacts of Inadequate Infrastructure for Software Testing Final Report, muestra la diferencia en el costo de corregir errores en cada fase de desarrollo:

Figura 1. Costo de reparar errores según etapa de desarrollo

The Economic Impacts of Inadequate Infrastructure for Software Testing

Table 5-1. Relative Cost to Repair Defects When Found at Different Stages of Software Development (Example Only)

X is a normalized unit of cost and can be expressed terms of person-hours, dollars, etc.

Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	5X	10X	15X	30X

Fuente: TASSEY, Gregory. Costo de reparar errores según etapa de desarrollo. [imagen]. Mayo, 2002. [Consultado el 10, enero, 2023]. Disponible en Internet: <https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf >.

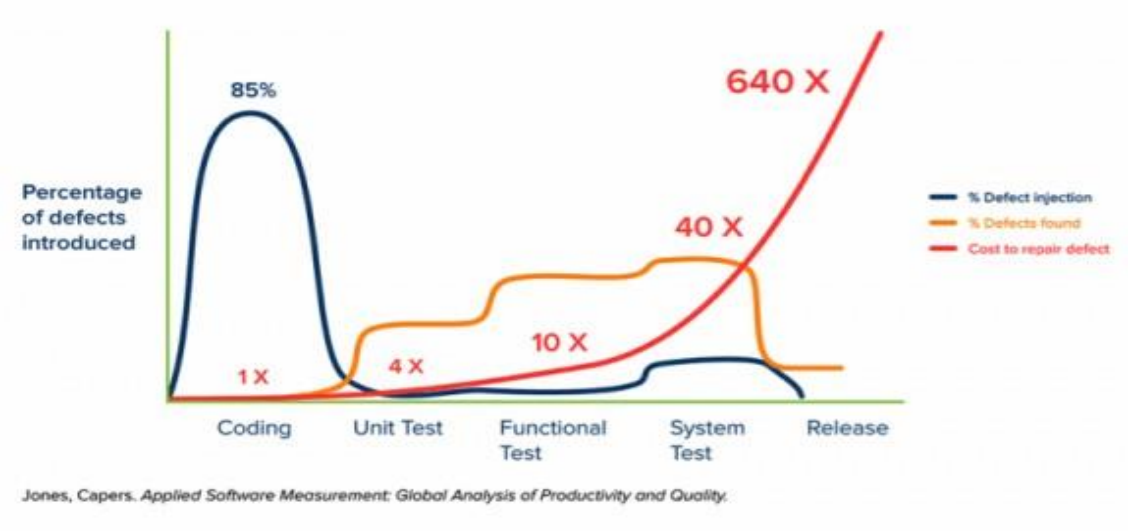
Así mismo, el NIST[8] explica que el costo de corregir un error en la etapa de análisis es de 1X, pero si este mismo es encontrado en la etapa de codificación seria de 5X, donde X puede expresarse en términos de horas-persona, dólares, etc.

Otra métrica que debemos tener en cuenta son las aplicadas a los errores de seguridad, según Capers, citado por SSH team, "El tiempo medio en solucionar un fallo de seguridad aumenta significativamente en relación con la fase del desarrollo en la que nos encontremos. Cuando antes abordemos los errores, menor será el tiempo en solucionarlos"[9].

8TASSEY, Gregory. The economic impacts of inadequate infrastructure for software testing final report [en línea]. Gaithersburg: RTI, 2002 [consultado el 10, enero, 2023]. 309 p. 97. Disponible en Internet: <https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf >

9SEGURIDAD EN el ciclo de vida del software - SSHTeam [Anónimo]. SSHTeam - Expertos en ciberseguridad [página web]. (2020). [Consultado el 30, enero, 2023]. Disponible en Internet: <https://sshteam.com/ssdlc-y-desarrollo-seguro/>.

Figura 2. Tiempo solución errores según etapa



Fuente: SEGURIDAD EN el ciclo de vida del software - SSHTeam [Anónimo] [imagen]. 2020. [Consultado el 10, enero, 2023]. Disponible en Internet: <https://sshteam.com/ssdlc-y-desarrollo-seguro />.

Como podemos observar en los diferentes estudios es mucho menor el costo de corregir un error en la fase donde se origina que en la siguiente fase y se multiplica si se debe corregir al final del ciclo, para Celis[10] es importante identificar en qué fase se producen la mayoría de errores, por lo general es en la fase de pruebas o de implementación y se asume que estos ocurrieron en desarrollo y que por la urgencia, corrigen y cierran sin identificar el origen, ni tomar medidas correctivas para mitigar costo y tiempos de corrección de errores de seguridad.

Con la implementación de metodologías ágiles cómo SCRUM el panorama del desarrollo de software ha venido mejorando en cuanto a los tiempos de entrega y cumplimiento de necesidad funcionales, pero no ha pasado lo mismo con la seguridad del software, PIÑA, Jéssica, et al[11], en su artículo *Análisis prospectivo de la industria de desarrollo de software en Colombia*, realizaron un análisis de algunas variables que afectan el macroentorno de la industria del software en Colombia, dónde identifican a las vulnerabilidades de software como un factor de alto impacto que amenaza la industria, por lo tanto el componente de ciberseguridad debe ser un plus diferencial al promover políticas de seguridad además de gestión, monitoreo y control, considerando esquemas de procedimiento, metodologías y políticas nacionales. Sus resultados muestran a esta variable como la de mayor impacto:

1. Vulnerabilidades del Software.
2. Ley 1341 de 2009.

10SEGURIDAD EN el ciclo de vida del software - SSHTeam [Anónimo]. SSHTeam - Expertos en ciberseguridad [página web]. (2020). [Consultado el 30, enero, 2023]. Disponible en Internet: <https://sshteam.com/ssdlc-y-desarrollo-seguro/>.

11PIÑA TABORDA, Jessica, *et al.* Análisis prospectivo de la industria de desarrollo de software en Colombia. En: Punto De Vista [en línea]. 2020. vol.10, no.2(16) [consultado el 26, enero, 2023], p. 23. Disponible en Internet: <https://doi.org/10.15765/pdv.v11i16.1415 >.

3. Parques científicos, tecnológicos y de innovación.
4. Vínculo Universidad - Empresa - Estado.

Pregunta del problema:
¿Como implementar las técnicas de desarrollo de software seguro al framework de SCRUM?

2 JUSTIFICACIÓN

Para este documento se mantiene confidencial la razón social e información de la casa de desarrollo en la que se implementó el proyecto por solicitud de esta. Desde este momento se denominará SAC.

Actualmente, las organizaciones están siendo atacadas por ciberdelincuentes, utilizan las vulnerabilidades asociadas a la aplicación web generando daños en la información, servicios informáticos sin acceso y poca confianza en los usuarios de estas organizaciones si tenemos en cuenta que sectores atacados como la salud y gobierno y que son fundamentales para el desarrollo social. El pasado mes de octubre el INVIMA fue víctima nuevamente de un ciberataque, según el Tiempo, "Esto produjo la no disponibilidad de información y de los aplicativos externos del Instituto, con excepción de la Ventanilla Única de Comercio Exterior (VUCE)"[12], la entidad tuvo que deshabilitar el portal web y la conexión a sus servidores.

Aplicar de manera ágil las técnicas de desarrollo seguro debe tomarse como una de las acciones a realizar para reducir los riesgos de los clientes, generando confianza y posicionamiento por su calidad en el mercado, viéndose reflejado en la confianza de los usuarios para acceder y gestionar sus actividades diarias en la web.

En Colombia la industria de software ha tenido un gran crecimiento, Fedesoft afirma "este sector no solo es clave para el desarrollo de la economía colombiana en áreas como la ciencia y la tecnología, sino que también es crucial en la generación de empleos y el impulso a la competitividad alrededor de habilidades computacionales. La industria viene creciendo de manera sostenida, con ventas por más de 33,9 billones de pesos en el último año. Así mismo, de acuerdo con el Banco de la República, está generando un 21% de crecimiento en el sector"[13]. Esto implica que cualquier impacto económico para las casas de software será directamente reflejado en las organizaciones, sus clientes.

Actualmente, para dar una correcta respuesta a estos retos se han adoptado metodologías ágiles de desarrollo como el SCRUM, marco de trabajo donde se potencializa el trabajo en equipo, para llegar a una ejecución y desarrollo de manera iterativa, rápida y de valor agregado cumpliendo con objetivos en corto plazo. Sin embargo, se sigue dejando de lado la seguridad en el desarrollo de software. Juan Manuel Haran, editor de welivesecurity describe el informe del FBI y la CISA sobre el grupo ramsomware Hive, desde "junio de 2021, hasta ahora, el grupo atacó a más de 1300 compañías en distintas partes del mundo y recaudó en todo este tiempo más de 100 millones de dólares a partir del pago de las víctimas para recuperar sus archivos y evitar la publicación de la información robada"[14].

---

[12] El tiempo. Invima, en alerta por ataque cibernético. [sitio web]. Bogotá: la entidad. [04, diciembre, 2022]. Disponible en: https://www.eltiempo.com/salud/invima-esta-en-alerta-por-ataque-cibernetico-707158

[13] FEDESOFT. Llega la décima versión de los premios más importantes del mundo del software y las TI en Colombia. [Sitio WEB]. Bogotá. La entidad. [04, diciembre, 2022]. Disponible en: https://fedesoft.org/llega-la-decima-version-de-los-premios-mas-importantes-del-mundo- del-software-las-ti-en-colombia/

[14] WHAT IS Scrum? [Anónimo]. Scrum.org [página web]. [Consultado el 30, noviembre, 2022]. Disponible en Internet: <https://www.scrum.org/resources/what-is-scrum >.

El ESET Security Report (ESR), genera datos importantes en materia de ciberseguridad, "El país con la mayor cantidad de detecciones es Perú (18%), seguido inmediatamente por México (17%), Colombia (12%), Argentina (11%) y Ecuador (9%)". Por otro lado, comentan "hemos cubierto campañas de espionaje dirigidas a entidades gubernamentales de la región. Una de las más recientes fue la campaña llamada Operación Discordia, descubierta en mayo del 2022, que utilizaba códigos espía y de control remoto en equipos de compañías pequeñas y medianas, así como de entidades gubernamentales en Colombia"[15].

Los cambios frecuentes, manejo de errores, comunicación continua y retroalimentación constante, bondades del framework SCRUM también se deben poder gestionar con las metodologías de desarrollo seguro.

Según la organización Scrum, el framework SCRUM es,

> "una forma de hacer el trabajo en equipo en pequeñas partes a la vez, con experimentación continua y ciclos de retroalimentación en el camino para aprender y mejorar a medida que avanza. SCRUM ayuda a las personas y los equipos a generar valor de forma incremental de forma colaborativa. Como marco ágil, Scrum proporciona la estructura suficiente para que las personas y los equipos se integren en su forma de trabajar, al tiempo que agrega las prácticas adecuadas para optimizar sus necesidades específicas" [16].

Uno de sus principales proyectos es la guía de pruebas de seguridad web de OWASP (Web Security Testing Guide) (WSTG) la cual genera su principal recurso de pruebas de ciberseguridad. Según OWAS, "WSTG proporciona un marco de mejores prácticas utilizado por evaluadores de penetración y organizaciones de todo el mundo" [17]. Esta guía permite confrontar la relación entre el costo del software inseguro y el impacto para el negocio. WSTG describe un marco de prueba con técnicas y tareas integradas a cada fase del ciclo de vida del desarrollo de software, SDLC por sus siglas en inglés.

Según la junta de Andalucía, en su página web "un programa efectivo de testeo de aplicaciones web, debe incluir como elementos a testear: Personas, Procesos y Tecnologías, OTP delinea en su primer parte conceptos claves, a la vez que introduce un framework específicamente diseñado para evaluar la seguridad de aplicaciones web a lo largo de su vida" [18]

---

[15] ESET. ESET-security-report-LATAM2022. [Sitio WEB]. Bratislava. La entidad. [30, Noviembre, 2022]. Disponible en: https://www.welivesecurity.com/wp-content/uploads/2022/10/ESET-security-report-LATAM2022.pdf

[16] Welivesecurity. Ransomware Hive atacó a más de 1300 compañías en el mundo. [Sitio WEB]. Bratislava. La entidad. [30, noviembre, 2022]. Disponible en: https://www.welivesecurity.com/la-es/2022/11/22/ransomware-hive-ataco-1300-companias-mundo/

[17] OWASP. OWASP Web Security Testing Guide. [Sitio WEB]. Edgewater Place. La entidad. [30, noviembre, 2022]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

[18] JUNTA DE ANDALUCÍA. Metodología y Frameworks de testeo de la seguridad de las aplicaciones. [Sitio WEB]. Andalucía. La entidad. [01, diciembre, 2022]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/sites/default/files/historico/1.3.1/contenido-recurso-216.html

Con estas premisas, se busca crear productos de calidad que no se conviertan fácilmente en un vector de ataque y que cumpla con las normativas, metodologías, técnicas o procedimientos de desarrollo seguro como OWASP, los equipos deben integrar la seguridad en cada una de las etapas del ciclo de vida del software.

La casa de desarrollo de software ha tenido un crecimiento exponencial del desarrollo de aplicaciones a la medida en el sector de la salud y está asumiendo el impacto generado por centrarse en el producto funcional y no aplicar la seguridad en el ciclo de vida del software, conscientes de esto y de la importancia que actualmente viene tomando, debe cumplir con el compromiso ético de incluir en su marco de trabajo ágil la seguridad en todo el proceso, mejorando la calidad del software.

Mediante este trabajo se pretende realizar una contribución al desarrollo de software seguro integrando las guías de OWASP al framework SCRUM por lo tanto se tendrá un procedimiento que se podrá integrar de manera rápida y contribuir al cambio de cultura hacia el desarrollo de software ágil y seguro.

3 OBJETIVOS

1. OBJETIVOS GENERALES

Proponer un modelo de integración de técnicas de desarrollo de software seguro, mediante el análisis de la guía OWASP aplicado al framework SCRUM para la empresa de desarrollo de software SAC.

2. OBJETIVOS ESPECÍFICOS

- Establecer el estado actual de la seguridad en las fases del ciclo de vida del desarrollo de software (SDLC) con SCRUM en la empresa SAC.
- Analizar los riesgos, amenazas o vulnerabilidades a las que están expuestas las aplicaciones web desarrolladas por la empresa SAC al no aplicar seguridad en su SDLC.
- Integrar las metodologías, técnicas y procedimientos de desarrollo de software seguro con base en la guía OWASP que pueden aplicarse con SCRUM para obtener software de confianza frente a ataques maliciosos.
- Proponer un modelo S-SDLC que integre las actividades de seguridad, principios y prácticas de la guía OWASP y el framework de SCRUM para obtener aplicaciones robustas y confiables.

4 MARCO REFERENCIAL

4.1 MARCO CONTEXTUAL

Este proyecto se aplicará en la empresa de desarrollo de software SAC, que tiene como sede la ciudad de Cali, cuyo objetivo es el desarrollo y comercialización de la aplicación SIS, sobre la cual tiene con una concesión, esta es una aplicación enfocada para el sector salud, farmacias, consultorios médicos, consultorios de oftalmología, Odontología, salud ocupacional; clínicas y hospitales de nivel 3 y 4, con el que se puede gestionar sus procesos administrativos y asistenciales, así como la mayoría de los subprocesos que dan soporte a estos. Así mismo presta servicios de integraciones con facturación electrónica y desarrollo web.

La empresa nace en el año 2019 como concesionario exclusivo para dar soporte a todos los niveles de servicio, implementación y nuevos desarrollos de la aplicación.

En cuanto a su Visión, es convertirse en el proveedor líder de soluciones de tecnología de software del sector salud.

Su propuesta de valor es el conocimiento como rasgo diferencial aplicado a un HIS con la mejor tecnología de software del mercado, 100% web, un producto robusto, maduro y probado en muchas instituciones médicas.

La importancia de este proyecto es contribuir a la visión y propuesta de valor de la organización SAC. Tendría un impacto significativo en toda la organización aplicar un modelo de desarrollo de software seguro, tanto para sus colaboradores con conocimiento avanzado en seguridad de aplicaciones web, como para la seguridad/calidad del producto que ofrece a sus clientes. Todo esto tendría gran impacto social aportando la reducción del riesgo de ataques informáticos con una aplicación segura.

2. MARCO TEÓRICO

1. Seguridad Informática.

En el libro I de la metodología MAGERIT el CN-CERT define la seguridad Informática como "la capacidad de las redes o de los sistemas de información para resistir, con un determinado nivel de confianza, los accidentes o acciones ilícitas o malintencionadas que comprometan la disponibilidad, autenticidad, integridad y confidencialidad de los datos almacenados o transmitidos y de los servicios que dichas redes y sistemas ofrecen o hacen accesibles"[19], en ese sentido el objetivo es proteger las 5 dimensiones, la Disponibilidad, Integridad, Confidencialidad además de la Autenticidad y Trazabilidad en los sistemas informáticos.

2. Amenazas.

Son los posibles eventos que pueden afectar un activo de información, para ROJAS Hernán[20], cualquier situación con la capacidad de afectar considerablemente generando un impacto en la confidencialidad, integridad y disponibilidad de la información, estos puedes ser relacionadas a equipos, desastres naturales, personas e instalaciones.

3. Vulnerabilidad.

Una vulnerabilidad en el software es una brecha, debilidad o falla que puede ser explotada por un ciberdelincuente para afectar una o todas las dimensiones de la seguridad informática, pueden encontrarse en cualquier fase de su construcción. Existen algunos sistemas que estandarizan el nombre o código de las vulnerabilidades y exposiciones de seguridad informática, como es el caso de la corporación MITRE "La misión del programa CVE® es identificar, definir y catalogar las vulnerabilidades de seguridad cibernética divulgadas públicamente. Hay un Registro CVE para cada vulnerabilidad en el catálogo. Las vulnerabilidades son descubiertas, luego asignadas y publicadas por organizaciones de todo el mundo que se han asociado con el Programa CVE"[21].

4. Riesgo.

Es la capacidad de impacto relacionado a una probable materialización de una amenaza sobre un activo, el riesgo es directamente proporcional al impacto de la amenaza y su probabilidad de presentarse, Rojas la describe como "toda posibilidad de ocurrencia de potencia de daño a las personas, equipos, medio ambiente o imagen de la compañía. El

---

19 MINISTERIO DE HACIENDA Y ADMINISTRACIONES PÚBLICAS. Magerit versión 3.0: metodología de análisis y gestión de riesgos de los sistemas de información. [en línea]. Magerit Libro I: Método. Madrid: La entidad. 127 p. [Consultado el 2, enero, 2023]. Disponible en Internet: <https://www.ccn-cert.cni.es/documentos-publicos/1789-magerit-libro-i-metodo/file.html >.

20 ROJAS PEÑA, Hernán Mauricio. Aplicación de la metodología Magerit para el análisis de riesgos de los sistemas de control en la estación Tenay del Oleoducto Alto Magdalena [en línea]. Trabajo de grado. Neiva: Universidad Nacional Abierta y a Distancia UNAD, 2019 [consultado el 11, enero, 2023]. 97 p. Disponible en Internet: <http://repositorio.espe.edu.ec/xmlui/handle/21000/20405>.

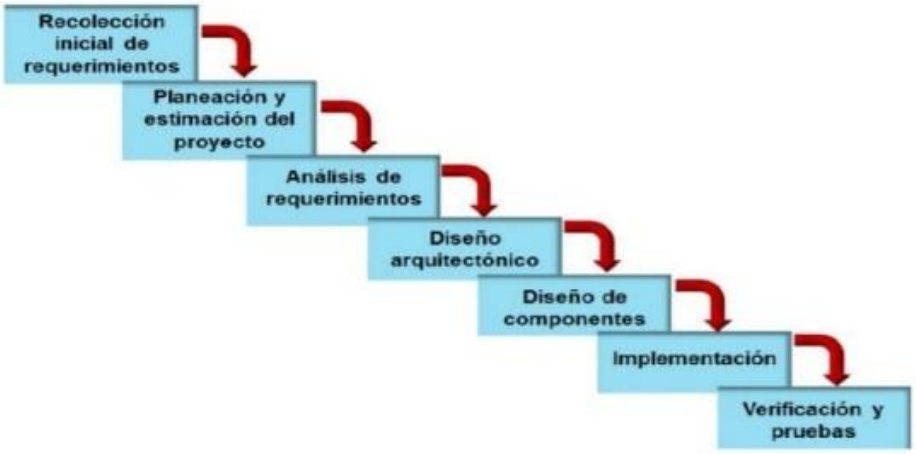
21 CVE®. cve-website. cve-website [página web]. [Consultado el 2, enero, 2023]. Disponible en Internet: <https://www.cve.org/About/Overview >.

riesgo normalmente es directamente proporcional con la vulnerabilidad, así que entre más grande sea esta, mayor es el riesgo que se corre"[22].

4.2.5 Ciclo de vida del software.

En la creación de software de calidad se utiliza una serie de procesos complejos desde su definición hasta la puesta en producción con el fin de satisfacer unas necesidades o cumplir un propósito, para llegar a este objetivo se debe tener una ruta que defina "el Qué se debe hacer" para tener un resultado exitoso, por esto se han definido un conjunto de fases bien delimitadas, a esto se le conoce como ciclo de vida del desarrollo de software SDLC por sus siglas en inglés: la Dra. María, del Carme et al[23], define que los métodos indican "el cómo" fabricar "técnicamente" el software; y este comprende las siguientes fases:

Figura 3. Actividades del proceso de desarrollo de software (SDLC)



Fuente: GOMEZ FUENTES, Maria Del Carmen; CERVANTES OJEDA, Jorge y GONZALEZ PEREZ, Pedro Pablo. Fundamentos de ingeniería de software [imagen]. 2019. [Consultado el 22, enero, 2023]. Disponible en Internet: <http://ilitia.cua.uam.mx:8080/jspui/handle/123456789/1000>.

"Un ciclo de vida es el proceso que se sigue para construir, entregar y hacer Funcionar el software, desde la concepción de la idea del sistema hasta su entrega Y el desuso."

"Un modelo de desarrollo de software es una representación abstracta del Proceso de Desarrollo de software, y determina el orden en el que se llevan a Cabo las actividades del proceso de desarrollo de software. El Modelo de Desarrollo es el procedimiento que se sigue durante el proceso de desarrollo de Un sistema de software y a este también se le llama paradigma del proceso. El Modelo indica el orden de las etapas involucradas en el desarrollo

22ROJAS PEÑA, Hernán Mauricio. Aplicación de la metodología Magerit para el análisis de riesgos de los sistemas de control en la estación Tenay del Oleoducto Alto Magdalena [en línea]. Trabajo de grado. Neiva: Universidad Nacional Abierta y a Distancia UNAD, 2019 [consultado el 11, enero, 2023]. 97 p. Disponible en Internet: <http://repositorio.espe.edu.ec/xmlui/handle/21000/20405>.

23GOMEZ FUENTES, Maria Del Carmen; CERVANTES OJEDA, Jorge y GONZALEZ PEREZ, Pedro Pablo. Fundamentos de ingeniería de software [en línea]. Cuajimalpa: México : UAM, Unidad Cuajimalpa, 2019 [consultado el 22, enero, 2023]. 277 p. Disponible en Internet: <http://ilitia.cua.uam.mx:8080/jspui/handle/123456789/1000 > . ISBN 978-607-28-1659-6.

del software y nos proporciona un criterio para comenzar, para continuar a la siguiente etapa y Para finalizar".

Estos modelos o formas de ejecutar o seguir cada fase del SDLC se dividen en 2 tipos, los convencionales y Ágil.

4.2.6 Framework Scrum.

Este framework es uno de los ejes centrales de este proyecto, como metodología ágil para desarrollo de software. Borges, Cardoso; Lamounier y Tavares mencionan que "Scrum es considerada actualmente una de las metodologías más utilizadas en proyectos de software, surgió a mediados de la década de 1990 y fue idealizada por Ken Schwaber y Jeff Sutherland, la cual se basó en métodos Lean (ciclo de proceso de producción de Toyota) y OODA (ciclo de aviación de combate de EE. UU.)", en su artículo también lo definen "como un *marco* para desarrollar, entregar y mantener productos complejos. Esta definición consta de roles, eventos, artefactos y reglas que los mantienen integrados"[24]. Este marco de trabajo se centra en la colaboración del equipo, permite entregas iterativas, de calidad, que agregan valor al producto en cortos periodos de tiempo.

Según Scrum.org, el marco de trabajo SCRUM es, "una forma de hacer el trabajo en equipo en pequeñas partes a la vez, con experimentación continua y ciclos de retroalimentación en el camino para aprender y mejorar a medida que avanza. Scrum ayuda a las personas y los equipos a generar valor de forma incremental de forma colaborativa. Como marco ágil, Scrum proporciona la estructura suficiente para que las personas y los equipos se integren en su forma de trabajar, al tiempo que agrega las prácticas adecuadas para optimizar sus necesidades específicas" [25]

AHMAD, HILL, LU, MACCLUSKEY Y WADE realizaron una investigación la cual buscaba comprender como encajan las metodologías ágiles y el desarrollo de software global (GSD), exponen el análisis de los desafíos del GSD cuando utilizan SCRUM, encuentran que puede ser complejo aplicar SCRUM al GDS, al mismo tiempo ofrecen algunas soluciones como la comunicación síncrona, Gestión de equipo, Espacios de oficina y la Capacitación a desarrolladores. Utilizaron la revisión de literatura que respondiera a las preguntas: Cuál es el papel del SCRUM, factores que causan restricciones en su uso y qué técnicas y métodos son utilizados para resolver esos problemas en GSD. Como resultado, se evidencia que el marco de trabajo SCRUM es el mejor y más rápido método de GSD, aunque hay muchos factores que generan restricciones en su uso que pueden ser superados con las soluciones ya nombradas [26].

[24] BORGES FRANCIA, Mauro; CARDOSO, Alejandro; LAMOUNER, Junior y TAVARES MOTA. Camila . Agile Short Unified Process - ASUP: A hybrid methodology supported by the adaptation of the Scrum framework and the Unified Process mode. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação. [En línea]. 2022, Vol.46. [consultado el 02 de enero de 2023]. ISSN 1646-9895. Disponible en: http://www.scielo.pt/scielo.php?script=sci_arttext&pid=S1646-98952022000200071&lang=es#B17

[25] SCRUM.ORG. What is Scrum?. [Sitio WEB]. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://www.scrum.org/resources/what-is-scrum

[26] AHMAD, Areeba; HILL, Richard; LU, Juan; MACCLUSKEY, Lee y WADE, Steve. Un análisis sobre la metodología Scrum en el desarrollo global de software - GSD. [En linea]. Lugar de publicación: 2020. [consultado el 02 de Enero de 2023]. Disponible en: https://ieeexplore-ieee-org.bibliotecavirtual.unad.edu.co/document/9457918

Este enfoque nos permite reconocer la importancia de aplicar un framework para desarrollo de software como SCRUM, aplicar las recomendaciones para solucionar los hallazgos que afectan negativamente la implementación del marco y evidenciar su gran capacidad de integración al ciclo de vida de desarrollo de software.

7. Vulnerabilidades y malas prácticas en el desarrollo de software.

El MITRE administra la lista de Enumeración de Debilidades Comunes, CWE por sus siglas en inglés, esta lista enumera las 25 vulnerabilidades más críticas y peligrosas del entorno de software:

1. 2022 CWE Top 25 Most Dangerous Software Weaknesses.

Esta lista, Common Weakness Enumeration (CWE™), es un recurso para ayudar a profesionales asociados a la seguridad informática a mitigar el riesgo, tiene como base los datos de la Common Vulnerabilities and Exposures (CVE®) que se encuentran en la base de datos nacional de vulnerabilidades (NVD) del National Institute of Standards and Technology (NIST). A continuación, se muestra la lista de las debilidades en el CWE Top 25 de 2022:

Figura 4 CWE Top 25 2022

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

Fuente: CWE - 2022 CWE top 25 most dangerous software weaknesses [Anónimo] [imagen]. 3, agosto, 2022. [Consultado el 8,marzo, 2023]. Disponible en Internet: <https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html>.

4.2.7.2 Incibe-Cert.

Es el centro de respuesta a incidentes de seguridad, operado por el INCIBE (Instituto nacional de seguridad) de España, proporciona una base de datos con información en español sobre las últimas vulnerabilidades registradas y conocidas, cuenta con más de 75.000 registros también basados en la NVD (*National Vulnerability Database*) y utiliza el estándar de la CVE (*Common Vulnerabilities and Exposures*).

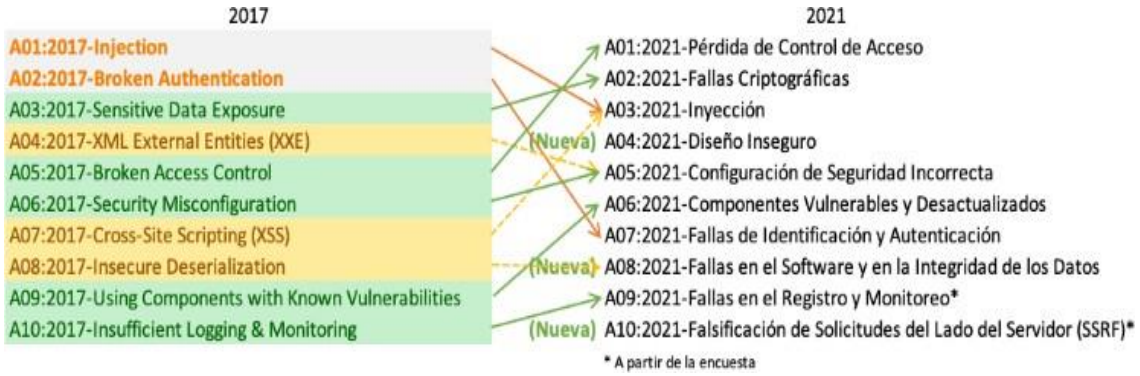
4.2.7.3 OWASP top 10:2021.

La fundación OWASP tiene un proyecto que se puede tomar como estándar base para conocer e identificar dichas vulnerabilidades, el OWASP TOP 10 es un informe que tiene como objetivo principal la concientización, ofreciendo una vista detallada de los riesgos más comunes, los categoriza y define las mejores prácticas de desarrollo de aplicaciones web para mitigar el riesgo en cada categoría.

La fundación OWASP[27], en el último informe del 2021, actualiza sus categorías, incluyendo A10:2021-Falsificación de Solicitudes del Lado del Servidor (SSRF) y A04:2021-Diseño Inseguro, mejorando temas de capacitación.

Enseguida se muestra la lista de OWASP Top 10, los 10 principales riesgos de seguridad de las aplicaciones web en el 2021:

Figura 5. OWASP Top 10: 2021



Fuente: OWASP. Introducción - OWASP Top 10:2021 [imagen]. 2021. [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://owasp.org/Top10/es/A00_2021_Introduction/>.

A continuación, se describen algunas categorías de riesgo con sus ejemplos:

A01:2021 – Pérdida de Control de Acceso. Implementa políticas para que los usuarios no puedan realizar actividades por fuera de su perfil de permisos, generan divulgación de información sin autorización, destrucción de datos. Las CWE incluyen, violación de principio

[27] INTRODUCCIÓN - OWASP Top 10:2021 [Anónimo]. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [página web]. (2022). [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://owasp.org/Top10/es/A00_2021_Introduction/>.

de privilegio mínimo, evitar comprobación de control de acceso por medio de la URL, permitir editar la cuenta de otro usuario conociendo el ID, elevación de privilegios, entre otras.

Ejemplo: Se utilizan datos sin verificación en una sentencia SQL que obtiene la información de una cuenta:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Se modifica el parámetro “acct” en el navegador, si no es validado el atacante entrará a la cuenta de cualquier usuario.

A02:2021 – Fallas Criptográficas. Fallas con la criptografía o ausencia de esta, genera la exposición de información sensible como número de tarjetas de crédito, contraseñas, información personal, información médica. Las CWE incluyen, uso de contraseña en código fuente, algoritmo criptográfico vulnerado o inseguro y entropía insuficiente. Se previene clasificando los datos sensibles según normativa y necesidad del negocio, cifrando los datos sensibles e, implemente protocolos, algoritmos y claves robustos.

Ejemplo:

La aplicación cifra los números de tarjetas de crédito automáticamente, así mismo, se descifran cuando se recuperan, lo que permite que por una inyección SQL se recuperen en texto sin cifrar.

A04:2021 – Diseño Inseguro. Esta nueva categoría describe los riesgos asociados al diseño y las fallas arquitectónicas, aconsejando “mover a la izquierda” de las fases de desarrollo las actividades de seguridad para obtenerla en el Diseño, usando arquitectura de referencia, modelado de amenazas y patrones de diseño seguro. Las CWE incluyen, generar mensajes de error con información confidencial, Violación de fronteras y credenciales con poca protección. Requiere un S-SDLC desde el comienzo y durante todas las fases, apoyado de un profesional de seguridad. Aconseja utilizar SAMM.

Ejemplo:

El proceso de recuperación de contraseña que incluye preguntas y respuestas ya no es una buena práctica según la NIST 80063b, OWASP ASVS y OWASP Top 10, más de una persona puede conocer la respuesta, debe aplicarse un diseño más seguro.

La fundación OWASP promueve el uso del estándar de verificación de seguridad de aplicaciones (ASVS) pues se puede verificar, testear y se aplica a todas las fases del desarrollo de software. Las herramientas por si solas no protegen de manera integral los riesgos definidos en el OWASP Top 10; es por esta razón en la primera fase del modelo propuesto se utilizará como componente de concientización y capacitación a desarrolladores.

4.2.8 Desarrollo de software seguro.

Durante el ciclo de vida del software se deben realizar una serie de pruebas que garanticen la calidad del software en cuanto a su seguridad, se busca reducir la probabilidad que los ciberdelincuentes puedan aprovechar las vulnerabilidades del producto. Actualmente, la seguridad del software debe ser un componente de gran importancia en todo el ciclo de vida del software.

En su trabajo LOPEZ, Diana describe un modelo de software seguro basado en el ciclo de vida del software que busca crear conciencia y generar habilidades para evaluar las vulnerabilidades que podría estar expuesto el software desarrollado, toma como base las métricas OWASP y el ciclo de vida de software

Se puede identificar en este análisis que la seguridad del software permitiría generar software seguro durante todo su ciclo y no al final de este, relaciona los conceptos básicos con los que se debería iniciar la implementación de prácticas de desarrollo seguro utilizando OWASP.

Existen algunas metodologías enfocadas en aplicar técnicas y procedimientos de seguridad en la creación de productos de software en cada una de sus fases, denominados ciclo de vida de desarrollo seguro S-SDLC por sus siglas en inglés, conceptos de gran importancia para esta investigación pues nos permite identificar claramente sus fases y como incluirlo en la metodología actual de desarrollo de software con SCRUM.

Nos centraremos en los modelos relacionados a la organización OWASP, realizaremos una revisión general por algunos de sus métodos (proyectos) siendo estos la base de nuestra investigación:

4.2.9 Application security verification standard 4.0.3.

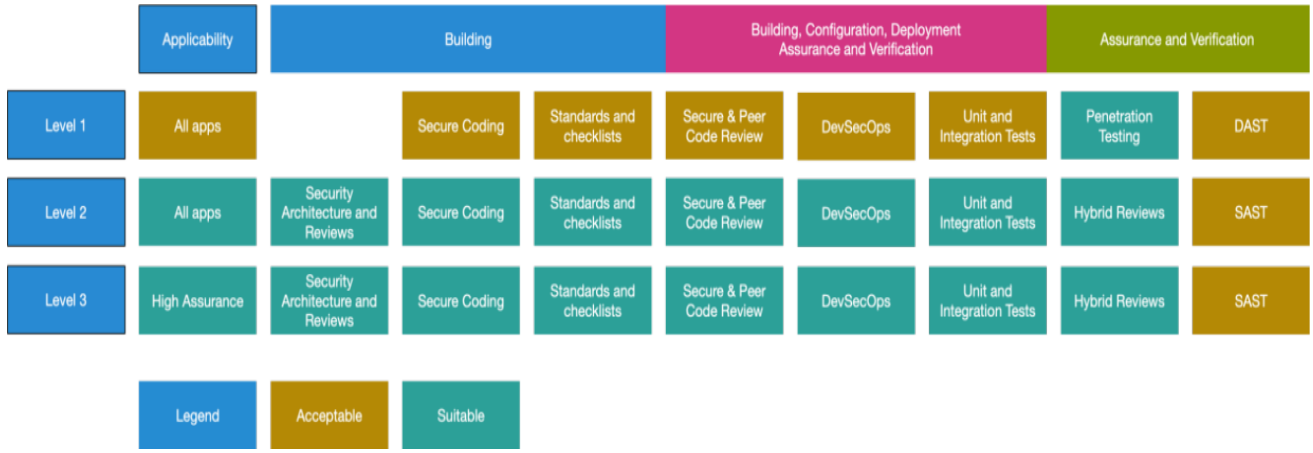
Estándar de verificación de seguridad de aplicaciones, facilita una serie de pruebas para los controles de seguridad de las aplicaciones web, a los desarrolladores, tester, especialistas en Seguridad y consumidores los enfoca en la creación, construcción, pruebas y verificación de aplicaciones seguras. Se alinea con los controles de autenticación avanzados, basados en evidencia de la identidad digital de NIST 800-63-3, así mismo con el OWASP TOP 10 y OWASP proactive Controls. Mapea la Enumeración de debilidades Comunes (CWE por sus siglas en inglés) relacionado en su check list.

Este estándar cuenta con 3 niveles de verificación de la seguridad

- ✓ **ASVS Nivel 1**, para aplicaciones con bajos niveles de garantía.
- ✓ **ASVS Nivel 2**, es el nivel recomendado para aplicaciones que manejan datos confidenciales que se deben proteger.
- ✓ **ASVS Nivel 3**, para aplicaciones que deben tener un alto nivel de confianza, manejan

datos críticos, transacciones de alto valor o datos médicos [28].

Figura 6. Niveles OWASP ASVS 4.0.3



Fuente: OWASP (2021). Application Security Verification Standard 4.0.3. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3en.pdf

4.2.10 Owasp web security testing guide v4.2.

Uno de los proyectos principales es la Guía de pruebas de seguridad web de OWASP (Web Security Testing Guide) (WSTG) la cual genera su principal recurso de pruebas de ciberseguridad. Según OWAS, "WSTG proporciona un marco de mejores prácticas utilizado por evaluadores de penetración y organizaciones de todo el mundo" [29]. Esta guiá permite confrontar la relación entre el costo del software inseguro y el impacto para el negocio. WSTG describe un marco de prueba con técnicas y tareas que serían las adecuadas en cada fase del ciclo de vida del desarrollo de software, SDLC por sus siglas en inglés. Ofrece a las casas de software un marco de prueba completo para que sus aplicaciones web sean seguras y confiables, así como las técnicas para implementarlo en la práctica.

La guía está dividida en capítulos:

- Capítulo 2. Requisitos para probar las aplicaciones web, su alcance, técnicas de pruebas y técnicas para informes.
- Capítulo 3. La presentación del marco de pruebas OWASP, describe las tareas y técnicas relacionadas con el ciclo de vida del software.
- Capítulo 4. Como probar vulnerabilidades mediante inspección del código y pruebas de penetración[30].

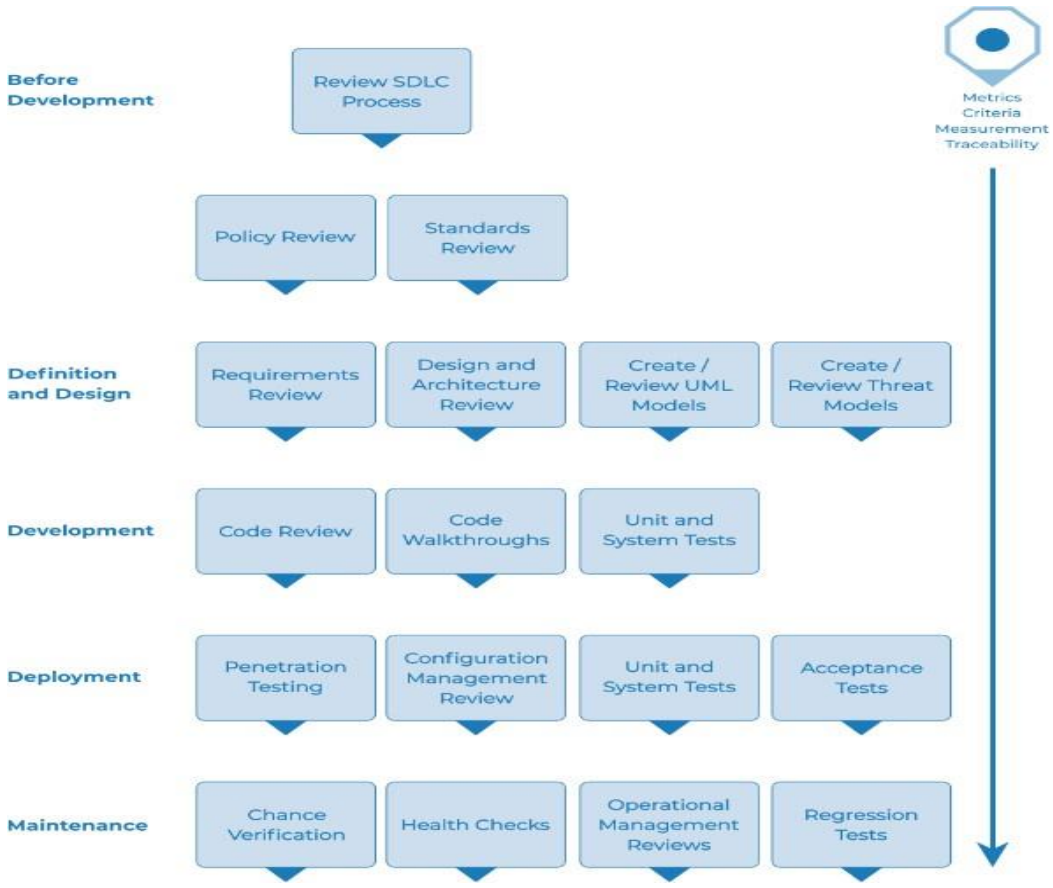
[28] OWASP. Application Security Verification Standard 4.0.3. [Sitio WEB]. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

[29] OWASP. Introduction. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/

[30] OWASP. OWASP Web Security Testing Guide. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

La siguiente figura describe el flujo de trabajo típico de pruebas en un SDLC.

Figura 7. Flujo de trabajo de pruebas en un SDLC



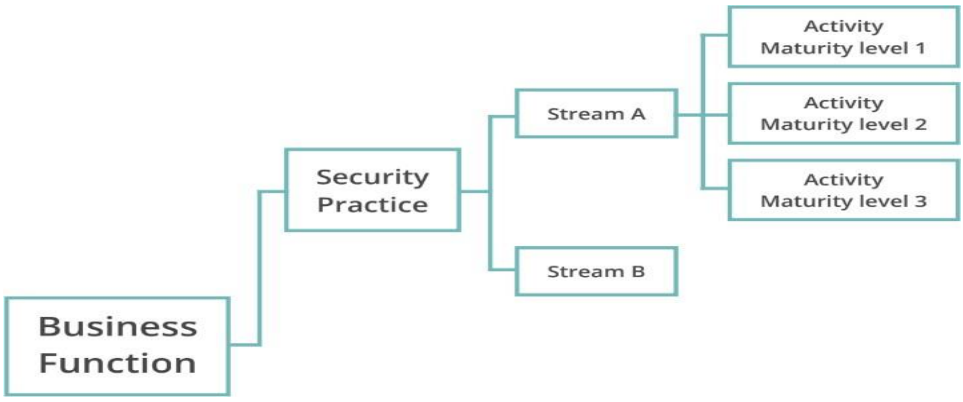
Fuente: OWASP (2022). The Web Security Testing Framework. Disponible en: https://owasp.org/www-project-web-security-testing-guide/v42/3-The_OWASP_Testing_Framework/0-The_Web_Security_Testing_Framework

4.2.11 Software assurance maturity model V2.0.3 (2022).

Es un marco fácil de usar, bien definido y medible, permite que organizaciones pequeñas, medianas y grandes puedan medir en donde se encuentran con sus actuales prácticas de seguridad de software, así como definir un programa de seguridad con iteraciones específicas para mejorar sus prácticas de seguridad, del mismo modo ayuda a definir y medir sus actividades relacionadas, las organizaciones definen su objetivo de madurez. Es flexible y se puede adaptar a cualquier SDLC.

Tiene como base 15 prácticas de seguridad agrupadas en 5 funciones de negocio, cada práctica asocia un conjunto de actividades estructuradas en 3 niveles de seguridad, las de menor nivel de madurez son más fáciles de ejecutar que las de nivel superior.

Figura 8. Estructura funciones del negocio Modelo SAMM



Fuente: OWASP (2022). OWASP SAMM versión 2. Disponible en: https://drive.google.com/file/d/1cI3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view

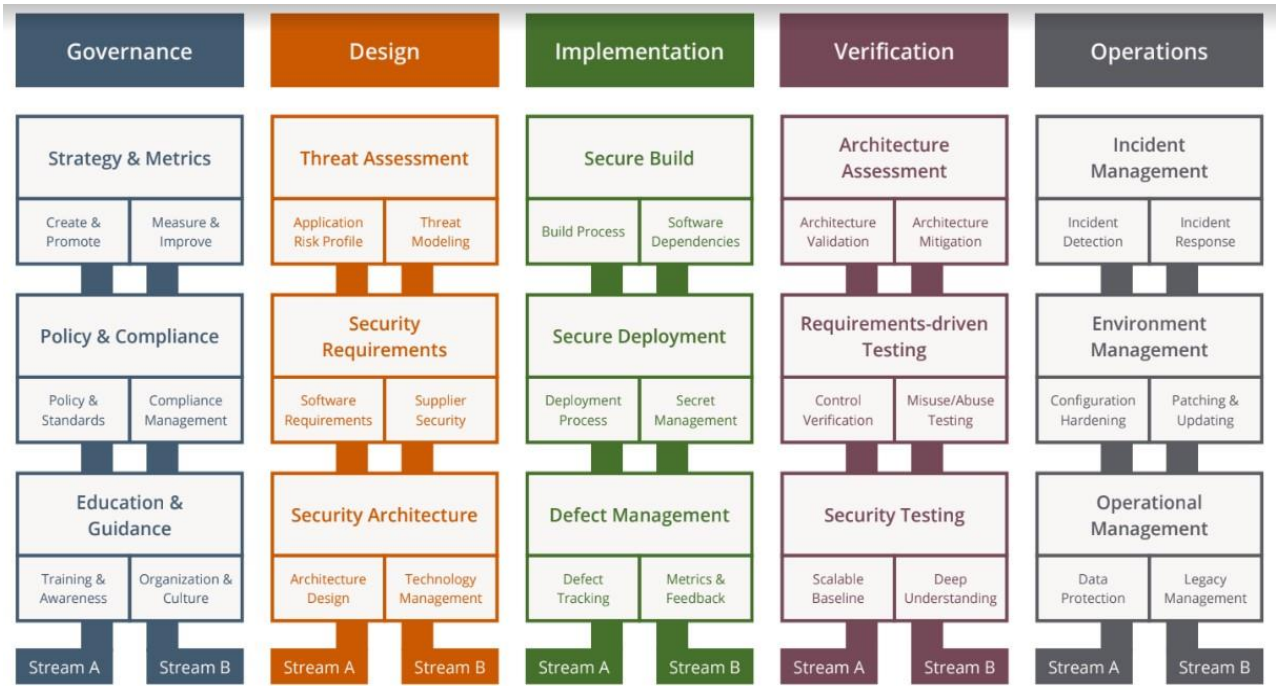
Cada función comercial es una categoría de actividades que las casas de desarrollo deberían aplicar en cierto nivel, cada función contiene 3 prácticas de seguridad agrupadas en dos flujos que abarcan diferentes aspectos de esta, con sus propios objetivos.

Se relaciona la estructura y configuración del Modelo SAMM:

- ✓ Evaluación del estado actual de seguridad del software de la casa de desarrollo.
- ✓ Definición del objetivo de la organización.
- ✓ Definición de hoja de ruta para llegar al objetivo.
- ✓ Consejos prescriptivos sobre como implementar actividades particulares[31].

[31] OWASP. OWASP SAMM versión 2. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 03 de enero de 2023]. Disponible en: https://drive.google.com/file/d/1cI3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view

Figura 9. Estructura y configuración Modelo SAMM



Fuente: OWASP (2022). OWASP SAMM versión 2. Disponible en: https://drive.google.com/file/d/1cI3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view

Uno de los trabajos que nos muestra la versatilidad del framwork SCRUM para integrarse con otros marcos es el de Borges, Cardoso, Lamounier y Tavares (2022) el cual describe una metodología híbrida llamada Ágil Short Unified Process - ASUP, extrae componentes del framework SCRUM y los inserta en las 4 fases del Unified Process (UP), concepción, Elaboración, Construcción y Transición, asociando en el proceso los roles, el sprint, los artefactos y la retrospectiva. Uno de los soportes para implementar ASUP fue la herramienta web de gestión de proyectos Redmine; su objetivo es evaluar su potencial en proyectos del sector TIC en dos organizaciones de educación superior, en las cuales se demostró transformaciones importantes, mejorando sus prácticas de planificación y ejecución de proyectos de software:

- ✓ A través de comunicaciones continuas.
- ✓ En organización de artefactos para su trazabilidad.
- ✓ En la dinámica de los procesos de Planificación, ejecución y seguimiento de actividades.
- ✓ En la adopción de una herramienta que permitió la aplicación de la metodología[32].

Estos dos marcos se destacan por el manejo de entregas iterativas e incrementales que permiten el manejo de cambios y el control de las fases, podemos evidenciar que es viable la integración del framework SCRUM con otras técnicas, métodos y seguir con el pensamiento ágil que caracteriza este marco y tan necesario para el desarrollo de software seguro.

[32]**BORGES FRANCIA, Mauro; CARDOSO, Alejandro;** LAMOUNER, Junior y TAVARES MOTA. Camila . Agile Short Unified Process - ASUP: A hybrid methodology supported by the adaptation of the Scrum framework and the Unified Process mode . RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação. [En línea]. 2022, Vol.46. [consultado el 03 de enero de 2023]. ISSN 1646-9895. Disponible en: http://www.scielo.pt/scielo.php?script=sci_arttext&pid=S1646-98952022000200071&lang=es#B17

4.3 MARCO LEGAL

En este apartado se relacionan las leyes y normas que deben estar alineadas con las buenas prácticas de desarrollo de software seguro.

4.3.1 Ley 1273 de 2009.

La ley 1273 de 2009 es una ley para la protección de los datos informáticos que castiga los delitos contra las bases de la seguridad informática, la confidencialidad, integridad y disponibilidad de los datos y sistemas informáticos, con esta ley se quiere salvaguardar los sistemas informáticos y de comunicaciones con artículos relacionados al acceso u obstaculización de sistemas informáticos, daño o interceptación de datos, también cubre la violación de datos personales y suplantación de sitios web [33].

4.3.2 Ley 1581 de 2012 - Decreto 1377 de 2013.

La ley describe el marco general de protección de datos personales, el decreto reglamenta "aspectos relacionados con la autorización del Titular de información para el Tratamiento de sus datos personales, las políticas de Tratamiento de los responsables y encargados, el ejercicio de los derechos de los Titulares de información, las transferencias de datos personales y la responsabilidad demostrada frente al Tratamiento de datos personales, este último tema referido a la rendición de cuentas" [34].

4.3.3 Ley 1928 del 2018.

Por medio de la cual se aprueba el "Convenio sobre la Ciberdelincuencia", adoptado el 23 de noviembre de 2001, en Budapest.

Esta ley busca que se aplique con prioridad una política penal común que proteja a la sociedad de los ciberdelincuentes, legislando adecuadamente y mejorando la cooperación Internacional [35].

4.3.4 Consejo nacional de política económica y social.

El consejo nacional de Política Económica y social (CONPES) ha creado una serie de políticas generales o documentos CONPES para que el sector público y privado gestione los riesgos de seguridad digital, su objetivo es que se conozca las amenazas a las que las organizaciones están expuestas y comprendan como prevenir y reaccionar a la materialización de dichas amenazas. Uno de los documentos a mencionar es el "**Documento CONPES 3701 del 14 de julio de 2011** Lineamientos de política para ciberseguridad y

33COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Ley 1273. (05, enero, 2009). por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado - denominado "de la protección de la información y de los datos"- y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones. En: Sistema único de Información Normativa. Bogotá D.C.. 2009. 4 p

34COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Decreto 1377. (27, junio, 2013). por el cual se reglamenta parcialmente la Ley 1581 de 2012. En: Sistema único de Información Normativa. Bogotá D.C.. 2013. 28 p.

35COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Ley 1928. (24, julio, 2018). por medio de la cual se aprueba el "Convenio sobre la Ciberdelincuencia", adoptado el 23 de noviembre de 2001, en Budapest.. En: Sistema único de Información Normativa. Bogotá D.C.. 2018. 20 p.

ciberdefensa”8, el objetivo es desarrollar un marco normativo e institucional para contrarrestar el aumento de las amenazas de ciberseguridad. Así mismo, el documento **CONPES 3854 del 11 de abril de 2016,** Su objetivo principal es “fortalecer las capacidades de las múltiples partes interesadas, para identificar, gestionar, tratar y mitigar los riesgos de seguridad digital en sus actividades socioeconómicas en el entorno digital”[36]**.**

[36] DEPARTAMENTO NACIONAL DE PLANEACIÓN. POLÍTICA NACIONAL DE SEGURIDAD DIGITAL. [en línea] Bogotá: La entidad. [consultado el 04 de enero de 2023]. Disponible en: https://colaboracion.dnp.gov.co/CDT/Conpes/Econ%C3%B3micos/3854.pdf

5 DISEÑO METODOLÓGICO

El enfoque de este proyecto es la propuesta de un modelo de desarrollo de software seguro integrando el OWASP y el Framework SCRUM, este modelo se implementará en la casa de desarrollo de software SAC de la ciudad de Cali, la cual desarrolla aplicaciones para el sector salud, se tomará uno de sus proyectos para una de las clínicas de la ciudad.
En primer lugar, es de gran importancia realizar el análisis de la guía de testeo, modelos y técnicas del OWAPS, así como los diferentes estudios de integración de otros modelos de software seguro al Framework SCRUM.

Para realizar el proyecto se adoptó un diseño metodológico de tipo Investigación Acción Participación el cual tiene como objetivo "producir conocimiento y sistematizar las experiencias con el propósito de cambiar una situación social sentida como necesidad, mediante un proceso investigativo donde se involucra tanto el investigador como la comunidad, siendo esta quien orienta el rumbo de la investigación"[37].

En un segundo momento, se evaluará el estado actual de las fases del SDLC integradas actualmente en SAC, para esto se realizarán entrevistas y encuestas estructuradas al jefe de operaciones y a cada uno de los roles del Team SCRUM, estas también nos servirán para conocer la situación actual general de SAC, responsables, tecnologías utilizadas, presupuestos y prácticas en materia de desarrollo de software seguro. También se realizará visitas para realizar observación de cada uno de los procesos actuales del desarrollo de software.

Posteriormente, con base en el análisis de los proyectos OWASP y el estado actual de SAC se definirá la metodología, técnicas y procedimientos de OWASP a integrar al framework SCRUM

Para terminar, se generará el documento con el modelo de integración del OWASP y el framework SCRUM.

[37] *LERMA GONZALES, Hector Daniel.* Metodología de la investigación: propuesta, anteproyecto, y proyecto. *Bogotá D.C.: Ecoe Ediciones, 2009. 72 p. ISBN 978-958-648-602-6.*

6 DESARROLLO DE LOS OBJETIVOS

En este capítulo se desarrollarán las actividades necesarias para estructurar la propuesta de un modelo de desarrollo seguro ágil a partir de los proyectos de OWASP, cuya base tendrá el framework Scrum.

1. ESTABLECER EL ESTADO ACTUAL DE LA SEGURIDAD EN LAS FASES DEL CICLO DE VIDA DEL DESARROLLO DE SOFTWARE (SDLC) CON SCRUM EN LA EMPRESA SAC

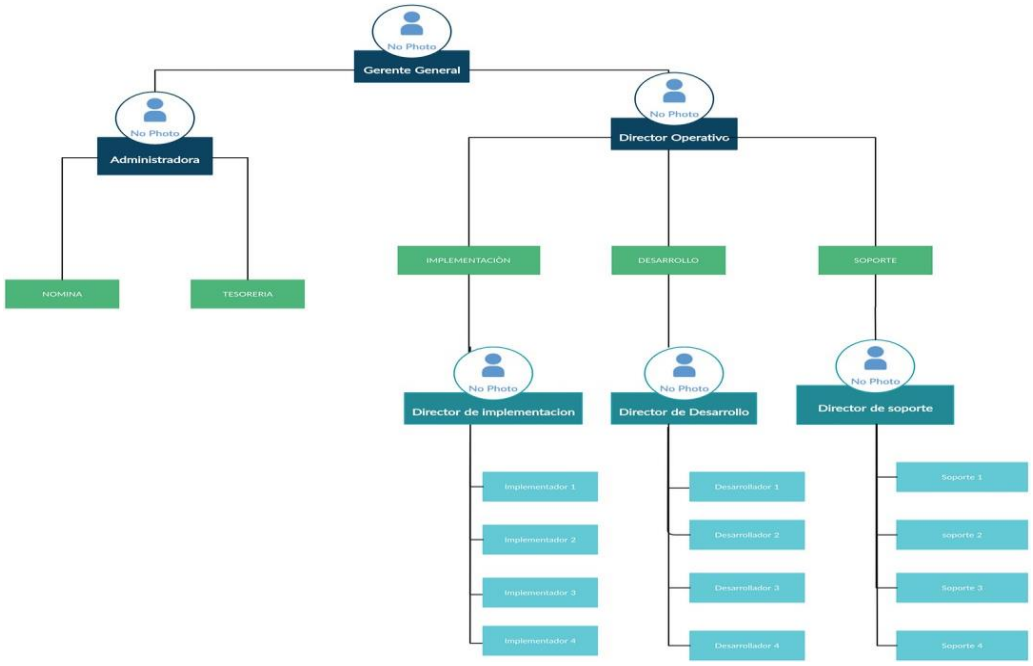
Se debe realizar una revisión general de cada componente de la organización a nivel de estructura, tecnología y su funcionamiento, sobre todo orientado a los procesos y actividades de cada fase de su ciclo de vida de desarrollo de software (SDLC), el uso de Scrum y sus prácticas de seguridad actual.

1. Situación Actual.

La organización cuenta con 14 funcionarios. 4 en implementación, una administradora, 4 en desarrollo, 4 en soporte, un jefe de operaciones y el gerente general.

A continuación, se relaciona el organigrama de la organización SAC:

Figura 10. Organigrama organización SAC



Fuente: elaboración propia.

2. Recolección de información para el diagnóstico.

Con el objetivo de conocer en detalle la cultura de seguridad informática de la organización utilizaremos un cuestionario para conocer, revisar y validar cuáles son sus prácticas de ciberseguridad en la organización SAAC,

Inicialmente, se realizó entrevista con el director general y el director de operaciones con el fin de conocer las políticas de seguridad y una aproximación general de sus prácticas de desarrollo de software y desarrollo seguro.

Se realiza entrevista a los líderes, los procesos de desarrollo, implementación y soporte, para conocer sus responsabilidades y actividades cada una de las fases del SDLC y SCRUM.

Así mismo, mediante observación en cada uno de los procesos se identifica el flujo del proceso de desarrollo actual, herramientas utilizadas y si existe o no buenas prácticas de desarrollo seguro.

Se realizaron 2 encuestas con base en marcos de trabajo que se deben cumplir

Para este caso la encuesta está basada en el estándar internacional ISO/EC 27001 cuyo objetivo es proteger sus 3 ejes principales confidencialidad, integridad y disponibilidad, y del marco de seguridad a utilizar, en este caso OWASP, todo esto con el fin de identificar su alineación a un marco, mejores prácticas y/o controles de seguridad en su producto principal, su HIS para clínicas y hospitales.

Se toma como base el documento de lineamientos de desarrollo seguro de software del ministerio de salud, que contiene lineamientos y recomendaciones para la construcción de software Seguro[38] y que relaciona los controles especificados en la norma ISO/IEC 27002.

La recolección de información del estado actual estará orientada a la ciberseguridad, y teniendo en cuenta el propósito de esta investigación se dejará de lado el componente global de seguridad de la información. Se toma como guía la secuencia de pasos de la implementación de un sistema integrado de gestión asociado a la familia de normas 27000 de cortes Diana[39]
Finalmente, se identifican que fases del SDLC están aplicando con el marco de trabajo SCRUM y que requisitos se deberían tener para integrar prácticas de desarrollo seguro.

3. Resultado del levantamiento de información

De acuerdo a los resultados de las reuniones y entrevistas, en primer lugar, se ratifica el ~~apoyo de la gerencia pa~~ra aplicar el proyecto en la organización, destacando su

[38] *DEPARTAMENTO DE SALUD Y PROTECCIÓN SOCIAL. [en línea] Bogotá: La entidad. [consultado el 26 de enero de 2023]. Disponible en:* https://www.minsalud.gov.co/Ministerio/Institucional/Procesos%20y%20procedimientos/CVSS01.pdf

[39] *CORTES RUGELES*, Diana Marcela; ARDILA GUZMAN, Alix Victoria. Metodología para la implementación de un sistema integrado de gestión con las normas ISO 9001, ISO 20000 e ISO 270001 [en línea]. Tesis especialización. Bogotá: Universidad EAN, 2019 [consultado el 26, enero, 2023]. 72 p. Disponible en Internet: <http://hdl.handle.net/10882/2779 >.

concientización de la importancia de estar bajo marcos de trabajo, normas internacionales y procedimientos de prácticas de desarrollo seguro para los diferentes stakeholders; con este tipo de lineamientos nuestros clientes y proveedores definen si podemos asegurar su información y así hacer negocios con nuestra empresa, así mismo pueden tener una idea de los riesgos que pueden llegar a enfrentar, si podrían cumplir con sus auditorías internas (políticas para contratación, SGSI) y externas; y lo más importante no ser afectados en su reputación y credibilidad. Recordemos que en la aplicación de gestión de salud y administrativa se pueden llegar a guardar y/o procesar datos del área contable, planes comerciales, jurídicos, historias clínicas, datos de identidad entre otros registros confidenciales.

Se relaciona como anexo la autorización.
En según lugar se definió el alcance del diagnóstico inicial, como ya se había dicho se enfocará en el componente de ciberseguridad relacionado con su producto principal el HIS de gestión médica-administrativa, se definió las fases SDLC que se evaluaran: recolección inicial del requerimiento, Diseño arquitectónico, Diseño de componentes, implementación y verificación y pruebas.

Luego se estableció el grupo de trabajo integrado por el director de operaciones como líder, el director de desarrollo, un desarrollador y quien realiza la investigación, siendo los designados por la organización los funcionarios con poder de decisión, con más experiencia y conocimiento de sus procesos, así como la capacitación en el Framework SCRUM. Se acuerdan 2 reuniones semanales para el levantamiento de la información, análisis y definición del ciclo de desarrollo seguro, así como las técnicas que se aplicarían, 3 reuniones semanales para la etapa de desarrollo.

Posteriormente, la asignación de otros recursos como 3 computadores, espacio de reuniones en el área de desarrollo. El proyecto asignado para esta investigación está definido como ENDO, proyecto para una clínica de la ciudad.

6.1.4 Diagnóstico.

Con las entrevistas y observaciones se pudo identificar la situación actual de la casa de desarrollo:

- Ausencia de estrategias y métricas para la seguridad del software.

-No se tiene una clasificación de nivel de los datos, no se puede identificar que datos son privados y cuáles sensibles.

-Al indagar por el cumplimiento de las normas de la familia ISO/EC 27000, no se tenía como políticas internas definidas, influyendo el costo de su implementación y no tienen un presupuesto para seguridad de la información.

-El manejo del framework SCRUM se realizó en un proyecto que ya se dio por terminado, se iniciara en el proyecto definido para esta investigación. Cuentan con 3 funcionarios con la formación para el manejo de este framework.

-Se tiene como controles implementados en las fases de SDLC, recolección del requerimiento, se aprueba, después pasa a desarrollo, Luego pasa a implementación para realizar pruebas y si es aceptado, pasa al cliente y esté válida condiciones de aceptación y finalmente llega a producción.

-Actualmente, la casa de desarrollo no cuenta con un profesional especializado o un rol asignado a ciberseguridad.

-Se identificó que la casa de desarrollo no contempla un modelo de desarrollo seguro en la construcción de su HIS, tampoco implementa alguna metodología de testing a sus productos web.

- No Utilizan Herramientas de análisis de código como SAST o DAST.

6.1.5 Diagnóstico a las prácticas de desarrollo de software.

Planificación. Se realiza con gerencia e implementación, se definen los requisitos con el cliente y ellos definen como hacerlo. Enfocados en cumplir con la funcionalidad del software, cumpliendo con la necesidad del cliente, sin políticas de seguridad ni prácticas de desarrollo seguro.

Análisis. El área de implementación construye el requerimiento con los parámetros de "para", "porque" y condiciones de aceptación, anexa todos los soportes normativos, estándares y demás soportes de la solicitud. No se tiene ninguna técnica asociada a seguridad.

Diseño. Se realiza el bosquejo del diseño para dar una idea al desarrollador, este documento de carga a la plataforma GLPI. No se tiene ninguna técnica asociada a seguridad.

Desarrollo. Con el GLPI ingresan a evaluar el alcance, tiempo de desarrollo, se clasifica por fases y se asignan a cada desarrollador, una por desarrollador. No se cuenta con analizador de código, se utilizan distintas interfaces con otros servicios y librerías que no se conoce si cuentan con prácticas de desarrollo seguro.

Pruebas. En desarrollo se realizan pruebas básicas, se entrega a implementación para realizar pruebas de funcionalidad respecto a si cumple criterios de aceptación definidos en el requerimiento. No se realizan pruebas de seguridad y calidad.

Despliegue. Los funcionarios de implementación realizan el despliegue en el ambiente de producción, no se tiene separado el ambiente de pruebas y producción.

Herramientas.
Visual Studio Code.
Sublime Text,
Compilador de código (JSON, XML) herramienta libre.
Postman para ejecutar interacciones entre aplicaciones.
Javascript.
Bootstrap.
Laravel.
PHP.
HTML.
CSS.
Java.
Postgesql.
Linux.

6.1.6 Componentes de seguridad actuales.

Validación de entradas y salidas: restricciones en INPUT y rutinas de salida.

Gestión de autenticación: Seguridad, validación, inyección SQL, Manejo de md5 en las contraseñas, usuario y contraseña para ingreso, tiempo de caducidad de cuentas, restricciones de multiusuario y manejo de cookies.

Gestión de sesiones: básica.

Gestión Criptografía: No se manejan estándares criptográficos para datos de nivel sensible.

Gestión de Errores: Se manejan mensajes de genéricos, Manejo de excepciones y envió a index o 404.

Gestión de Ambientes: No se tiene segmentada la red para los ambientes de producción y de pruebas, no hay credenciales diferentes para cada ambiente.

Se debe contemplar en cada fase de SDLC políticas de desarrollo seguro integrado en su framework SCRUM, métricas de pruebas y el análisis de reporte de pruebas periódicas de pentesting para tener capacidad de dar respuesta a incidentes de seguridad, ser una organización escalable, reducción costos en recuperación de eventos de seguridad, mejora su reputación, siendo priorizada en procesos de expansión internacionales.

Con base en la información recopilada, el entendimiento del estado actual de la organización, disponibilidad de sus recursos, prácticas de seguridad y uso de Scrum, se puede tener más claro las actividades a realizar para establecer un modelo de ciclo de vida de desarrollo seguro.

2. ANALIZAR LOS RIESGOS, AMENAZAS O VULNERABILIDAD A LAS QUE ESTÁN EXPUESTAS LAS APLICACIONES WEB DESARROLLADAS POR LA EMPRESA SAC AL NO APLICAR SEGURIDAD EN SU SDLC

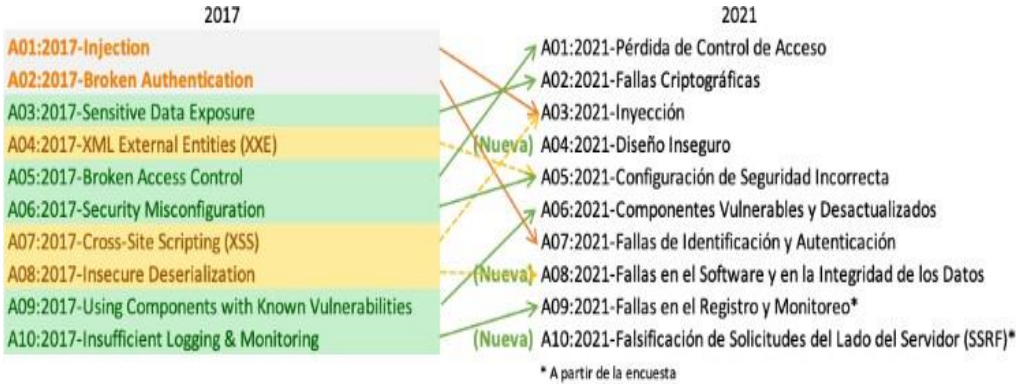
Uno de los principales vectores de ataque utilizados por los ciberdelincuentes son las aplicaciones web, por medio de estas, podrían ingresar a la estructura de red y de datos de una organización, por lo tanto, es necesario conocer las amenazas o vulnerabilidades a las que se expone la organización SAC si no aplica seguridad en su SDLC. Reconocer el riesgo supone la concientización para integrar metodologías, buenas prácticas y herramientas de seguridad informática en cada una de las fases del SDLC.

A continuación, se revisan 2 de los más importantes recursos que nos ofrece OWASP y el MITRE para conocer las vulnerabilidades de seguridad más frecuentes en aplicaciones web.

1. OWASP Top 10:2021.

La fundación OWASP tiene un proyecto que se puede tomar como estándar base para conocer e identificar dichas vulnerabilidades, el OWASP TOP 10 es un informe que tiene como objetivo principal la concientización, ofreciendo una vista detallada de los riesgos más comunes, los categoriza y define las mejores prácticas de desarrollo de aplicaciones web para mitigar el riesgo en cada categoría.
La fundación OWASP[40], en el último informe del 2021 actualiza sus categorías, incluyendo A10:2021-Falsificación de Solicitudes del Lado del Servidor (SSRF) y A04:2021-Diseño Inseguro mejorando temas de capacitación.
Enseguida recordamos la lista de OWASP Top 10, los 10 principales riesgos de seguridad de las aplicaciones web en el 2021:

Figura 11. OWASP Top 10: 2021



Fuente: OWASP. Introducción - OWASP Top 10:2021 [imagen]. 2021. [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://owasp.org/Top10/es/A00_2021_Introduction/>.

[40] INTRODUCCIÓN - OWASP Top 10:2021 [Anónimo]. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [página web]. (2022). [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://owasp.org/Top10/es/A00_2021_Introduction/>.

A continuación, se describen las categorías de riesgo con sus ejemplos:

1. A01:2021 – Pérdida de Control de Acceso.

Implementa políticas para que los usuarios no puedan realizar actividades por fuera de su perfil de permisos, generan divulgación de información sin autorización, destrucción de los datos. Las CWE incluyen, violación de principio de privilegio mínimo, evitar comprobación de control de acceso por medio de la URL, permitir editar la cuenta de otro usuario conociendo el ID, elevación de privilegios, entre otras.

Las CWE con más importancia son:
CWE-200: Exposición de información sensible a un actor no autorizado.
CWE-201: Exposición de información confidencial a través de datos enviados.
CWE-352: Falsificación de Peticiones en Sitos Cruzados (CSRF).

Ejemplo:
Se utilizan datos sin verificación en una sentencia SQL que obtiene la información de una cuenta:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Se modifica el parámetro “acct” en el navegador, si no es validado el atacante entrara a la cuenta de cualquier usuario.

2. A02:2021 – Fallas Criptográficas.

Fallas con la criptografía o ausencia de esta, genera la exposición de información sensible como número de tarjetas de crédito, contraseñas, información personal, información médica.
Las CWE con más importancia son:

CWE-259: Uso de contraseña en código fuente.
CWE-327: Algoritmo criptográfico vulnerado o inseguro.
CWE-331: Entropía insuficiente.

Se previene clasificando los datos sensibles según normativa y necesidad del negocio, cifrando los datos sensibles e implemente protocolos, algoritmos y claves robustos.

Ejemplo:

La aplicación web cifra los números de tarjetas de crédito con la función de cifrado automático de la BD, así mismo, se descifran cuando se recuperan, lo que permite que por una inyección SQL se recuperen estos datos en texto sin cifrar.

3. A03 Inyección.

La aplicación web se considera que puede ser atacada en esta categoría, cuando:

- ✓ No se validan los datos suministrados por el usuario.
- ✓ Los parámetros no están codificados, al invocar consultas dinámicas.
- ✓ Para extraer registros sensibles, se utilizan datos no confiables en los parámetros de búsqueda en consultas ORM.

Las CWE más importantes son:

CWE-79: Secuencia de Comandos en Sitios Cruzados (XSS).

CWE-89: Inyección SQL.

CWE-73: Control Externo de Nombre de archivos o ruta.

Se recomienda pruebas automatizadas en parámetros, encabezados, URL. Cookies, JSON, SOAP, XML e incluir herramientas para SAST, DAST o IAST.

Ejemplo:

La aplicación usa datos inseguros en su SQL:

```
String query = "SELECT \* FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

El ciberdelincuente al modificar el valor del parámetro Id en su navegador:

http://example.com/app/accountView?id='UNION SELECT SLEEP(10);--

Retorna los datos de la tabla accounts. Es claro que se pueden hacer ataques mucho más peligrosos.

6.2.1.4 A04:2021 – Diseño Inseguro.

Relaciona los riesgos asociados al diseño y las fallas arquitectónicas aconsejando “mover a la izquierda” de las fases de desarrollo, las actividades de seguridad para obtenerla en el Diseño, usando arquitectura de referencia, modelado de amenazas y patrones de diseño seguro.

Las CWE más importantes son:

CWE-209: Generación de mensaje de error que contiene información confidencial.

CWE-256: Almacenamiento desprotegido de credenciales.

CWE-501: Violación de las fronteras de confianza.

CWE-522: Credenciales protegidas insuficientemente.

Requiere un S-SDLC desde el comienzo y durante todas las fases, apoyado de un profesional de seguridad. Aconseja utilizar SAMM.

Gestionar requerimientos y recursos.

Agrupe y negocie los requerimientos de la aplicación con la organización, incluyendo los de seguridad (funcionales y no funcionales), según la lógica del negocio.

Definir nivel de exposición de la aplicación, separación de funcionalidades y control de acceso.

Planificar el presupuesto que incluya cada fase del S-SDLC

Ejemplo:

El proceso de recuperación de contraseña que incluye preguntas y respuestas ya no es una buena práctica según la NIST 80063b, OWASP ASVS y OWASP Top 10, más de una persona puede conocer la respuesta, debe aplicarse un diseño más seguro.

6.2.1.5 A05:2021 – Configuración de Seguridad Incorrecta.

Veamos algunas brechas de seguridad asociadas:
Funciones instaladas o habilitadas que no son necesarias.

Cuentas predeterminadas activas.

Manejo de errores muy informativos.

Funciones de seguridad deshabilitadas.

Software desactualizado.

Las CWE más importantes son:

CWE-16 Configuración.

CWE-611 Restricción incorrecta entidades externas referenciadas de XML.

En los procesos de instalación seguros se debe implementar hardening repetible, los entornos de desarrollo, control de calidad y producción deben ser idénticos, segmentación entre componentes o instancias y un proceso automático para validar configuraciones y ajustes en cada entorno.

Ejemplo:

En el servidor de aplicaciones en producción, dejan aplicaciones de prueba que tienen brechas de seguridad, por ejemplo, no se cambian las cuentas por default, el ciberdelincuente puede controlar a través de estas.

6. A06:2021 – Componentes Vulnerables y Desactualizados.

Cuando se desconocen las versiones de los componentes que utiliza.

No se tiene soporte para el software o no está actualizado: SO, servidor web, administrador BD, las API, bibliotecas y entornos.

No realizar análisis de vulnerabilidades frecuente.

Desarrollo no prueba si las bibliotecas actualizadas son compatibles.

Las CWE más importantes son:

CWE-1104: Uso de componentes de terceros no mantenidos.

Las dos CWE del OWASP Top 10 2013 y 2017.

Se debe tener procesos para:

Eliminar dependencias, componentes o funcionalidades que no son utilizadas.

Realizar inventario de las versiones de los componentes con herramientas como versions, OWASP Dependency Check, retire.js, etc.

Revisar bases de datos como CVE y NVD.

Obtener componentes de fuentes oficiales.

Ejemplo:

Shodan IoT es un motor de búsqueda que le ayuda al ciberdelincuente a ubicar dispositivos que tiene la vulnerabilidad Heartbleed.

7. A07:2021 – Fallas de Identificación y Autenticación.

Se tienen debilidades de autenticación cuando:

Permite ataques de fuera bruta.

Admite contraseñas por defecto o débiles.

Proceso no efectivo de olvido de contraseña como “respuesta basada en el conocimiento”

No tiene autenticación multifactor.

Las CWE más importantes son:

CWE-297: Validación incorrecta de Certificado con discrepancia de host.

CWE-287: Autenticación incorrecta.

CWE-384: Fijación de sesiones.

Ejemplo:

La práctica de solicitar el cambio y complejidad en las contraseñas alientan el uso de contraseñas débiles, se recomienda cambiar a las prácticas de la guía NIST 800-63 y utilizar autentificación multifactor.

8. A08:2021 – Fallas en el software y en la integridad de los datos.

Una aplicación es vulnerable cuando:

Depende de plugins, bibliotecas, repositorios o CDN no confiables.
Un pipeline CI/CD inseguro.
Actualización de aplicaciones sin verificar su integridad.
Datos serializados en estructuras que un delincuente puede ver y modificar.

Las CWE más importantes son:
CWE-829: Inclusión de funcionalidades provenientes de fuera de la zona de confianza.
CWE-494: Ausencia de verificación de integridad en el código descargado.
CWE-502: Deserialización de datos no confiables.

Para prevenirlo se describen algunas acciones:

Verificar que el software o datos viene de sitio esperado y no han sido modificados utilizando firmas digitales.
Utilizar herramientas para analizar los componentes de terceros: OWASP Dependency Check u OWASP CycloneDX.
Proceso de revisión para detectar cambios de código y configuraciones maliciosas en su pipeline CI/CD.
Los datos sin cifrar o firmar no serán enviados a clientes no confiables, adicionar verificación de integridad.

9. A09:2021 – Fallas en el Registro y Monitoreo.

Las vulnerabilidades no pueden ser detectadas sin registro y monitoreo:
Fallas en el inicio de sesión, inicio de sesión y transacciones de alto valor no son registradas.
No se generan o no son claros los registros de advertencias o errores.
Los escaneos realizados con herramientas de pruebas dinámicas como OWASP ZAP no generan alertas.

Las CWE más importantes son:
CWE-117 Neutralización de salida incorrecta para registros.
CWE-223 Omisión de información relevante para la seguridad.
CWE-532 Inserción de información sensible en archivo de registro.
Se describe algunos controles a implementar:

Los registros deben tener un formato fácil de procesar por las herramientas de gestión de registros.

Los datos de registro son codificados para prevenir inyección en el monitoreo o registros.
Plan de respuesta y recuperación del NIST 800-61r2.

Ejemplo: Una aerolínea tuvo una pérdida de datos (pasaportes, tarjetas de crédito) de pasajeros por 10 años, la produjo un proveedor de servicios en la nube.

6.2.1.10 A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF).

Esto se presenta cuando la aplicación web accede a un recurso sin validar la URL que envió el usuario, la aplicación podría enviar una solicitud falsa a cualquier destino así este protegido por firewall, VPN o ACL.

Se puede prevenir implementando controles de defensa en profundidad:
Segmentación de acceso a recursos remotos.
Políticas de denegar por defecto.
Sanitizar y validar datos de entrada del cliente.
Deshabilitar las direcciones HTTP.

Ejemplo:

El atacante puede acceder a los archivos locales de servicios internos para llegar a datos confidenciales como file:///etc/passwd y `http://localhost:28017/`

La fundación OWASP promueve el uso del estándar de verificación de seguridad de aplicaciones (ASVS) pues se puede verificar, testear y se aplica a todas las fases del desarrollo de software. Las herramientas por si solas no protegen de manera integral los riesgos definidos en el OWASP Top 10; es por esta razón en la primera fase del modelo propuesto se utilizará como componente de concientización y capacitación a desarrolladores.

6.2.2 2022 CWE Top 25 Most Dangerous Software Weaknesses.

Esta lista, Common Weakness Enumeration (CWE™), es un recurso para ayudar a profesionales asociados a la seguridad informática a mitigar el riesgo, tiene como base los datos de la Common Vulnerabilities and Exposures (CVE®) que se encuentran en la base de datos nacional de vulnerabilidades (NVD) del National Institute of Standards and Technology (NIST).

En la siguiente imagen, se muestra la lista de las debilidades en el CWE Top 25 de 2022:

Figura 12. CWE Top 25 2022

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

Fuente: CWE - 2022 CWE top 25 most dangerous software weaknesses [Anónimo] [imagen]. 3, agosto, 2022. [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html>.

A continuación, se describen algunas de las debilidades relacionadas en este top:

6.2.2.1 CWE-89: Neutralización incorrecta de elementos especiales utilizados en un comando SQL ('Inyección SQL').

En el puesto 3.

Si no se elimina la sintaxis SQL en las entradas controlables por el usuario, se puede utilizar para alterar la lógica de la consulta y evitar las validaciones de seguridad o insertar sentencias adicionales para modificar la base de datos, incluyendo ejecución de comandos del sistema.

La inyección SQL es un problema común en las aplicaciones web con base de datos, este se detecta y explota fácilmente.

Consecuencias:
Pérdida de confidencialidad en los datos.
Pérdida de control de acceso al usar comando SQL deficientes para verificar nombres de usuario y contraseñas.
Afecta la integridad de los datos, pues permitiría cambiarlos o eliminarlos.

2. CWE-20: Validación de entrada incorrecta.

En el puesto 4.
La validación de entradas comprueba entradas potencialmente peligrosas para garantizar que estas son seguras al procesarlas en el código o al comunicarse con otros componentes. Si la aplicación no válida la entrada adecuadamente, llevara que partes del sistema reciban entradas no deseadas, que pueden dar lugar al control arbitrario de un recurso o de código.

Consecuencias.
Disponibilidad: El ciberdelincuente puede proporcionar valores inadecuados y causar bloqueos o consumo excesivo de recursos.
Integridad, confidencialidad y disponibilidad: Podría usar información maliciosa para modificar datos, alterar el flujo de control y ejecución arbitraria de comandos.

3. CWE-434: Carga sin restricciones de archivos de tipo peligroso.

En el puesto 10.
La aplicación permite al atacante cargar o transferir archivos de tipo peligroso que se pueden procesar en el entorno de la aplicación.

Consecuencias.
Integridad, confidencialidad y disponibilidad: Ejecutar comandos no autorizados es posible si se interpreta y ejecuta un archivo que se cargó como código. Por ejemplo, en un entorno unix los programas no pueden ejecutarse sin el bit de ejecución, pero el servidor web puede ejecutar los programas PHP sin invocarlo.

4. CWE-862: Autorización faltante.

En el puesto 16**.**
La aplicación no realiza verificación de autorización cuando un actor intenta acceder a un recurso o realizar una acción y los usuarios podrían acceder a los datos o realizar acciones no autorizadas, que puede llegar a exposición de información, DoS y ejecución de código arbitrario.

Consecuencias.
Confidencialidad, integridad: El atacante podría leer o modificar datos confidenciales, desde un almacén de datos que no esté restringido o con funcionalidad privilegiada.
Control de acceso: El atacante podría obtener privilegios leyendo o modificando datos críticos.

5. CWE-918: Falsificación de solicitud del lado del servidor (SSRF).

En el puesto 21.

Si se da una URL a hosts o puertos no esperado, puede hacer que parezca que el servidor es quien está enviando la solicitud omitiendo controles de acceso como firewall. El servidor se podría utilizar como proxy para escanear los puertos de los hosts de la red interna y acceder a documentos.

Consecuencias.

Confidencialidad: leer datos de la aplicación.

Integridad: Ejecutar código o comandos no autorizados.

6. Incibe-Cert.

Es el centro de respuesta a incidentes de seguridad, operado por el INCIBE (Instituto nacional de seguridad) de España, proporciona una base de datos con información en español sobre las últimas vulnerabilidades registradas y conocidas, cuenta con más de 75.000 registros también basados en la NVD (*National Vulnerability Database*) y utiliza el estándar de la CVE (*Common Vulnerabilities and Exposures*).

Uno de los métodos de detección de algunas de las vulnerabilidades descritas son las pruebas de software, esta etapa del SDLC permite encontrar brechas de seguridad y acercarnos a tener un software seguro.

6.2.3 Análisis estático automatizado: Las pruebas estáticas de seguridad de aplicaciones (SAST).

Según la CWE[41], estas pruebas pueden encontrar brechas de seguridad analizando el código fuente sin tener que ejecutarlo. Se construye un modelo de flujo de datos y flujo de control para buscar patrones potencialmente vulnerables que conecten "fuentes" (orígenes de entrada) con "sumideros" destinos, donde los datos interactúan con componentes externos, con el SO, por ejemplo.

Herramientas para realizar pruebas SAST:

FindBug.

SonarQube. Realiza pruebas tanto SAST como DAST.

CAÑO, Jose[42], describe el plugin Warning Next Generation que integra otros plugins y herramientas:

Maven console, Registra avisos y errores durante la compilación del proyecto.

Checkstyle, Analiza el estilo del código con referencia al estilo de codificación.

CDP, Busca código duplicado.

[41] CWE - 2022 CWE top 25 most dangerous software weaknesses [Anónimo]. 3, agosto, 2022. [Consultado el 3, marzo, 2023]. Disponible en Internet: <https://cwe.mitre.org/data/definitions/918.html>

[42] CAÑO QUINTERO, Jose Joaquin. DevSecOps: integración de herramientas SAST, DAST y de análisis de Dockers en un sistema de integración continua. [en línea]. Trabajo de Grado. Universitat Oberta de Catalunya, Catalunya.: 2019. [Consultado 10, marzo,2023]. Disponible en: https://openaccess.uoc.edu/handle/10609/95987

PMD, Busca variables no utilizadas, creación de objetos innecesarios, entre otros.
SpotBugs, Encontrar bugs en el código.
Task Scanner, busca las tareas abiertas marcadas en el código con TODO o TOFIX.

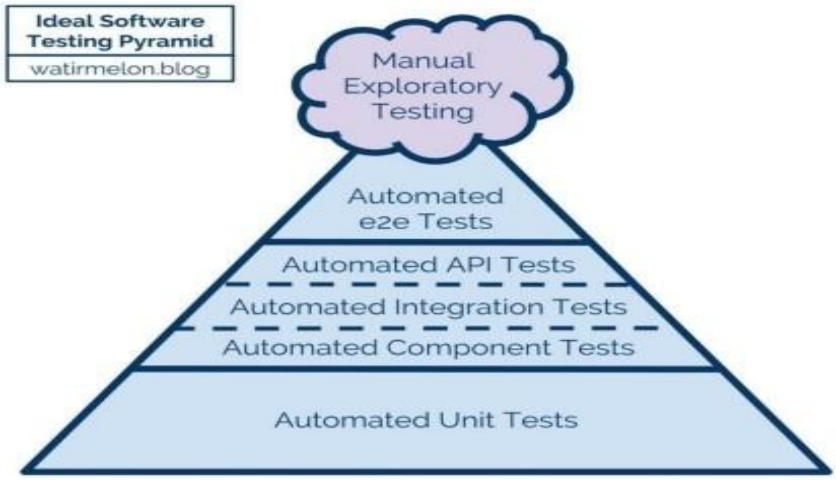
6.2.4 Análisis Dinámico del código (DAST).

Para LÓPEZ, Michelle[43], es un análisis de caja negra cuyo objetivo es analizar las aplicaciones en busca vulnerabilidades, lanzando peticiones y analizando las respuestas que genera, para comprobar los resultados y determinar si funcionan correctamente.

Herramientas para realizar pruebas DAST:
ZAP, se posiciona entre el navegador y la aplicación web para revisar los mensajes que se envían entre estos, modificar el contenido si es necesario y reenviar al destino.

Estas pruebas son más complejas de automatizar, para esto se debe tener en cuenta su grado de automatización, según las siguientes imágenes definidas por MikeCohn:

Figura 13. Pirámide de Cohn



Fuente: CAÑO QUINTERO, Jose Joaquin. DevSecOps: integración de herramientas SAST, DAST y de análisis de Dockers en un sistema de integración continua. [en línea]. Trabajo de Grado. Universitat Oberta de Catalunya, Catalunya.: 2019. [Consultado 10, marzo,2023]. Disponible en: https://openaccess.uoc.edu/handle/10609/95987

Conocer los recursos más importantes a nivel mundial, que describen en detalle las características de las vulnerabilidades más críticas de las aplicaciones web, además, los tipos de pruebas que podrían ayudar a detectar ágilmente dichas vulnerabilidades, es uno de los primeros pasos para uno de los componentes con más impacto en cualquier modelo de S-SDLC, la concientización y capacitación.

[43] LÓPEZ RODRIGUEZ, Michelle. Análisis de contenedores Docker e integración de herramientas SAST y DAST. [en línea]. Trabajo de Grado. Universidad Autónoma de Barcelona, Barcelona.: 2020. [Consultado 28, marzo,2023]. Disponible en: https://ddd.uab.cat/record/226837

3. INTEGRAR LAS METODOLOGÍAS, TÉCNICAS Y PROCEDIMIENTOS DE DESARROLLO DE SOFTWARE SEGURO CON BASE A LA GUÍA OWASP QUE PUEDEN APLICARSE CON SCRUM PARA OBTENER SOFTWARE DE CONFIANZA FRENTE A ATAQUES MALICIOSOS

En este capítulo se describe como se realizará la integración de los 3 proyectos más importantes del OWASP, así mismo, como sus formas ágiles se integran con el framework Scrum. También, se relacionará y unificarán las actividades recomendadas por estos marcos en cada una de las fases de S-SDLC. Por su puesto estarán descritas las herramientas, buenas prácticas, procedimientos y recomendaciones aplicadas durante la integración del modelo.

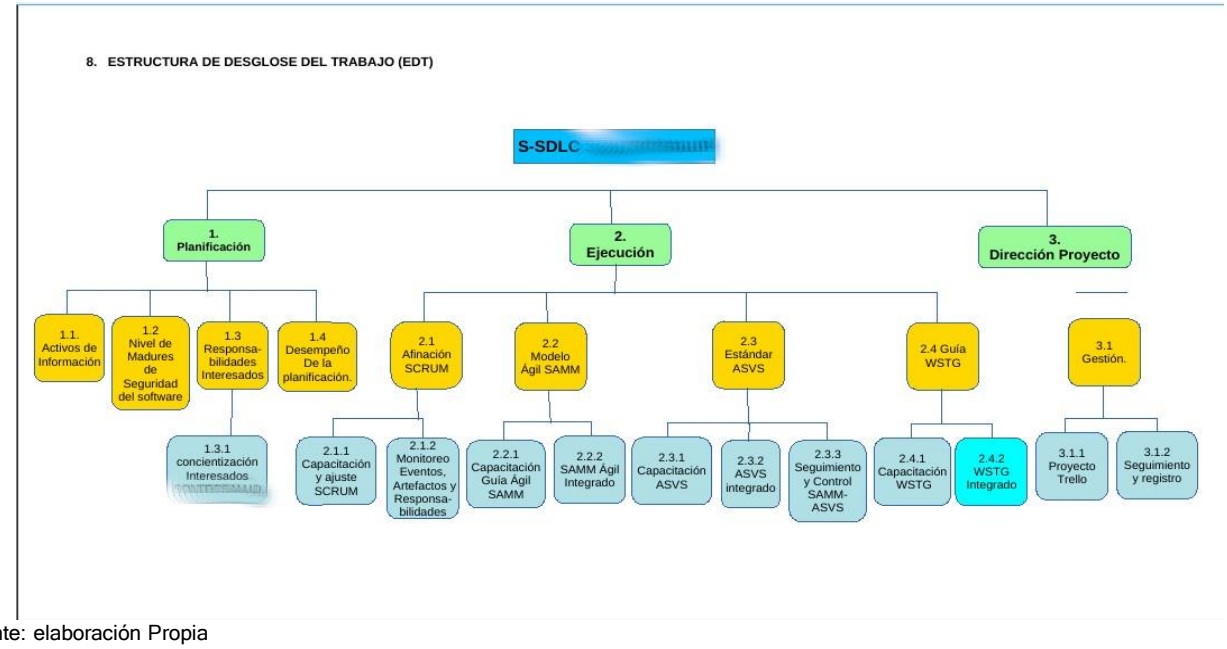
1. Gestión de la integración.

Para la gestión general de este proyecto se alinea con algunos dominios del estándar de dirección de proyectos PMBOK 7ª adaptándolo al contexto de la organización SAC para tener un panorama más claro con la gestión y así alcanzar los objetivos propuestos.

1. Planificación.

En la etapa de la planificación, se socializa el *Plan de Gestión del Alcance* del proyecto a todo el personal de SAC, que tiene como propósito aplicar el modelo de ciclo de vida de desarrollo de software seguro de OWASP. Se define una de las aplicaciones, en este caso la que mayor ingreso genera; se define el director del proyecto, el equipo de trabajo, las responsabilidades de los interesados, los paquetes de trabajo, entre otras delimitaciones importantes.

Figura 14. EDT SAC



Fuente: elaboración Propia

Para alcanzar este propósito, se adaptó la metodología suministrada por el OWASP con 3 de sus proyectos insignia de desarrollo de software seguro, teniendo como base el framework SCRUM:

I.GUÍA ÁGIL DEL SOFTWARE ASSURANCE MATURITY MODEL. (OWASP SAMM) Versión 2.0.3 (2022).

II. APPLICATION SECURITY VERIFICATION STANDARD 4.0.3 (ASVS).

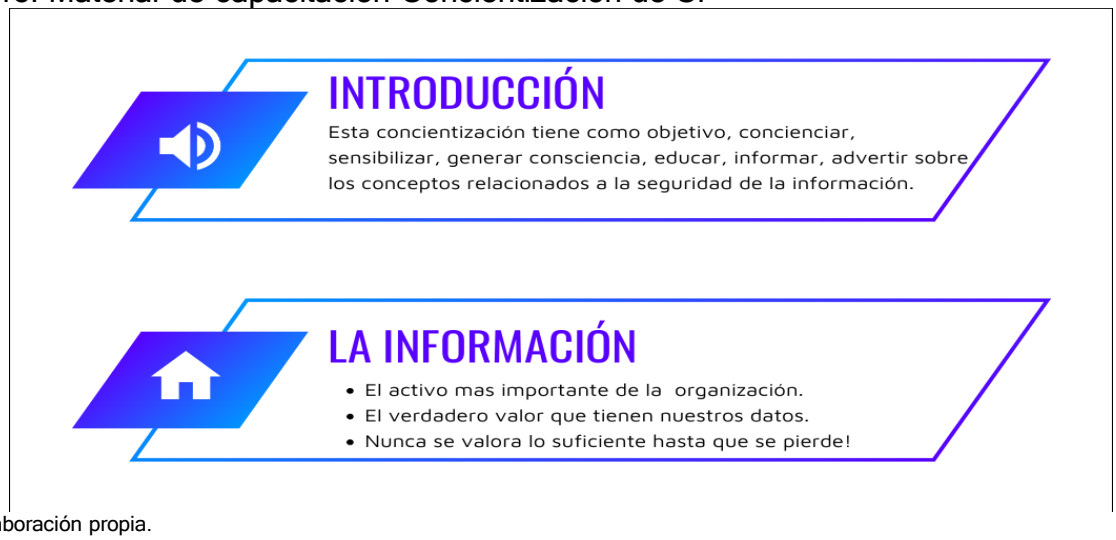
III. OWASP WEB SECURITY TESTING GUIDE V4.2 (WSTG).

Cada una de las etapas que se describen a continuación tuvo un componente de capacitación general de cada framework, qué se aplicara y como se debe integrar al ciclo de vida de desarrollo seguro (S-SDLC). La colaboración y constante comunicación entre los interesados fue fundamental para el éxito del proyecto.

6.3.1.2 Ejecución.

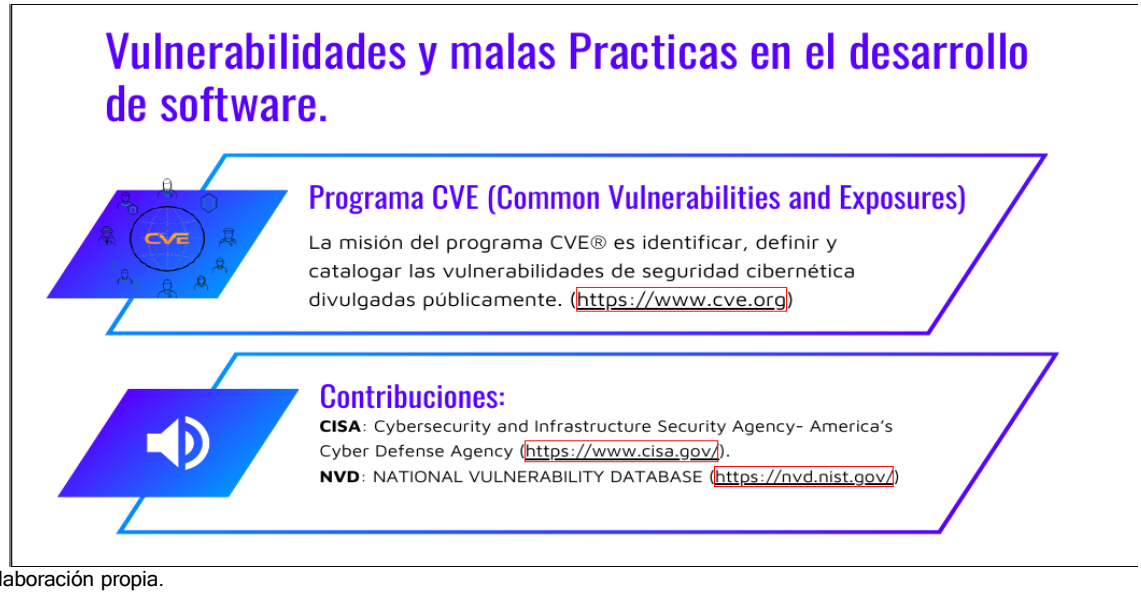
En la etapa de *Ejecución* se inicia con la concientización sobre los conceptos relacionados a la seguridad informática y desarrollo de software seguro a todo el equipo de trabajo; esto les permitió prepararse, saber lo importante que son para acompañar la estrategia y crear cultura de seguridad informática en la organización.

Figura 15. Material de capacitación Concientización de SI



Fuente: elaboración propia.

Figura 16. Material de capacitación Concientización de SI- CVE



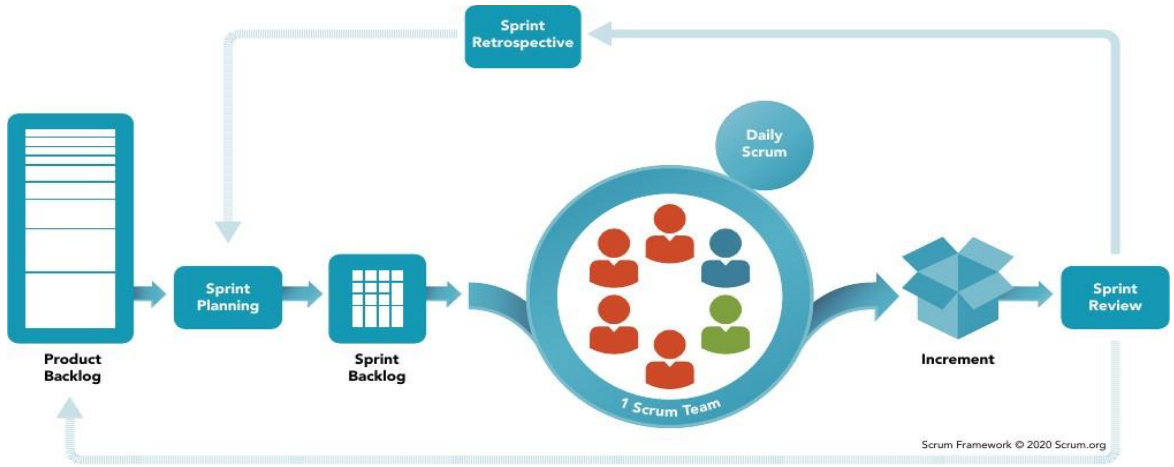
Fuente: elaboración propia.

Primero se revisó y se depuró el correcto uso del framework SCRUM, este es el eje central para el desarrollo de software con pensamiento ágil y sería la base para integrar la guía ágil SAMM.

6.3.2 Framework Scrum como base

El objetivo es aprovechar las ventajas del framework SCRUM que permite el manejo de las fases de SDLC de forma continua, con iteraciones cortas y con una retroalimentación en cada fase, teniendo previsibilidad, control del riesgo y adaptación a los cambios de manera inmediata.

Figura 17. Framework SCRUM



Fuente: WHAT IS Scrum? [Anónimo] [imagen]. [Consultado el 12, marzo, 2023]. Disponible en Internet: <http://www.scrum.org/learning-series/what-is-scrum/>.

Este framework con su enfoque iterativo e incremental hace parte de las metodologías de desarrollo ágiles que permite la iteración de las fases del SDLC. Esto permitirá tener en cuenta la seguridad en cada una de las fases, desde la inicial hasta la final. En la siguiente imagen podemos ver el SDLC ágil.

Figura 18. Proceso Metodología Ágil

Fuente: The Agile Process 101: Understanding the Benefits of Using Agile Methodology [Anónimo] [imagen]. [Consultado el 10, octubre, 2023]. Disponible en Internet: <https://www.nvisia.com/insights/agile-methodology>.

Uno de sus pilares es la transparencia, fundamental para el control y seguimiento, la inspección de los requisitos de seguridad que permite adaptarse de acuerdo con los eventos que se vayan presentando. SCRUM permite incluir las metodologías y prácticas de desarrollo seguro[44].

Por consiguiente, se integra la metodología de desarrollo de software seguro al framework para cumplir con los requisitos de seguridad de software de manera iterativa, incluyendo todo el ciclo de vida de desarrollo seguro S-SDLC.

[44] scrumguides.org. La Guía de Scrum. [Sitio WEB]. La entidad. [consultado el 28 de Enero de 2023]. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

6.3.3 Afinamiento al proceso actual Scrum.

Se da inició con el paquete de trabajo 2.1 y su componente de capacitación del framework SCRUM, importante que todo el equipo se capacite.

Figura 19 Material capacitación Scrum

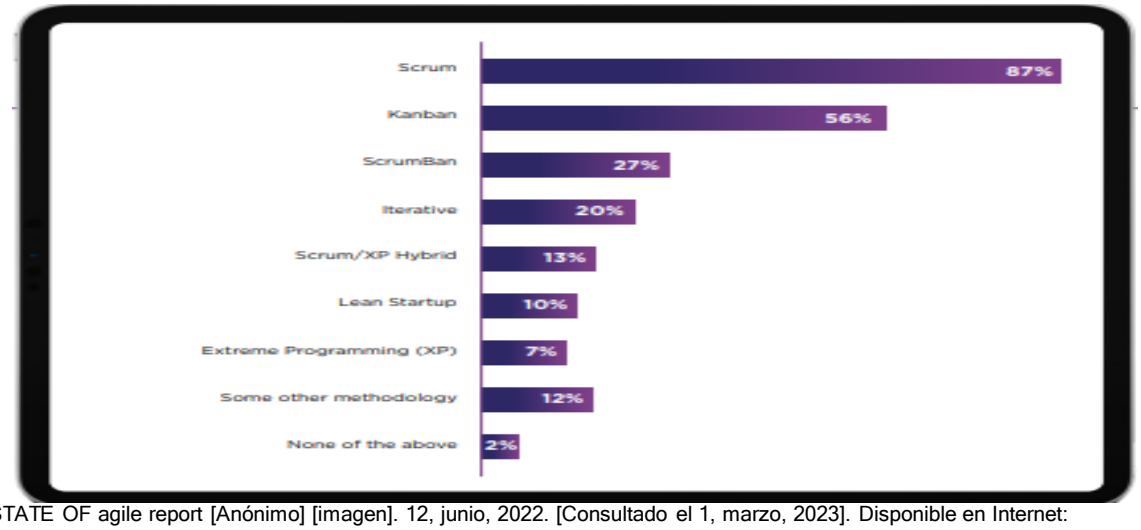
Fuente: elaboración propia.

Fue fundamental el seguimiento al uso adecuado de este marco de trabajo donde se verificó el correcto uso de sus roles, eventos y artefactos, así como la correcta definición de las Historias de usuario, siendo el artefacto más importante para integrar la seguridad en este modelo de desarrollo de software seguro.

En el informe anual de State Of Agile Reporte, para el año 2022, de los encuestados de su comunidad, el 80% están utilizando metodología Ágil como su enfoque principal. Así mismo, "Scrum continúa liderando, aumentando de 58% en la encuesta 14 a 87% en la encuesta actual. Kanban, su uso se ha disparado del 7% en la 14ª encuesta al 56% en la encuesta actual."[45]

[45] STATE OF agile report [Anónimo] [en línea]. [s.l.]: stateofagile, 2022 [consultado el 1, marzo, 2023]. 22 p. 2022-16th. Disponible en Internet: <https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf >.

Figura 20. State Of Agile Reporte



Fuente: STATE OF agile report [Anónimo] [imagen]. 12, junio, 2022. [Consultado el 1, marzo, 2023]. Disponible en Internet: <https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf>.

Para ajustar el proceso del Framework SCRUM a un procedimiento más fácil de asimilar por los funcionarios de SAC, se utilizó la investigación realizada por César, *et al*, *Modelo de referencia para la adopción e implementación de Scrum en la industria de software,* quienes presentan un modelo para facilitar la implementación de SCRUM en organizaciones de software. En su implementación se encontraron con inconvenientes que se pueden presentar en la implementación del framework:

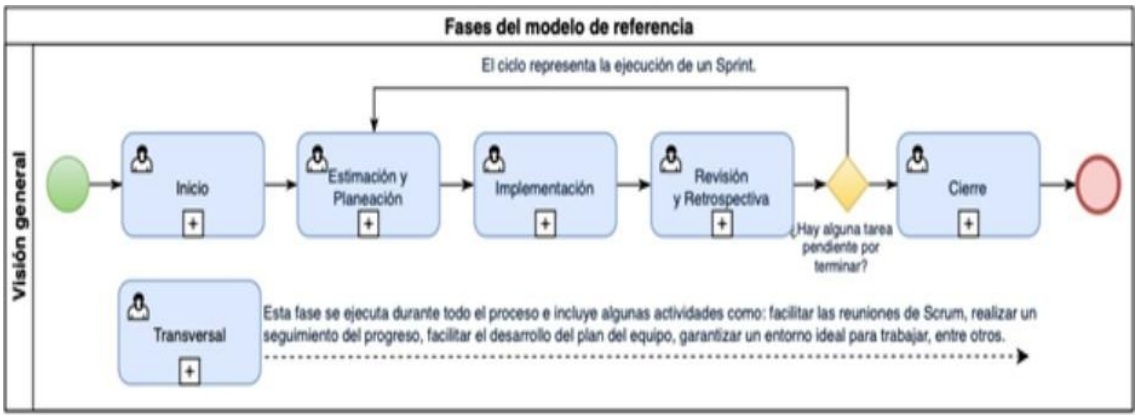
> "i) problemas al definir el tablero de tiempo del sprint con relación a la velocidad del equipo;
> (ii) dificultad para definir los tiempos y objetivos de las reuniones;
> (iii) problemas para generar adecuadamente la acumulación de productos;
> (iv) falta de conocimiento en la composición del equipo;
> (v) falta de conocimiento del nivel de detalle en las historias de los usuarios;
> (vi)dudas al generar y actualizar los artefactos, entre otros.
> Además, la falta de formación y conocimiento de los roles,
> así como problemas relacionados con el cambio de enfoque y paradigma en los equipos,
> la falta de transformación de la cultura operativa y organizacional,
> y la no adopción del manifiesto ágil y valores de Scrum, también pueden generan problemas."[46]

Algunos pasos de su metodología fue seleccionar las guías, descomponerlas, igualarlas y diseñar el modelo. El resultado fue el modelo de referencia para la implementación de SCRUM, el cual plantea un propósito, objetivos, roles, 6 fases, 29 actividades y procesos para una visión más detallada y práctica.

[46] PARDO, César, *et al.* Modelo de referencia para la adopción e implementación de Scrum en la industria de software. En: Investigación e innovación en ingenierías [en línea]. 2020. vol. 8, no.3 [consultado el 6, febrero, 2023], p.15. Disponible en Internet: <https://dialnet.unirioja.es/servlet/articulo?codigo=7799075>. ISSN 2344-8652.

En este modelo se tomarán los 3 roles principales del framework SCRUM, SM, PO y DevT

Figura 21. Fases del modelo de referencia Scrum



Fuente: Modelo de referencia para la adopción e implementación de Scrum en la industria de software [PARDO, César, *et al.*] [imagen]. [Consultado el 25, marzo, 2023]. Disponible en Internet: <https://dialnet.unirioja.es/servlet/articulo?codigo=7799075>.

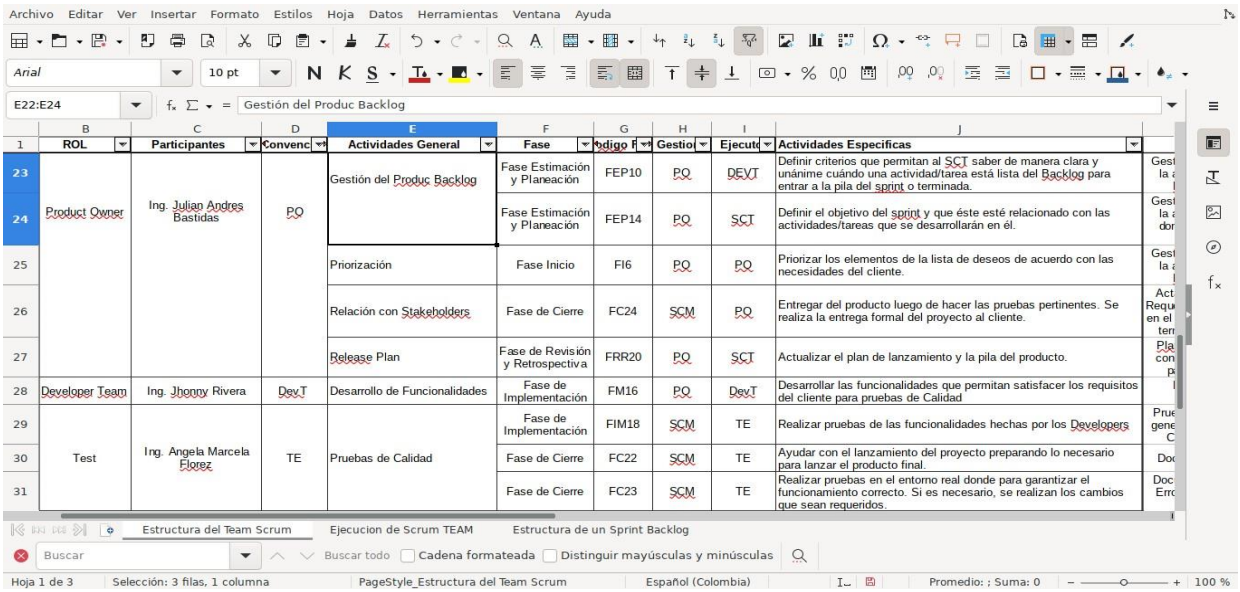
Todo esto facilitaría la integración de las fases del S-SDLC, con las fases de la guía ágil SAMM, SAVS y WSTG, homologando sus fases y adicionando cada una de las actividades o prácticas de cada fase.

Se pudo comprobar una de las definiciones de SCRUM de la investigación de Marco Vinicio, *et al* "cuyo objetivo es el control permanente del estado actual del software, donde el cliente establece las prioridades; mientras que el equipo SCRUM se autoorganiza a fin de determinar la mejor forma de entregar los resultados" [47]

La lista de actividades de las fases de SCRUM, fue adaptada para las necesidades específicas de SAC:

[47] ESTRADA-VELASCO, Marco Vinicio, *et al.* Revisión sistemática de la metodología scrum para el desarrollo de software. En: Dominio de las ciencias [en línea]. 2021. vol.7, no.4 [consultado el 9, febrero, 2023]. Disponible en Internet: <https://dominiodelasciencias.com/ojs/index.php/es/article/view/2429/0>.

Figura 22. Ajuste SCRUM - SAC

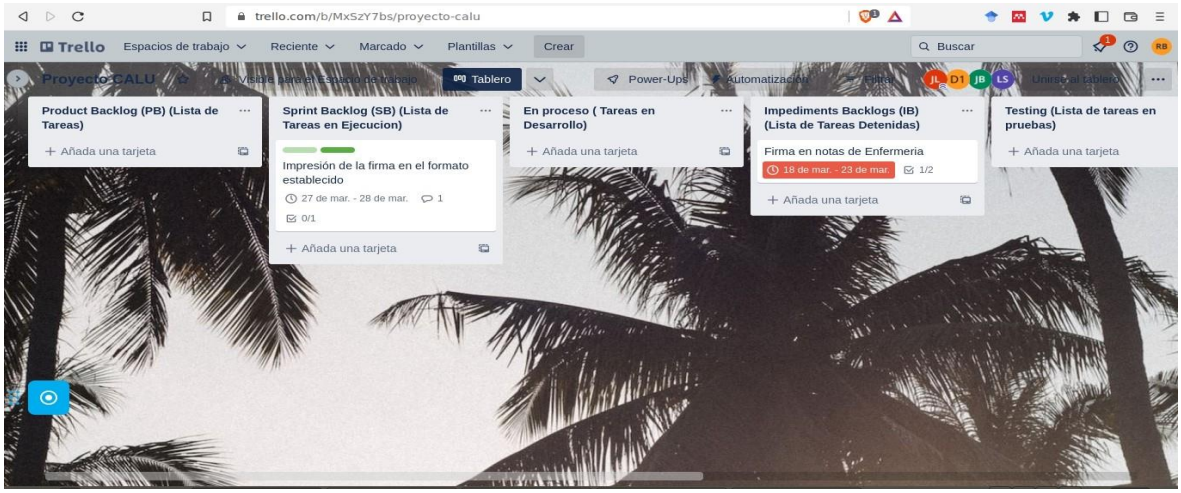


	B	C	D	E	F	G	H	I	J
1	ROL	Participantes	Convenc	Actividades General	Fase	odigo F	Gestio	Ejecute	Actividades Especificas
23	Product Owner	Ing. Julian Andres Bastidas	PO	Gestión del Produc Backlog	Fase Estimación y Planeación	FEP10	PO	DEVT	Definir criterios que permitan al SCT saber de manera clara y unánime cuándo una actividad/tarea está lista del Backlog para entrar a la pila del sprint o terminada.
24					Fase Estimación y Planeación	FEP14	PO	SCT	Definir el objetivo del sprint y que éste esté relacionado con las actividades/tareas que se desarrollarán en él.
25				Priorización	Fase Inicio	FI6	PO	PO	Priorizar los elementos de la lista de deseos de acuerdo con las necesidades del cliente.
26				Relación con Stakeholders	Fase de Cierre	FC24	SCM	PO	Entregar del producto luego de hacer las pruebas pertinentes. Se realiza la entrega formal del proyecto al cliente.
27				Release Plan	Fase de Revisión y Retrospectiva	FRR20	PO	SCT	Actualizar el plan de lanzamiento y la pila del producto.
28	Developer Team	Ing. Jhonny Rivera	DevT	Desarrollo de Funcionalidades	Fase de Implementación	FM16	PO	DevT	Desarrollar las funcionalidades que permitan satisfacer los requisitos del cliente para pruebas de Calidad
29	Test	Ing. Angela Marcela Florez	TE	Pruebas de Calidad	Fase de Implementación	FIM18	SCM	TE	Realizar pruebas de las funcionalidades hechas por los Developers
30					Fase de Cierre	FC22	SCM	TE	Ayudar con el lanzamiento del proyecto preparando lo necesario para lanzar el producto final.
31					Fase de Cierre	FC23	SCM	TE	Realizar pruebas en el entorno real donde para garantizar el funcionamiento correcto. Si es necesario, se realizan los cambios que sean requeridos.

Fuente: elaboración Propia.

Se aplicaron los cambios al proyecto en curso:

Figura 23. Scrum Board en Trello



Fuente: Elaboración Propia.

Con esta adaptación del desarrollo de software de SAC con SCRUM se pudo tener más claridad en la integración de las fases de este marco de trabajo con el S-SDLC de OWASP.

6.3.4 SAMM guía Ágil.

Recordemos que el OWASP SAMM, es el modelo de madurez de aseguramiento de software, cuya estructura y configuración le permitirá a SAC evaluar su estado actual de madurez e implementar la estrategia para llegar al siguiente nivel de seguridad de software. Este se integrará en cada fase del SDLC Ágil de SAC.

- Es un punto de partida para implementar un programa de seguridad.

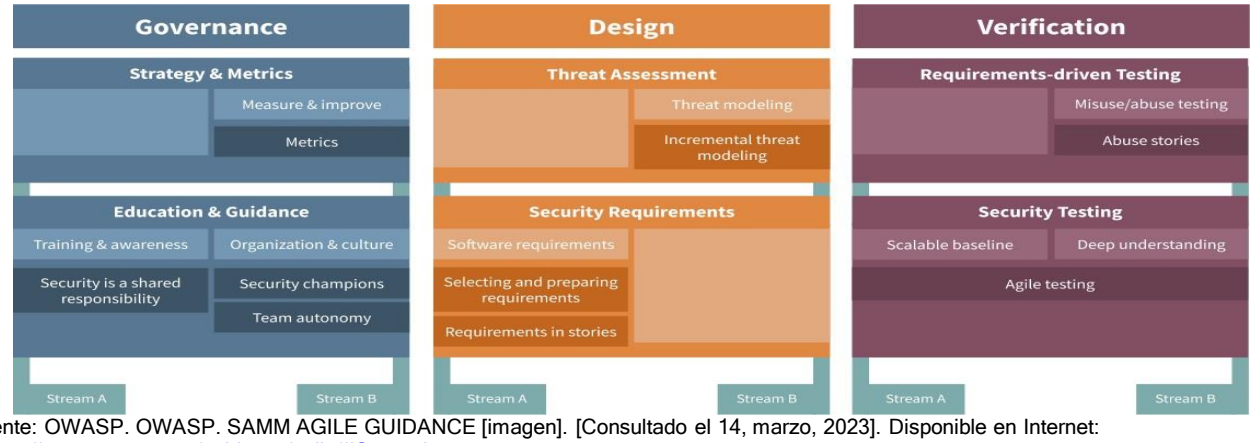
- Facilita la puesta en marcha del programa de seguridad del software.

Este será el modelo general de seguridad propuesto, pues permitió conocer la postura actual de seguridad y lo que se necesita para mejorar el aseguramiento del software, aunque es un modelo robusto, OWASP pensó en ajustarlo a las metodologías ágiles, así es como nos ofrece la GUÍA ÁGIL SAMM, haciendo más fácil la integración con el framework SCRUM.

Según Rob van der Veer[48], fundador del proyecto SAMM, la guía ágil de SAMM ofrece en detalle de como las actividades son un poco diferentes para ágil. Detalla el cómo implementar actividades en Ágil y las dificultades que se deben tener en cuenta.

A continuación, se describe las funciones de negocio y las estrategias de seguridad que define esta guía:

Figura 24. Relación Ágil-SAMM



Fuente: OWASP. OWASP. SAMM AGILE GUIDANCE [imagen]. [Consultado el 14, marzo, 2023]. Disponible en Internet: <https://owaspsamm.org/guidance/agile/#General >.

6.3.4.1 Gobernance.

ESTRATEGY & METRICS.

Medir y Mejorar. Los bucles para retroalimentar son cortos y se debe verificar con frecuencia. Es necesario tener datos para poder medir y con base en estos, mejorar.

[48] OWASP. SAMM AGILE GUIDANCE. [Sitio WEB]. La entidad. [consultado el 17 de marzo de 2023]. Disponible en: https://owaspsamm.org/guidance/agile/#General

✓Se debe tener por ejemplo un dashboard con la madurez y resultados que le sirva a los stakeholter para la toma de decisiones.

EDUCATION & GUIDANCE.
Entrenamiento y concientización.
La seguridad es una responsabilidad compartida. Para entornos ágiles, la responsabilidad de la seguridad es compartida por todos los miembros del equipo.

✓ Es fundamental incluir la seguridad desde la fase de planificación.

Actualmente, el valor comercial lo miden en cuanto a las características de funcionalidad de las aplicaciones y la seguridad como un costo adicional, pero debería verse como lo necesario para tener calidad, evitar daños y sobre todo la protección del valor del negocio.

✓Los demos deben integrar lo realizado en materia de seguridad, así los stakeholters experimentarán su importancia.

Organización y cultura.
El *Security champions*, es de gran apoyo, un líder en seguridad que ayude a fundamentar el conocimiento del equipo de desarrollo:

✓ Enlace entre desarrolladores y seguridad de información para el desarrollo seguro.

El equipo debe ser autónomo, debe aprender de los expertos en seguridad y con la ayuda de la automatización, la seguridad irá escalando en cada una de las fases SDLC.

✓ La seguridad debe estar presente en el refinamiento y en la retrospectiva.

6.3.4.2 Desing.

THREAT ASSESSMENT.
Modelado de amenazas incremental. El modelado de amenazas permite al equipo:

✓ Visualizar un mapa mental de posibles ataques.
✓ Seleccionar los requisitos de seguridad adecuados en las HU.
✓ Se puede incrementar un nuevo requisito de seguridad en cada nueva HU, así mismo debería ampliar el modelado para identificar nuevas amenazas y eventualmente actualizarse para corregir errores de cada incremento.

El modelado de amenazas no debe ser la única forma de integrar seguridad, se puede complementar:

✓Con una lista de requisitos que se puedan seleccionar fácilmente para cada tipo de desarrollo.

SECURITY REQUIREMENTS.

Requisitos de software.

Seleccionando y preparando requisitos. En cada sprint algunos requisitos deben ser seleccionados y probados, para que sea posible es necesario un proceso para seleccionar un conjunto reducido de requerimientos e instrucciones a desarrolladores; y pruebas importantes por sistema y por Historia.

POR SISTEMA:

Son requisitos generales según estándares y cumplimiento de la industria, en este caso el sector salud. La selección tiene como base conceptos como el rol, los riesgos de la tecnología, etc, que al ser requisitos generales de la aplicación, integrarían la *definición de Terminado,* DoD por sus siglas en inglés.

Los requisitos especiales de la HU se seleccionan según su función y según el desarrollo específico que se va a realizar, estos integrarían los *criterios de aceptación* de la Historia de usuario como instrucciones para los desarrolladores; y pruebas automatizadas e instrucciones para tester.

POR HISTORIA:

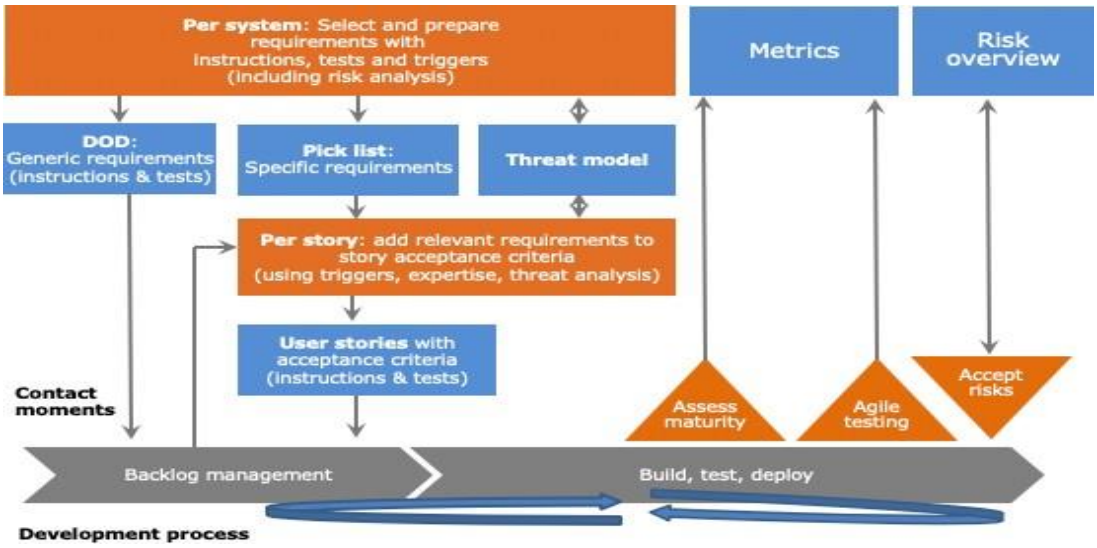
Los requisitos de seguridad específicos de cada historia se seleccionan durante la creación de la Historia de usuario y el refinamiento del backlog (Con ayuda del experto de Seguridad). Se eligen:

- ✓ Utilizando triggers en la lista de selección. Según su función agrupa un conjunto de requisitos.
- ✓ Según la experiencia en seguridad.
- ✓ Con historias de abuso, son el “como” se puede vulnerar el sistema.
- ✓ Si no es suficiente se debe usar modelado de amenazas.

Estos requisitos se integran en los criterios de aceptación para que los desarrolladores los tengan presente al momento de la planificación.

La siguiente imagen muestra una visión general del SDLC con relación al flujo de requisitos de seguridad.

Figura 25. Flujo de requisitos



Fuente: OWASP. OWASP. SAMM AGILE GUIDANCE [imagen]. [Consultado el 15, marzo, 2023]. Disponible en Internet: <https://owaspsamm.org/guidance/agile/#General >.

Requisitos en Historias.

Las historias se deben cargar con los requisitos de seguridad más importantes e incluirlos en los criterios de aceptación según el trabajo a realizar, para que los desarrolladores los tengan en cuenta en la planificación y pruebas. Recordar que se debe hacer en la creación de la historia y en el refinamiento del backlog y con la ayuda del experto en seguridad.

- ✓ La clave de la seguridad Ágil es aplicar los requisitos correctos en el momento adecuado.
- ✓ La selección de requisitos se debe hacer en la creación de la historia y en el refinamiento del backlog, no en la planificación.

3.Verification. Testing
basado en requisitos.
Pruebas de mal uso/abuso.
Historias de Abuso. Es la descripción de cómo se puede "abusar" un sistema a través de una vulnerabilidad, desde la perspectiva del atacante. Ayuda al equipo a saber que podría salir mal para seleccionar los requisitos específicos para cada caso. Por ejemplo, podría explicar a un pentester contra que amenaza debe probarse el requisito. En todo caso, es un complemento a las formas de selección de requisitos de seguridad que se relacionaron anteriormente.

Testing de seguridad.
Testing Agil. En un entorno Ágil, se realizan pruebas con mayor frecuencia, es fundamental automatizarlas tanto como se pueda, programando pruebas unitarias y de nivel de integración con herramientas de prueba como Comprobación de dependencias (SCA), Dynamic application security test (DAST) y Static analysis security test (SAST).

No se puede automatizar todas las pruebas, el tiempo invertido en las pruebas manuales puede reducirse si se enfoca al código añadido o modificado, según el comportamiento que pueda verse afectado por el cambio. Se recomienda realizar pruebas de penetración completa.

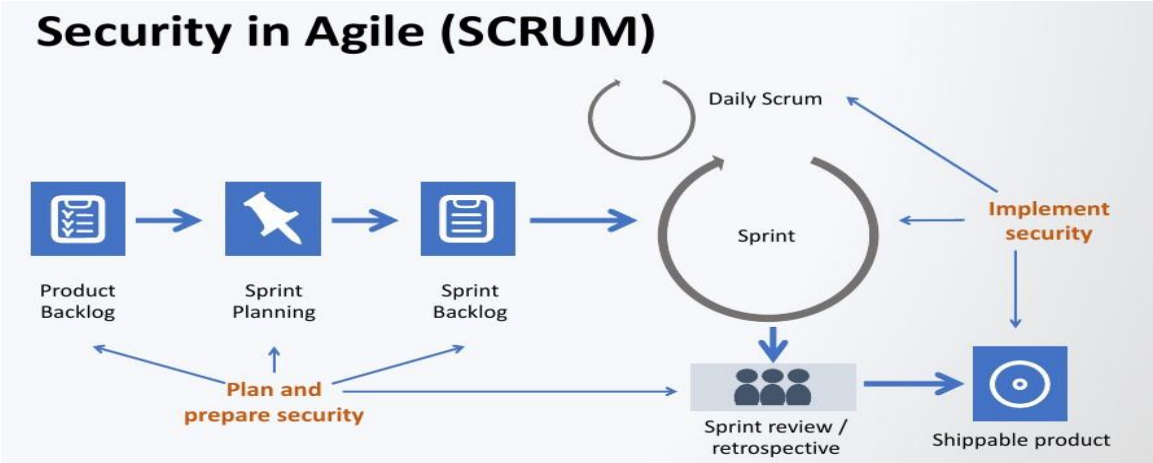
En este entorno es necesario la selección de pruebas basadas en cambios, por eso es importante que el sistema sea modular y poder probar los módulos por separado. Este trabajo manual debe tenerse en cuenta para realizar su planificación.

Para Ágil se recomienda tener competencia de QA en el equipo de desarrollo que incluiría la competencia en pruebas de seguridad.

- ✓ No se debe omitir las pruebas manuales, debe ser parte del control de calidad.
- ✓ Es un error realizar el testing tarde, antes del lanzamiento, cuando genera más costo y no hay tiempo.

Es importante conocer y complementar la formación que entrega la fundación OWASP en *SAMM Agile guidance* a través de su fundador Rob van der Veer[49] describe como construir seguridad continuamente apoyada en Ágil, y lista las recomendaciones.

Figura 26 Seguridad en SCRUM



Fuente: OWASP. Secure Agile development according to SAMM [imagen]. [Consultado el 15, marzo, 2023]. Disponible en Internet: <https://drive.google.com/file/d/14aaHVZtfOGzBj2JKi1GXRtvAeN22_zEf/view >.

1. La seguridad no debería planificarse en el sprint planing, ya debería estar en la HU.
2. Automatizar lo que más se pueda, con herramientas.
3. Reducir la superficie de ataque, utilizar tecnología y bibliotecas probadas, no utilice criptografía propia, utilice lo que ya este hecho.
4. Las pruebas de lo realizado por desarrollo deben cambiar, debe estar orientada "hacia izquierda". El especialista de seguridad es asesor y apoyo … . No es calidad.
5. Todos deben tener seguridad, el PO, el test y desarrollo.
6. Agilizar el trabajo manual, no siempre se logra con el modelado de amenazas, sobre todo en cosas triviales.
7. Análisis vs. higiene: definir requisitos de seguridad generales sin hacer tanto análisis de riesgo... Higiene Básica.
8. Reducir la cantidad de ejecuciones que los desarrolladores tengas que realizar, menor cantidad de pruebas, automatizar.
9. Trigger para probar que se cumple los requisitos.
10. Algunos requisitos se deben obviar para desarrollo o diferir o aplazar. Otros no son relevantes según el caso o no aplica.
11. Hacer pruebas pequeñas.
12. Usar fuentes de requisitos como ASVS.
13. Nuevamente, en las historias de usuario y en el refinamiento se agregan requisitos relevantes, criterios de aceptación utilizando disparadores, vincule las herramientas que con ello vienen, las instrucciones y pruebas.

[49] OWASP. SAMM AGILE GUIDANCE. [Sitio WEB]. La entidad. [consultado el 20 de marzo de 2023]. Disponible en: https://owaspsamm.org/guidance/agile/#General

Con apoyo del profesional en seguridad en el refinamiento encuentre la higiene necesaria o realice el análisis de amenazas:

Historias de usuario (HU):

✓ Criterios de aceptación.

Esto lleva a revisar en la Defintion Of Done (DoD):

✓ Que los controles de seguridad generales estén.
✓ Instrucciones para los desarrolladores.
✓ Pruebas que se deben realizar.
✓ Que se validen los requisitos de seguridad.

6.3.5 Integración guía Ágil SAMM.

En un segundo momento, para el paquete de trabajo *2.2 Modelo Ãgil SAMM* (ya se había iniciado en el paquete 1.1), evaluaremos el estado actual de madurez de aseguramiento del software para implementar la estrategia que permita mejorar el nivel de seguridad de la aplicación seleccionada.

Se inicia con la primera actividad, el componente de capacitación que tiene como objetivo conocer el modelo SAMM y como su G*uía Ãgil* permite integrarse con Scrum.

Figura 27. Material de capacitación Guía Ágil SAMM

Fuente: Elaboración Propia.

Anteriormente, se realizó la autoevaluación que ofrece el modelo SAMM, aplicando la herramienta **SAMM_spreadsheet.sxlx.** Esta se puede descargar desde el sitio de OWASP SAMM que podemos ver en el link :

https://github.com/owaspsamm/core/releases/download/v2.0.3/SAMM_spreadsheet.xlsx

Esta contiene una serie de preguntas asociadas a cada una de las funciones y prácticas de seguridad del modelo SAMM, calcula el puntaje de cumplimiento actual de la organización para cada práctica de seguridad y describe la hoja de ruta para planificar las actividades a realizar para alcanzar el siguiente nivel de madurez en camino a mejorar el S-SDLC.
SAMM, es el programa de principio (políticas para manejo de la seguridad en la empresa) a fin (como Manejo de respuesta a incidentes), abarca más allá del desarrollo en sí, Inicia antes (Planificación) y termina después (Operaciones). Aunque que para este proyecto se aplicara su guía ágil, se realizaron algunas actividades del modelo SAMM general, de las corrientes que se evaluaron y podían complementar las actividades de la Guía Ágil.

6.3.5.1 Estado actual SAMM en SAC.

Se realizó la entrevista a manera de taller con las personas involucradas en S-SDLC.
Ver el archivo **SAMM_entrevistaSAC_FaseI.xlsx** hoja Interview en el link relacionado más adelante en el punto de definición de objetivo SAMM FASE I.

Ver resultado de entrevista en el link:
https://docs.google.com/spreadsheets/d/1bFN2v9O0ULnBVhho3cJgpKa8R0qDYSco/edit?usp=sharing&ouid=107699249681285147847&rtpof=true&sd=true

En cada función comercial (Gobernanza, Diseño, Implementación, Verificación y Operaciones), tiene relacionada dos corrientes o estrategias de seguridad, las cuales tienen una pregunta de acuerdo con cada nivel (0,1, 2, y 3); estas se responden con una lista de 4 (y el campo vació) respuestas relacionadas según el caso, además contiene una descripción general de lo que debería hacer para cumplir con dicha pregunta.

En la siguiente imagen se describe un ejemplo de cómo se clasifica el puntaje de la evaluación para una de las preguntas.

Práctica: Educación y Orientación.
Estrategia: Entrenamiento y concientización.
Pregunta:
¿Requiere que los empleados involucrados en el desarrollo de aplicaciones reciban capacitaciones en SDLC?

Figura 28 Niveles de Madurez

Nivel de madurez		Puntuación de la Evaluación	
0	Practica incumplida	0	NO
1	Disposición Ad-Hoc	0,25	Si, alguno de ellos.
2	Mayor Eficiencia y Eficacia.	0,5	Si, al menos la mitad de ellos
3	Dominio Integral a Escala.	1	Si, la mayoría o todos ellos.

Fuente: OWASP SAMM | OWASP Foundation [Anónimo] [imagen]. Enero, 2023. [Consultado el 20, febrero, 2023]. Disponible en Internet: <https://github.com/owaspsamm/core/releases/download/v2.0.3/SAMM_spreadsheet.xlsx>..

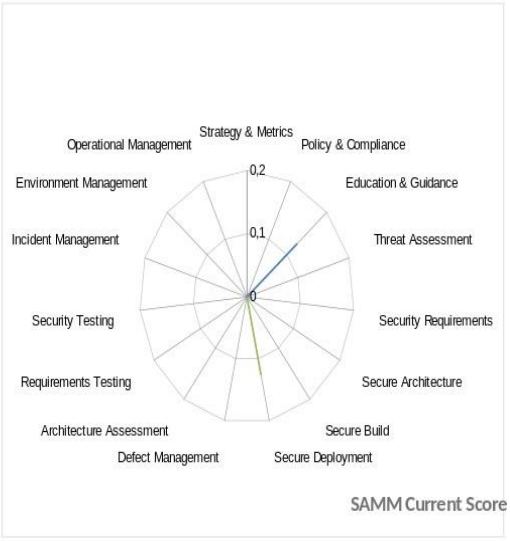
Análisis del dashboard de SAC:

Con esta Herramienta se tiene un panorama completo de la postura de seguridad de SAC. Para SAC es fundamental aplicar el modelo propuesto, pues su nivel de aseguramiento de software es casi nulo, solo en las prácticas de proceso de implementación cuenta con un puntaje de **0.13**, no aplica ninguna práctica de seguridad en el desarrollo de su software, SAC no tiene políticas, Modelado de amenazas, evaluación estática, dinámica y no hay higiene básica.

Figura 29. Nivel seguridad de software SAC

			Maturity		
Business Functions	Security Practices	Score	1	2	3
Governance	Strategy & Metrics	0,00	0,00	0,00	0,00
Governance	Policy & Compliance	0,00	0,00	0,00	0,00
Governance	Education & Guidance	0,13	0,13	0,00	0,00
Design	Threat Assessment	0,00	0,00	0,00	0,00
Design	Security Requirements	0,00	0,00	0,00	0,00
Design	Secure Architecture	0,00	0,00	0,00	0,00
Implementation	Secure Build	0,00	0,00	0,00	0,00
Implementation	Secure Deployment	0,13	0,13	0,00	0,00
Implementation	Defect Management	0,00	0,00	0,00	0,00
Verification	Architecture Assessment	0,00	0,00	0,00	0,00
Verification	Requirements Testing	0,00	0,00	0,00	0,00
Verification	Security Testing	0,00	0,00	0,00	0,00
Operations	Incident Management	0,00	0,00	0,00	0,00
Operations	Environment Management	0,00	0,00	0,00	0,00
Operations	Operational Management	0,00	0,00	0,00	0,00

Business Functions	Score
Governance	0,04
Design	0,00
Implementation	0,04
Verification	0,00
Operations	0,00
Overall	0,02



Fuente: elaboración Propia.

Después de conocer con más detalle la metodología del modelo SAMM y su guía ágil, se define el alcance del proyecto y se dará prioridad a las prácticas de seguridad relacionadas a la función GOBERNANZA, es crucial tener el apoyo de los stakeholters, las capacitaciones, no tener un gran presupuesto hace que se deban enfocar a cumplimiento de requisitos básicos de riesgo, capacitación y concientización; y sobre todo el objetivo será integrar la seguridad tanto como sea posible en cada una de las fases de S-SDLC. Si no se incluye la seguridad, se estará detrás de otras organizaciones, perderá credibilidad, la marca irá perdiendo valor comercial y sobre todo tendrá impacto negativo en futuras negociaciones.

El siguiente nivel de prioridad se dará a la función de DISEÑO, con las prácticas de Evaluación de amenazas y Requisitos de seguridad. Por último, en la función de VERIFICACIÓN con su práctica de seguridad, Pruebas de seguridad. Todas estás solo en el nivel L1 de la línea de acción (stream) acordada.

6.3.5.2 Definición del objetivo SAMM fase I.

Se define el plan objetivo de aseguramiento de software para la fase I en la hoja de ruta de la herramienta de evaluación.

La herramienta permite planear o definir las actividades que serán necesarias para lograr el objetivo según la Fase, se cuenta con 4 fases para lograr cumplir con las prácticas de seguridad en los 3 niveles de seguridad.

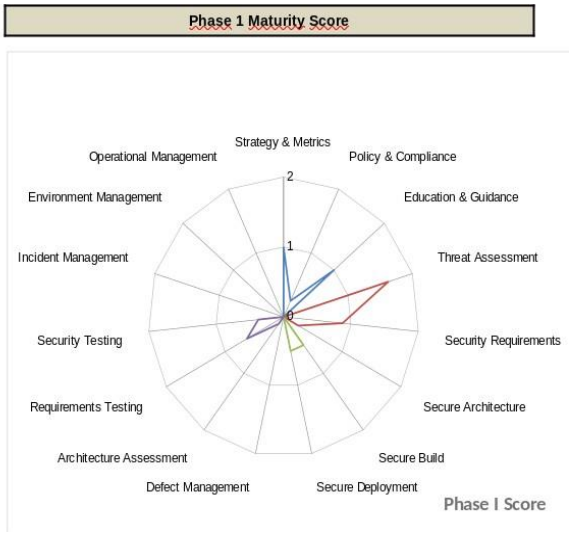
En la siguiente imagen se puede ver el objetivo de la fase I (inicial):

Figura 30. Plan objetivo Fase I

Phase 1 Maturity Score

Business Functions	Security Practices	Score	Maturity 1	2	3
Governance	Strategy & Metrics	1,00	0,38	0,13	0,50
Governance	Policy & Compliance	0,25	0,25	0,00	0,00
Governance	Education & Guidance	1,00	1,00	0,00	0,00
Design	Threat Assessment	1,63	0,63	0,00	1,00
Design	Security Requirements	0,88	0,25	0,50	0,13
Design	Secure Architecture	0,25	0,25	0,00	0,00
Implementation	Secure Build	0,50	0,50	0,00	0,00
Implementation	Secure Deployment	0,50	0,50	0,00	0,00
Implementation	Defect Management	0,00	0,00	0,00	0,00
Verification	Architecture Assessment	0,13	0,13	0,00	0,00
Verification	Requirements Testing	0,63	0,63	0,00	0,00
Verification	Security Testing	0,38	0,25	0,13	0,00
Operations	Incident Management	0,00	0,00	0,00	0,00
Operations	Environment Management	0,00	0,00	0,00	0,00
Operations	Operational Management	0,00	0,00	0,00	0,00

Business Functions	Score
Governance	0,75
Design	0,92
Implementation	0,33
Verification	0,38
Operations	0,00
Overall	0,48



Fuente: elaboración propia.

Ver objetivo fase I en el link:
https://docs.google.com/spreadsheets/d/1bFN2v9O0ULnBVhho3cJgpKa8R0qDYSco/edit?usp=sharing&ouid=107699249681285147847&rtpof=true&sd=true

Se cumplió con el objetivo inicial de la entrevista para la evaluación de prácticas de seguridad:

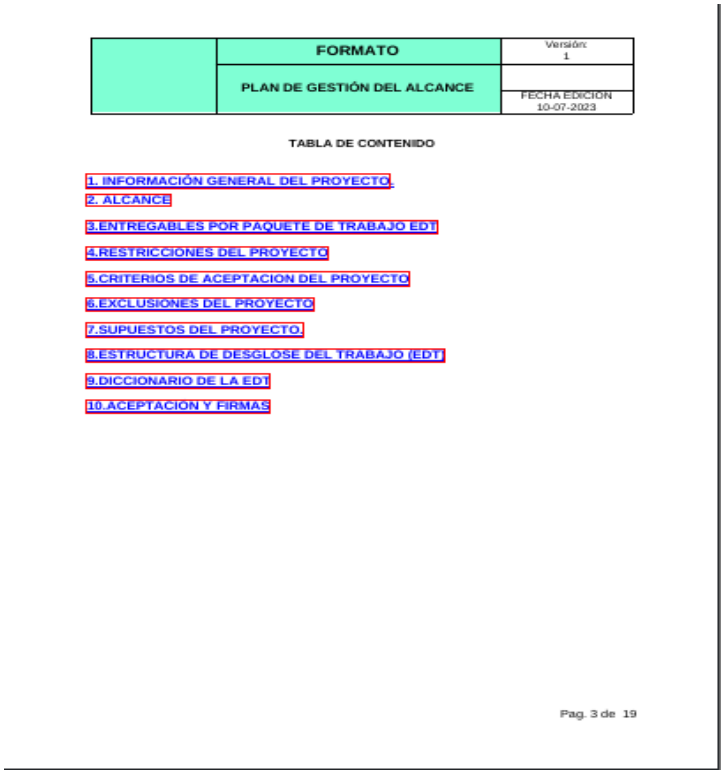
1. Se conoce el estado actual de las prácticas de seguridad de SAC.
2.Se define la estrategia o el plan objetivo (la hoja de ruta) de la primera fase, ajustando el alcance inicial del proyecto.
3.SAC comprende las recomendaciones para implementar actividades de seguridad para pasar al siguiente nivel de madurez.

A continuación, se mencionan las actividades más importantes realizadas para dar cumplimiento al plan objetivo de la fase I teniendo como base la guía ágil SAM y algunas de las actividades complementarias del modelo general.

GOBERNANZA.
ESTRATEGIA Y MÉTRICAS.

Crear y promover. En esta corriente se contempló la total atención y apoyo de los interesados en el alcance del plan estratégico de seguridad, el cual se presentó a gerencia y a los equipos de trabajo en el comité de gestión que se realiza en SAC; aceptando seguir y apoyar el plan estratégico de seguridad.

Figura 31. Presentación de la estrategia de seguridad



	FORMATO	Versión: 1
	PLAN DE GESTIÓN DEL ALCANCE	
		FECHA EDICION 10-07-2023

TABLA DE CONTENIDO

Pag. 3 de 19

Fuente: elaboración propia.

Medir y mejorar. Para que el plan objetivo fuera eficiente, se seleccionaron y priorizaron las actividades que permitirán obtener métricas y hacer el seguimiento a las prácticas de seguridad y su mejoramiento.

- ✓ Registro de las horas de capacitación.
- ✓ Registro de horas dedicadas al desarrollo de requisitos de seguridad.
- ✓ Datos generados del escaneo y modelado de amenazas, cantidad de vulnerabilidades pendientes, relacionando datos importantes como la prioridad e impacto.
- ✓ Cantidad de aplicaciones.
- ✓ Dashboard con el plan objetivo (hoja de ruta) para la toma de decisiones de los stakeholters.

EDUCACIÓN Y ORIENTACIÓN.

Entrenamiento y concientización. Para el equipo de trabajo se realizaron capacitaciones sobre concientización de la estrategia de seguridad, así como de los marcos utilizados para crear el S-SDLC de SAC:

- ✓ Framework SCRUM.
- ✓ Conceptos relacionados a la seguridad informática, como bases de datos de vulnerabilidades, CWE, NIST, amenazas, riesgo.
- ✓ OWASP Top 10 2021,
- ✓ OWASP SAMM
- ✓ OWASP SAMM guía Ágil.
- ✓ Modelado de amenazas.
- ✓ OWASP ASVS.
- ✓ Selección y preparación de requisitos de seguridad.
- ✓ OWASP Proactive Controls
- ✓ OWASP Cheat Sheeet Serie.

Para cumplir con las recomendaciones de la guía ágil de SAMM, en cada capacitación se enfatiza y se demuestra el porqué la seguridad debe ser compartida y estar desde la fase inicial hasta la final del S-SDLC, incluyendo la socialización de lo realizado en materia de seguridad en cada *sprint review* o demo.

Figura 32. Material de capacitación Modelado de amenazas

Fuente: elaboración propia.

Organización y cultura.

En la presentación del alcance de la estrategia de seguridad se definieron los líderes de los equipos, quienes serían los *Guardianes de la seguridad* en SAC, quienes ayudaran a investigar, verificar y priorizar los requisitos de seguridad, así como en la revisión de los problemas que se puedan presentar en cuanto la seguridad de las aplicaciones.

✓ Son el enlace entre desarrolladores y la seguridad de información para el S-SDLC.

Se prioriza la autonomía del equipo en temas de seguridad, presentes en el refinamiento del backlog y la retrospectiva.

DISEÑO.

Evaluación de amenazas.

Perfil de riesgo de la aplicación. A pesar de que esta corriente no la contempla la guía ágil, es necesario tener la capacidad de identificar que aplicaciones tienen un mayor riesgo para la organización y así concentrar los esfuerzos en materia de seguridad en estas aplicaciones.

✓Se creó un inventario de aplicaciones, donde relacionan una serie de preguntas que permitan clasificar las aplicaciones según el riesgo.

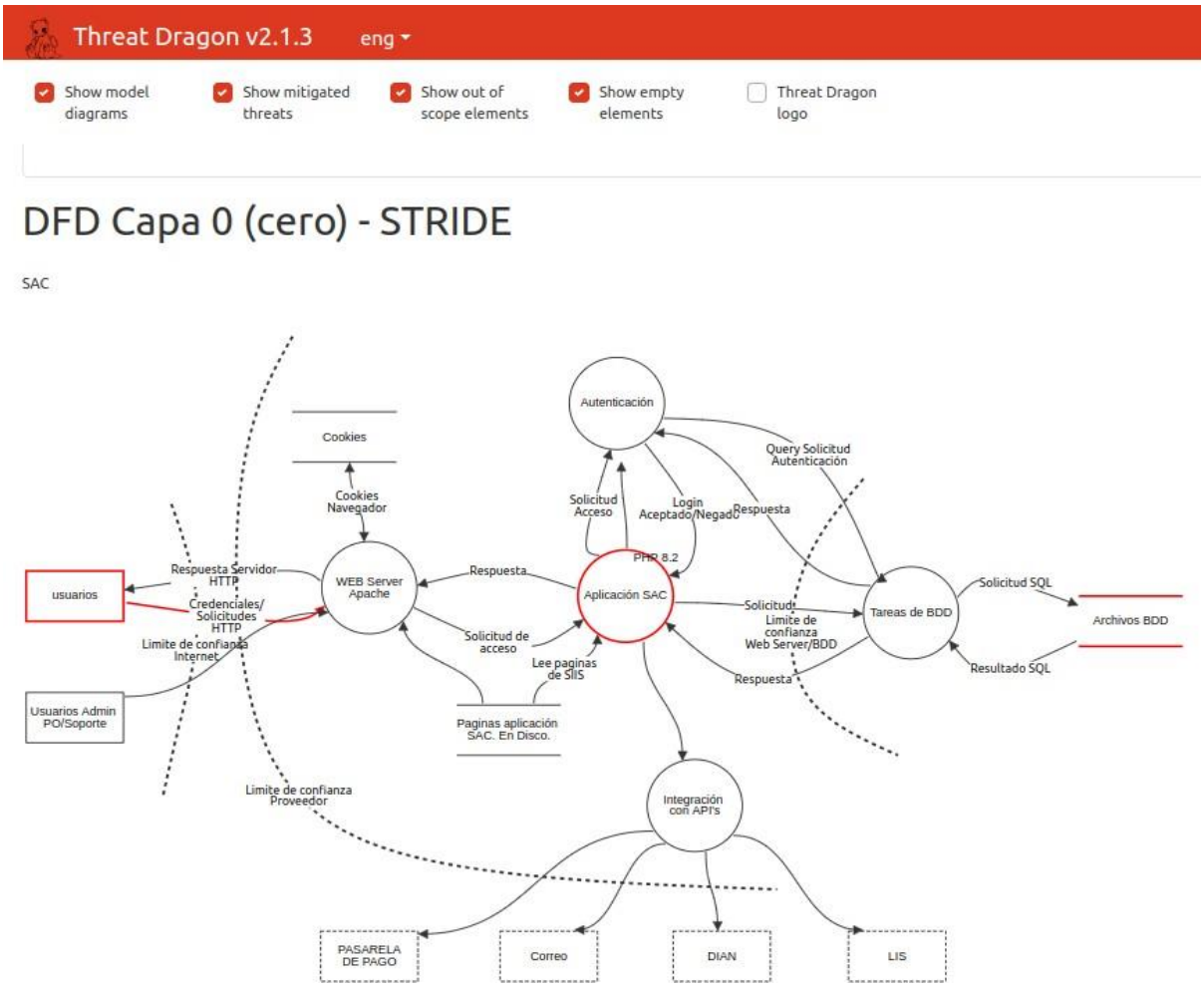
Modelado de amenazas. Con base en la prioridad definida en el perfil de riesgo de la aplicación, se realizó el modelado de amenazas para la aplicación seleccionada, permitiendo a los equipos de desarrollo, implementación y tester, identificar, clasificar y proponer las actividades de mitigación de los riesgos encontrados en esta actividad.

Se tuvieron en cuenta las recomendaciones de la guía:

- ✓ Creación de diagramas de amenazas con la herramienta THREAT DRAGON.
- ✓ Igualmente, se evaluó la herramienta Microsoft Threat Modeling Tool.
- ✓ Se clasificaron las amenazas según STRIDE.
- ✓ Se inició con diagramas de amenazas para desarrollos nuevos, que se van creando o actualizando con cada Sprint, cumpliendo con el modelado incremental.
- ✓ Se asocia al menos un nuevo requisito de seguridad en cada Sprint, sumando otro en el próximo sprint,
- ✓ La selección de requisitos de seguridad que mitigarían las amenazas clasificadas en el modelado y según el desarrollo a realizar, se integró con las listas de requisitos de ASVS.

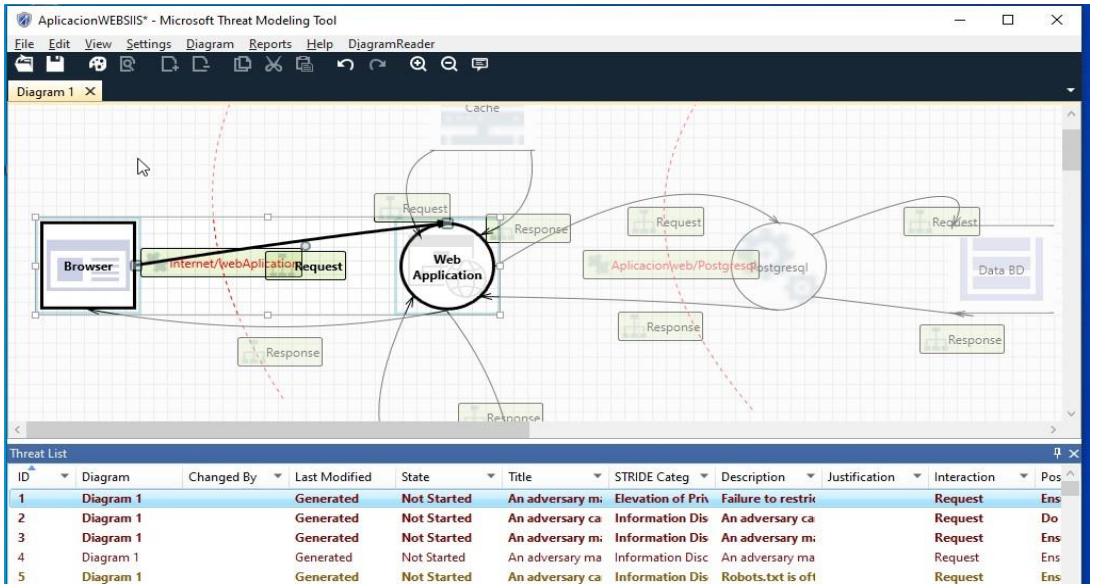
A continuación, se muestra la imagen del diagrama de Flujo de datos (DFD) del modelado de amenazas del primer nivel.

Figura 33. DFD aplicación selecciona SAC en TD



Fuente: elaboración propia.

Figura 34. DFD aplicación selecciona SAC en MMT



Fuente: elaboración propia.

REQUERIMIENTOS DE SEGURIDAD.

Requisitos de software.

Selección y preparación de requisitos. Concientizar y capacitar al equipo de trabajo para realizar una selección de los requisitos más relevantes, fue uno de los objetivos de cada inducción y reunión de seguimiento y control relacionada con la implementación de SCRUM, SAMM, del modelado de amenazas y la selección de requisitos del ASVS.

- ✓ En cada Sprint se identificó un conjunto mínimo de requisitos de seguridad según el tipo de desarrollo.
- ✓ Estos debían ser los más relevantes para mitigar la amenaza identificada y priorizada.
- ✓ En la selección de requisitos también se evaluó que estuvieran alineados con SAMM, ASVS, NIST o recomendaciones de seguridad de otros proyectos OWASP.
- ✓ Estas recomendaciones se registran como Definición de terminado (DoD).
- ✓ Como apoyo se analizaban las bases de datos de vulnerabilidades como CWE y sus recomendaciones de mitigación.

Requisitos en historias. Después de definir los DoD, como requisitos generales de las historias de usuario del spring, se adicionan los requisitos de seguridad específicos para cada una, según el DoD y el entorno del desarrollo, registrándose como criterios de aceptación. Esto se actualiza, si es el caso, en el refinamiento.

6.3.6 Estándar de verificación de aplicaciones ASVS-v4.0.3.

A continuación, analizaremos el APPLICATION SECURITY VERIFICATION STANDARD 4.0.3 (ASVS). Sabemos que el ASVS está alineado con los modelos de S-SDLC y según este estándar se debe identificar en qué nivel de requerimientos debería estar el HIS principal, y con ayuda de su check list de requisitos, precisar cuáles son necesarios para la aplicación según su nivel.

Este estándar de verificación de seguridad de aplicaciones facilita una serie de pruebas para los controles de seguridad de las aplicaciones web, podemos decir que es una lista de controles para evaluar si cumplimos con los requisitos de seguridad, la cual tenemos que incluir desde la fase inicial del S-SDLC, con este podemos definir las características de seguridad de las aplicaciones. ASVS tiene 3 niveles de profundidad, para este caso aplicado y según el plan objetivo definido para la primera fase, se priorizará el nivel 1(L1), tomando requisitos de valor del nivel 2 (L2) para aplicaciones que manejan datos confidenciales y es el recomendado, a pesar de que el HIS de SAC maneja registros médicos y debería tener el nivel 3, es mucho más complejo.

Se utiliza ASVS para definir requisitos de seguridad específicos que se definen según las recomendaciones de la guía ágil SAMM en las Historias de usuarios.

El estándar cuenta con 14 categorías de verificación, cada una cuenta con sus subcategorías y estás con la definición de sus requisitos y en qué nivel debe aplicarse, así como su CWE respectiva y su sección en la NIST[50]. Por ejemplo:

V2 Autenticación (categoría).
V2.1 Seguridad de Contraseña (subcategoría).

Figura 35. Check list ASVS

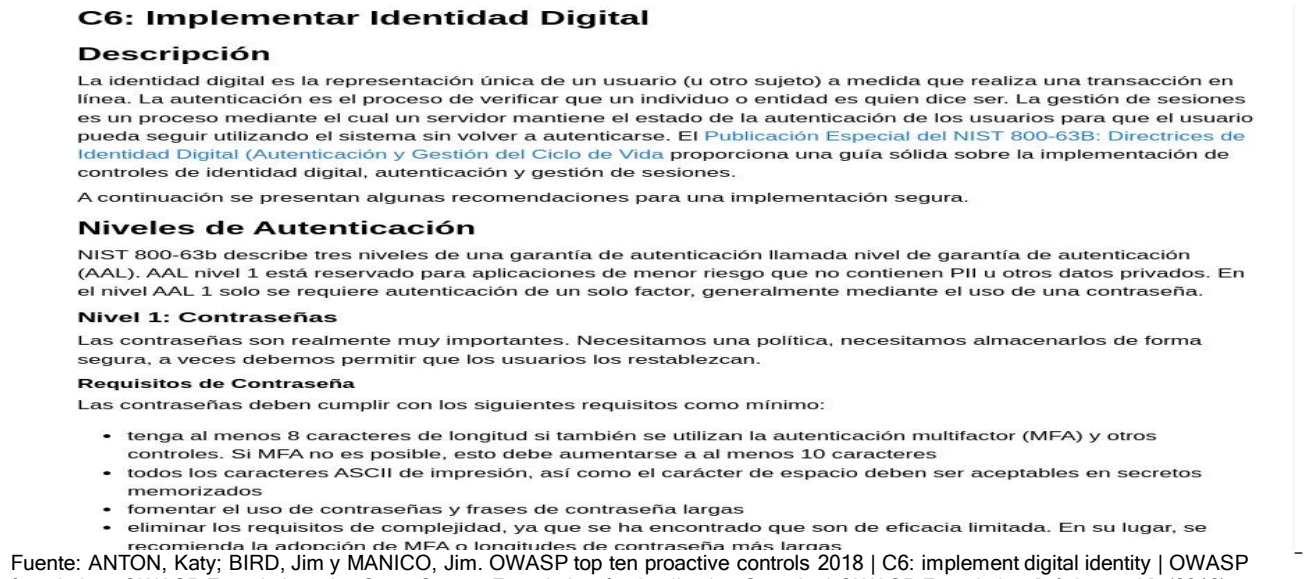
#	Descripción	L1	L2	L3	CWE	NIST §
2.1.1	Verifique que las contraseñas del usuarios tienen al menos 12 caracteres de longitud (después de combinar varios espacios). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verifique que se permitan contraseñas de al menos 64 caracteres y que se denieguen contraseñas de más de 128 caracteres. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verifique que no se realiza el truncamiento de contraseña. Sin embargo, varios espacios consecutivos pueden ser reemplazados por un solo espacio. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verifique que cualquier carácter Unicode imprimible, incluidos los caracteres neutros del idioma, como espacios y Emojis esté permitido en las contraseñas.	✓	✓	✓	521	5.1.1.2
2.1.5	Verifique que los usuarios pueden cambiar su contraseña.	✓	✓	✓	620	5.1.1.2
2.1.6	Verifique que la funcionalidad de cambio de contraseña requiere la contraseña actual y nueva del usuario.	✓	✓	✓	620	5.1.1.2

Fuente: GROSSMAN, Josh C. OWASP application security verification standard | OWASP foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [página web]. (Octubre, 2021). [Consultado el 28, marzo, 2023]. Disponible en Internet: <https://owasp.org/www-project-application-security-verification-standard/>.

[50] OWASP. OWASP Application Security Verification Standard. [Sitio WEB]. La entidad. [consultado el 29 de Enero de 2023]. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-es.pdf

El ASVS es el “QUE” deberíamos chequear, aporta el problema, pero no la solución específica; la mayor parte de sus requisitos relaciona una CWE, sección NIST y proyecto OWASP que recomienda el uso de una herramienta, metodología o buena práctica para aplicar, probar o depurar el requisito que se deba cumplir, estos serían “el CÓMO”.
En este caso, para la categoría de **V2.1 Seguridad de Contraseña**, el requisito *2.1.1* tiene relacionado la categoría *C6 (Implementar Identidad Digital)* del proyecto *OWASP Proactive Controls* y este relaciona la sección del NIST 800-63B (*Directrices de identidad digital)*, proporcionando una guía sólida para la implementación de este requisito de seguridad; por ejemplo, específica para cada nivel de garantía (del NIST), las recomendaciones de implementación del control y al mismo tiempo enlaza con la sección del proyecto *OWASP Cheat sheet series* que corresponda (*authentication*). Todo esto para que se tenga un conocimiento profundo para implementar lo necesario y cumplir con los requisitos de la categoría **V2.1 Seguridad de Contraseña**.

Figura 36. Sección C6 Implementar Identidad Digital proyecto OWASP Proactive Controls



C6: Implementar Identidad Digital

Descripción

La identidad digital es la representación única de un usuario (u otro sujeto) a medida que realiza una transacción en línea. La autenticación es el proceso de verificar que un individuo o entidad es quien dice ser. La gestión de sesiones es un proceso mediante el cual un servidor mantiene el estado de la autenticación de los usuarios para que el usuario pueda seguir utilizando el sistema sin volver a autenticarse. El Publicación Especial del NIST 800-63B: Directrices de Identidad Digital (Autenticación y Gestión del Ciclo de Vida proporciona una guía sólida sobre la implementación de controles de identidad digital, autenticación y gestión de sesiones.

A continuación se presentan algunas recomendaciones para una implementación segura.

Niveles de Autenticación

NIST 800-63b describe tres niveles de una garantía de autenticación llamada nivel de garantía de autenticación (AAL). AAL nivel 1 está reservado para aplicaciones de menor riesgo que no contienen PII u otros datos privados. En el nivel AAL 1 solo se requiere autenticación de un solo factor, generalmente mediante el uso de una contraseña.

Nivel 1: Contraseñas

Las contraseñas son realmente muy importantes. Necesitamos una política, necesitamos almacenarlos de forma segura, a veces debemos permitir que los usuarios los restablezcan.

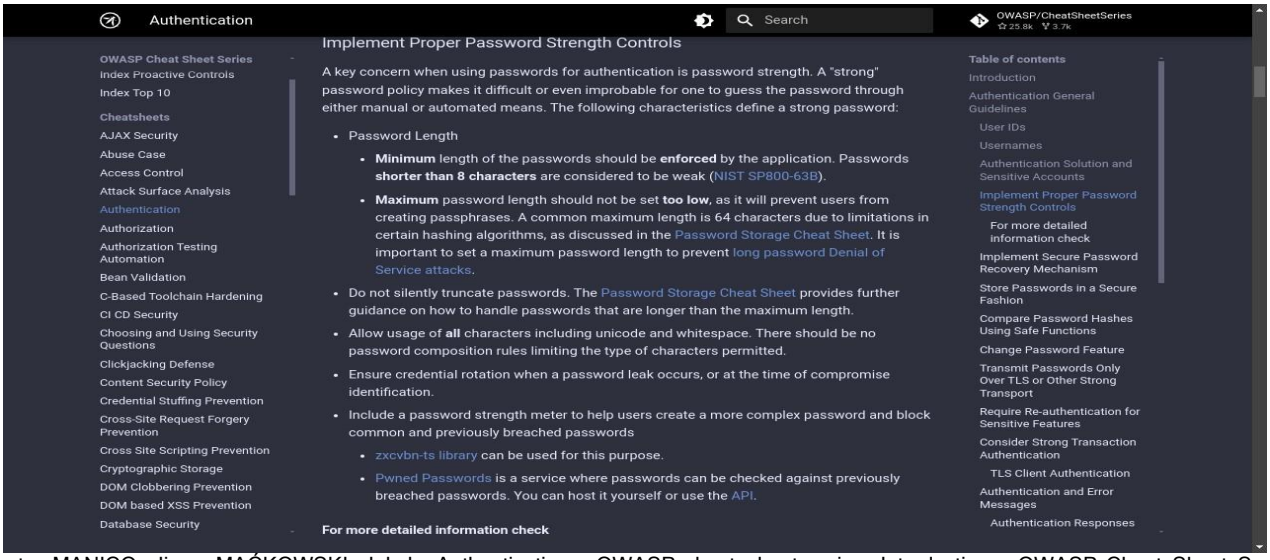
Requisitos de Contraseña

Las contraseñas deben cumplir con los siguientes requisitos como mínimo:

- tenga al menos 8 caracteres de longitud si también se utilizan la autenticación multifactor (MFA) y otros controles. Si MFA no es posible, esto debe aumentarse a al menos 10 caracteres
- todos los caracteres ASCII de impresión, así como el carácter de espacio deben ser aceptables en secretos memorizados
- fomentar el uso de contraseñas y frases de contraseña largas
- eliminar los requisitos de complejidad, ya que se ha encontrado que son de eficacia limitada. En su lugar, se recomienda la adopción de MFA o longitudes de contraseña más largas

Fuente: ANTON, Katy; BIRD, Jim y MANICO, Jim. OWASP top ten proactive controls 2018 | C6: implement digital identity | OWASP foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [página web]. (2018). [Consultado el 26, marzo, 2023]. Disponible en Internet: <https://owasp.org/www-project-proactive-controls/v3/en/c6-digital-identity>.

Figura 37. Sección *authentication* proyecto OWASP *Cheat sheet series*



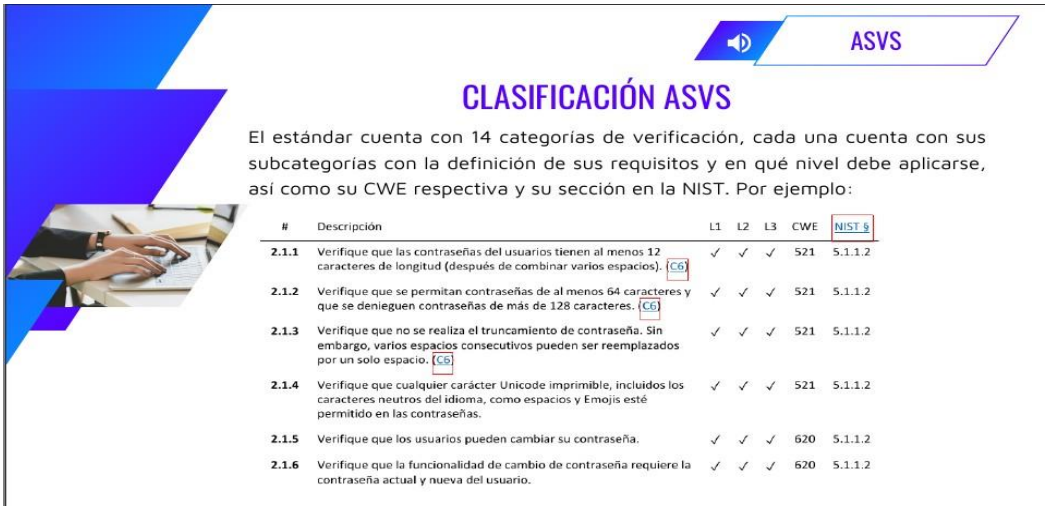
Fuente: MANICO, Jim y MAĆKOWSKI, Jakub. Authentication - OWASP cheat sheet series. Introduction - OWASP Cheat Sheet Series [página web]. (2019). [Consultado el 28, marzo, 2023]. Disponible en Internet: <https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#implement-proper-password-strength-controls>.

6.3.7 Integrando ASVS con guía ágil SAMM.

El primer paquete de trabajo fue la capacitación del estándar ASVS al equipo de trabajo, así mismo se realizó inducción de los proyectos de OWASP relacionados:

- ✓ Proactive Controls.
- ✓ Cheat Sheet Series.
- ✓ Top 10 risk.
- ✓ Security Knowledge Framework (SKF)

Figura 38. Material de Capacitación ASVS



Fuente: elaboración propia.

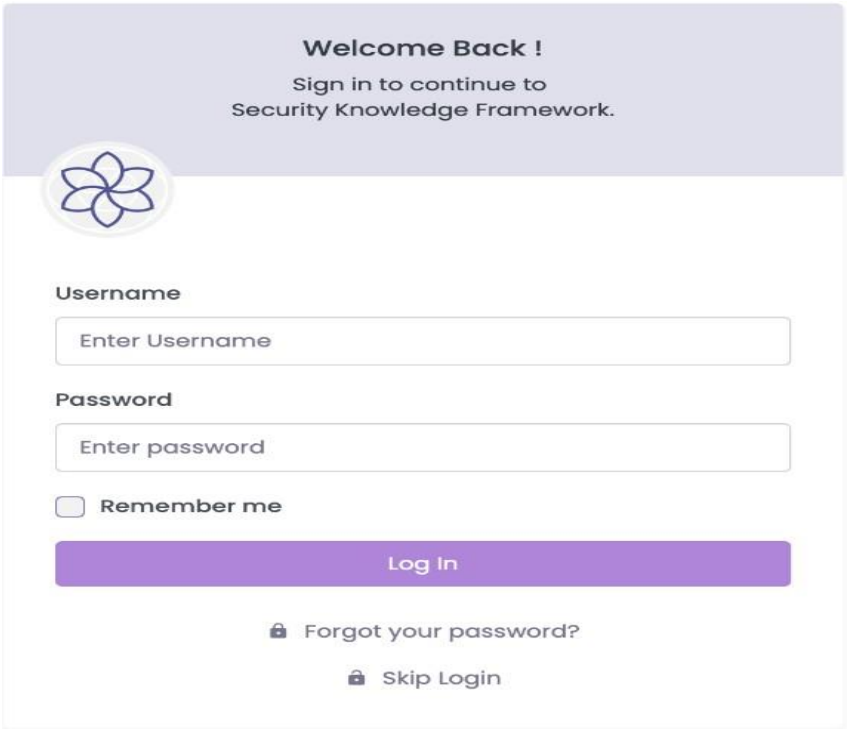
Hay que recordar que la guía ágil SAMM recomienda utilizar una lista de requisitos de seguridad ligada a las historias de usuario y para su uso ágil, se maneje como un activador según el entorno y el tipo de desarrollo; también que se puedan asociar a una métrica para medir un indicador de resultados.

Para esto, la herramienta que ofrece el OWASP es el SKF, según Vandan Verma, "El OWASP Security Knowledge Framework es increíblemente relevante para la seguridad actual de las aplicaciones y debe ser requerido en cualquier organización para capacitar a desarrolladores, investigadores de seguridad e incluso reunir requisitos"[51].

Ofrece a los desarrolladores ejemplos de lo que deben hacer, la posibilidad de crear sus proyectos con listas de verificación (contiene ASVS) ejemplos de código y un laboratorio para hacer pruebas de vulnerabilidades.

Para el segundo componente se instaló localmente la Herramienta SFK que permitió el control de los requisitos de seguridad de ASVS con su Check list para cada una de las historias de usuario.

Figura 39. Herramienta SFK



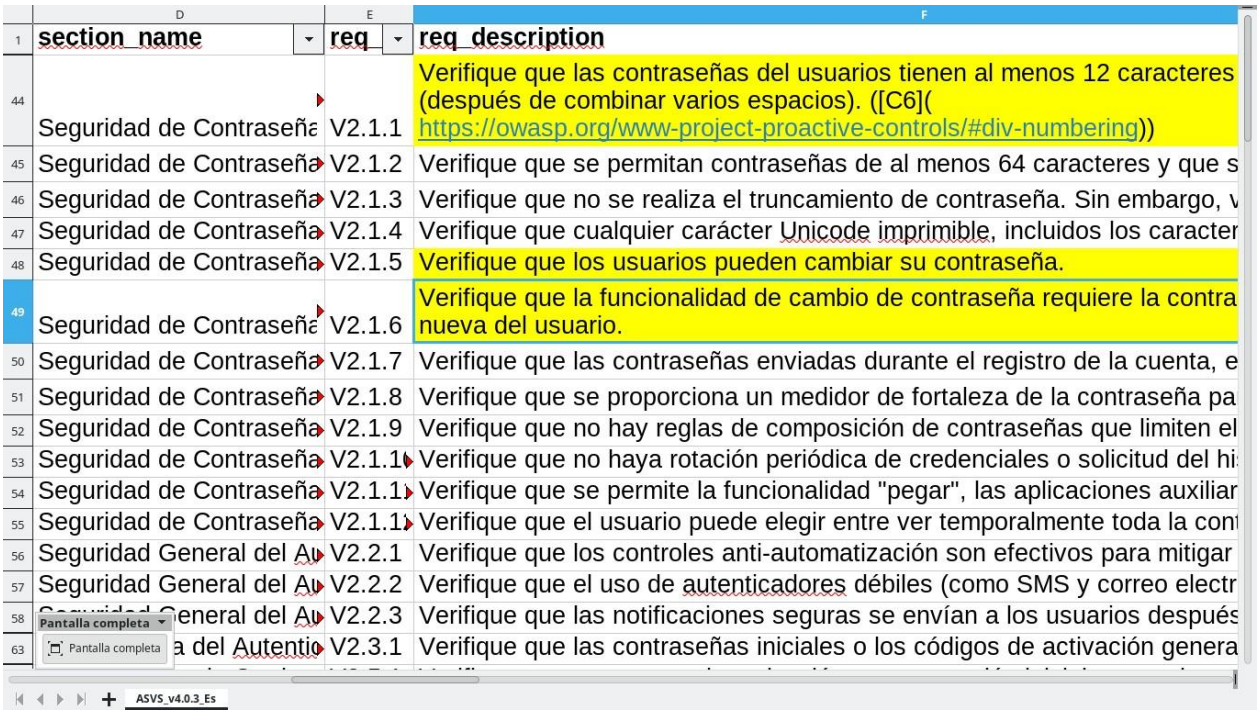
Fuente: Software SFK.

En este punto, se definió asignar los requisitos de seguridad como lo recomienda la guía rápida de SAMM, en cada Historia de usuario iniciar con pocos requisitos e ir incrementando

[51] OWASP. Security Knowledge Framework. [Sitio WEB]. Vandana Verma. [consultado el 20 de agosto de 2023]. Disponible en: https://owasp.org/projects/spotlight/historical/2021.02.03/

según sea el caso. Por ejemplo, para la validación de las contraseñas se inició con 3 requisitos básicos de seguridad para esta sección.

Figura 40. Requisitos ASVS Historia de usuario



	D	E	F
1	section_name	req	req_description
44	Seguridad de Contraseña	V2.1.1	Verifique que las contraseñas del usuarios tienen al menos 12 caracteres (después de combinar varios espacios). ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering))
45	Seguridad de Contraseña	V2.1.2	Verifique que se permitan contraseñas de al menos 64 caracteres y que s
46	Seguridad de Contraseña	V2.1.3	Verifique que no se realiza el truncamiento de contraseña. Sin embargo, v
47	Seguridad de Contraseña	V2.1.4	Verifique que cualquier carácter Unicode imprimible, incluidos los caracter
48	Seguridad de Contraseña	V2.1.5	Verifique que los usuarios pueden cambiar su contraseña.
49	Seguridad de Contraseña	V2.1.6	Verifique que la funcionalidad de cambio de contraseña requiere la contra nueva del usuario.
50	Seguridad de Contraseña	V2.1.7	Verifique que las contraseñas enviadas durante el registro de la cuenta, e
51	Seguridad de Contraseña	V2.1.8	Verifique que se proporciona un medidor de fortaleza de la contraseña pa
52	Seguridad de Contraseña	V2.1.9	Verifique que no hay reglas de composición de contraseñas que limiten el
53	Seguridad de Contraseña	V2.1.1	Verifique que no haya rotación periódica de credenciales o solicitud del hi
54	Seguridad de Contraseña	V2.1.1	Verifique que se permite la funcionalidad "pegar", las aplicaciones auxiliar
55	Seguridad de Contraseña	V2.1.1	Verifique que el usuario puede elegir entre ver temporalmente toda la cont
56	Seguridad General del Au	V2.2.1	Verifique que los controles anti-automatización son efectivos para mitigar
57	Seguridad General del Au	V2.2.2	Verifique que el uso de autenticadores débiles (como SMS y correo electr
58	Pantalla completa / eneral del Au	V2.2.3	Verifique que las notificaciones seguras se envían a los usuarios después
63	Pantalla completa / a del Autentic	V2.3.1	Verifique que las contraseñas iniciales o los códigos de activación genera

ASVS_v4.0.3_Es

Fuente: elaboración propia

En el tercer componente se definió el proceso para el registro de requisitos de seguridad en la definición de terminado (DoD) y en criterios de aceptación de las historias de usuarios de los nuevos desarrollos.

Figura 41. Propuesta de HU ASVS Ágil_1

NOMBRE PROYECTO:	ENDO			
TITULO REQUERIMIENTO:	CARGAR ARCHIVOS EN EL MODULO DE PROGRAMACIÓN DE QX.			
RESUMEN:	ADJUNTAR ARCHIVOS EN FORMATO PDF EN EL PROCESO DE PROGRAMACIÓN DE CIRUGÍA.			
PRIORIDAD	ALTA			
Horas Sec	(Horas desarrollo controles de seguridad)			
TIPO REQUERIMIENTO:	**NUEVA CAMPO**	**X**	**SOLUCIÓN ERROR**	
DEFINICIÓN DE TERMINADO. DOD: Requisitos que deben cumplir todas las historias de usuario del sprint.	Casos de uso.			
	Controles de seguridad según entorno y modelado. (Autenticación, validación)			
	Actualización de modelado de amenazas.			
	Visto bueno de Product Owner .			
	Visto bueno del cliente.			
	Requisitos de seguridad ASVS del capítulo V12 Archivos y Recursos.			
DESCRIPCIÓN HISTORIA DE USUARIO (HU): • **Como** usuarios con el perfil de programación de cirugía. • **Quiero:** adjuntar archivos en formato pdf en el proceso de programación de cirugía. • **Para:** consultar (ver) y eliminar archivos PDF del paciente en el modulo de Programación de cirugía.				
CRITERIOS DE ACEPTACIÓN:				
Escenario 1 función solicitada:	**Dado que**: he ingresado con el perfil de programación de cirugía. **Cuando**: estoy en la opción PROGRAMACIONES CIRUGÍAS, en el bloque INSUMOS Y MATERIAL DE OSTEOSÍNTESIS REQUERIDOS. **Entonces**: debe permitir adjuntar, ver y eliminar archivos .pdf 1. La funcionalidad se debe activar por medio de una variable de modulo. 2. Permitir adjuntar, visualizar y eliminar archivos .PDF. 3. Los archivos se deben guardar en el servidor cloud.			

Fuente: elaboración propia.

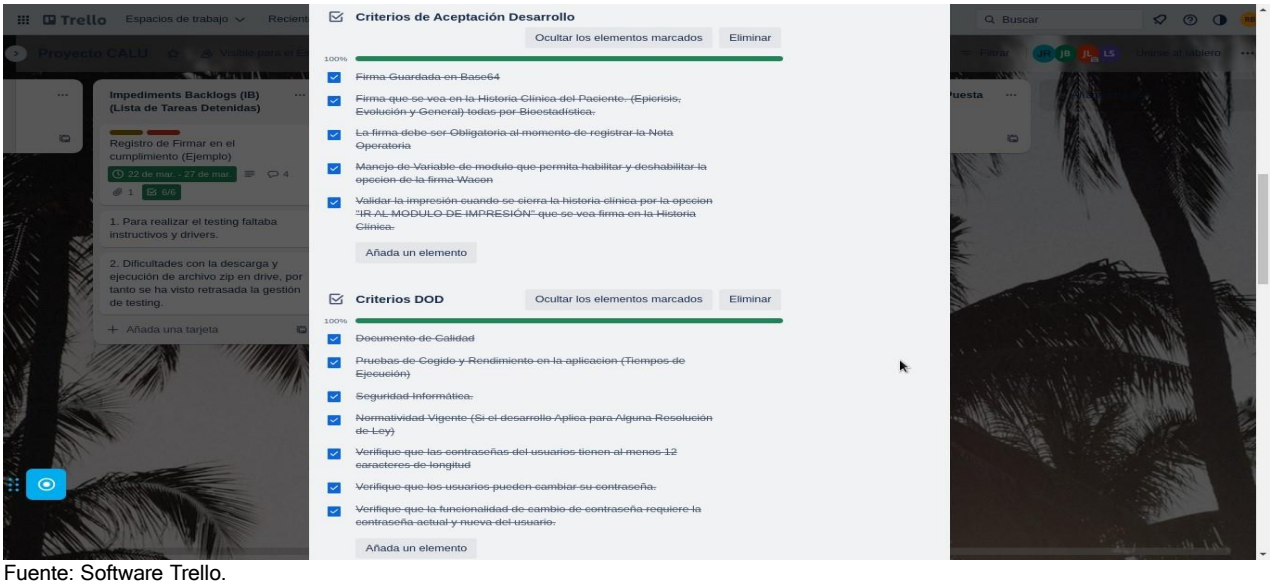
Figura 42. Propuesta de HU ASVS Ágil_2

DESCRIPCIÓN HISTORIA DE USUARIO (HU) SEGURIDAD:		
Horas: 5. **COMO** usuario de la aplicación, **QUIERO** que se definan restricciones de los archivos que se puedan cargar a la aplicación, **PARA** que esta no se quede sin recursos para procesar los archivos cargados.		
CRITERIOS DE ACEPTACIÓN:		
Escenario 1: No Acepte archivos grandes.	**Requisito ASVS :** <v4.0.3-12.1.1>. Implemente/Verifique que la aplicación no aceptará archivos grandes que puedan llenar el almacenamiento o provocar una denegación de servicio. **DADO** que la aplicación tiene la opción para cargar archivos **Y** se define un tamaño máximo del archivo, **CUANDO** un archivo de tamaño superior al tamaño máximo definido se carga en la aplicación **ENTONCES** falla la carga del archivo **Y** se muestra un mensaje de error para esa acción.	
Escenario 2: Restricción para el número de archivos por usuario	**Requisito ASVS : <v4.0.3-12.1.3>** Implemente/Verifique que se aplica una cuota de tamaño de archivo y un número máximo de archivos por usuario para asegurarse de que un solo usuario no puede llenar el almacenamiento con demasiados archivos o archivos excesivamente grandes. **DADO** que la aplicación tiene la opción para cargar archivos **Y** se define un tamaño máximo del archivo **Y** se define una cantidad máxima de archivos por usuario. **CUANDO** un usuario carga un archivo de valor superior al tamaño máximo o cantidad máxima definido, **ENTONCES** falla la carga del archivo **Y** se muestra un mensaje de error para esa acción.	
Aprobado	**Nombre:** **Cargo:**	**Firma:**

Fuente: elaboración propia.

Este formato se traslada a la definición de las HU en el software de gestión del Sprint.

Figura 43. Requisitos de seguridad en DoD



Fuente: Software Trello.

8. Web Security Testing Guide -v4.2.

1. Introducción.

En seguida analizaremos el OWASP WEB SECURITY TESTING GUIDE V4.2 (WSTG). En esta guía tenemos un framework de pruebas que describe técnicas y actividades para incluir en cada una de las fases del SDLC, ayuda a tener un conocimiento profundo de qué, por qué, cuándo, dónde y cómo probar aplicaciones web, es decir, que adicional al "COMO" hacerlo también tenemos el "CUANDO". Le permitirá a esta casa de desarrollo que su HIS principal que es 100% web sea segura y confiable.
Tiene como premisa realizar pruebas a 3 componentes, el "QUË":

Personas. Para garantizar que hay una educación y concienciación adecuada.
Procesos. Con el fin de garantizar que existen políticas y estándares adecuados y las personas saben seguirlas.
Tecnología. Con el objetivo de asegurar que el proceso es efectivo en su implementación.

Un concepto importante, "**Pruebas:** es un proceso de comparar el estado de un sistema o aplicación con un conjunto de criterios"[52].

Para tener un control y seguimiento de las pruebas se deben identificar con el formato WSTG<versión> <categoría><número>.

Un ejemplo:
Test de credenciales predeterminadas.
ID*:* ***WSTG-ATHN-02***
La guía da una recomendación de como probar, "*Busque archivos de configuración que contengan nombres de usuario y contraseñas. Examine la política de contraseñas y, si la aplicación genera sus propias contraseñas para nuevos usuarios, verifique la política en uso para este procedimiento*". Recomienda las herramientas[53]:

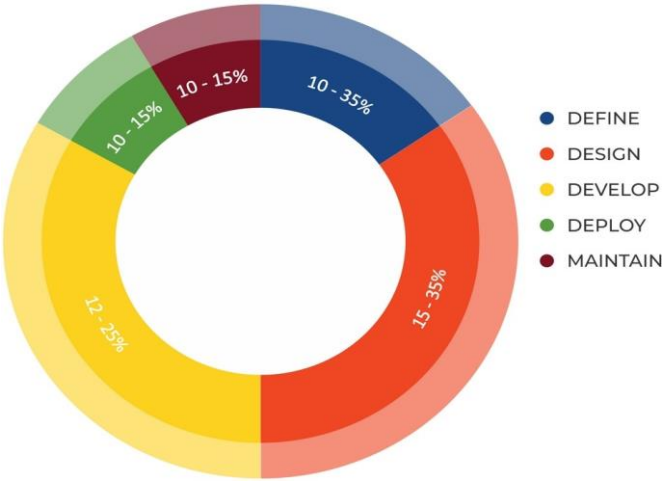
Burp Intruder
THC Hydra
Nikto 2

Describe las técnicas de prueba como, revisiones manuales, modelado de amenazas, revisión del código fuente y pruebas de penetración. Así mismo, recomienda un enfoque combinado de varias técnicas de prueba, pero sobre todo que se realicen en todas las fases del SDLC.

[52] OWASP. The OWASP Testing Project. [Sitio WEB]. Eoin Keary. [consultado el 30 de junio de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/

[53] OWASP. WSTG-v4.2. [Sitio WEB]. La entidad. [consultado el 29 de Enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

Figura 44. Proporción de esfuerzo de pruebas en SDLC



Fuente: OWASP. The OWASP Testing Project [imagen]. [Consultado el 30, junio, 2023]. Disponible en Internet: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/

Los objetivos de las pruebas están relacionados con la definición específica de los requisitos de seguridad, por eso es necesario:

- ✓ La documentación de requisitos de seguridad: requisitos comerciales, verificación de estándares, regulaciones y políticas.
- ✓ La validación de requisitos de seguridad: Modelado de amenazas, revisión de código seguro y pruebas de penetración.
- ✓ Requisitos de prueba funcionales y no funcionales.
- ✓ Requisitos de seguridad basados en riesgos.
- ✓ Requisitos de seguridad a través de casos de uso y mal uso.

Hace énfasis en la clasificación de las pruebas:

Pruebas de seguridad en el flujo de trabajo de desarrollo. Afectan los componentes de software individuales que han desarrollado antes de integrarlos con otros componentes, donde se realiza análisis del código fuente, dinámico y estático (pruebas unitarias).

Pruebas de seguridad en el flujo de trabajo de prueba.
Pruebas a la aplicación como entidad completa (prueba integrada). Incluyen pruebas de caja blanca, como el análisis del código fuente, y pruebas de caja negra, como las pruebas de penetración y de caja gris.

Pruebas de seguridad del desarrollador.

Pruebas de seguridad en la fase de codificación: pruebas unitarias.

Los requisitos de seguridad que deben seguir los desarrolladores deben documentarse en estándares de codificación segura y validarse con análisis estáticos y dinámicos.

Los desarrolladores que cuentan con:

- ✓ Una herramienta Análisis de código fuente integrada en su IDE,
- ✓ Estándares de codificación segura y
- ✓ Un marco de pruebas unitarias de seguridad como JUnit, NUnit y CUnit

Les resulta más fácil evaluar y verificar los requisitos de las pruebas de seguridad en los componentes que están desarrollando.

Pruebas de seguridad de los probadores funcionales.

Pruebas de seguridad durante la fase de INTEGRACIÓN y VALIDACIÓN: pruebas del sistema integrado y pruebas de Operación (funcionamiento). En una prueba de seguridad más profunda, el evaluador puede requerir técnicas y herramientas mucho más especializadas

Por ejemplo:

- ✓ Código fuente e inyección de fallas binarias,
- ✓ Análisis de propagación de fallas y Cobertura de código,
- ✓ Pruebas fuzz e
- ✓ Ingeniería inversa.

Los informes. Se realizan a partir de los objetivos para las métricas y mediciones de las pruebas de seguridad que permiten la gestión y análisis de los riesgos. Estos requieren la siguiente información:

- ✓ **Categorización** de cada vulnerabilidad por tipo.
- ✓ **La amenaza** de seguridad a la que está expuesto cada problema.
- ✓ **Causa raíz** de cada problema de seguridad, como el error o la falla.
- ✓ **Técnica de prueba** utilizada para encontrar los problemas.
- ✓ **Contramedida** para cada vulnerabilidad.
- ✓ **Clasificación de riesgo** de cada vulnerabilidad (por ejemplo, puntuación alta, media, baja o CVSS).

2. El framework de testing de OWASP.

El framework ayuda a las organizaciones a crear un proceso de pruebas estratégico completo, que no está orientado a consultores o contratista, es definido por fases:

Fase 1 antes que inicie el Desarrollo.

- ✓ **Definir un DSLC.**
- ✓ **Revisión de políticas y estándares**. Asegúrese de que existan políticas, estándares y documentación adecuados.
- ✓ **Desarrollar criterios de medición y métricas y garantizar la trazabilidad.** Si se define los criterios que deben medirse, va a tener visibilidad de los defectos tanto en el proceso como en el producto.

Fase 2 Durante la Definición y el Diseño.

- ✓ **Revisar los requisitos de seguridad.** Los requisitos de seguridad definen cómo debe funcionar una aplicación desde el punto de vista de seguridad.
- ✓ **Revisión del diseño y la arquitectura.** Debe estar documentado el diseño y la arquitectura.
- ✓ **Crear y revisar modelos UML.** Se utilizan para confirmar con los diseñadores de sistemas una comprensión exacta de cómo funciona la aplicación.
- ✓ **Crear y revisar modelos de amenazas.** Realice un ejercicio de modelado de amenazas y cree escenarios de amenazas realistas.

Fase 3 Durante el DESARROLLO.

- ✓ Aunque el Desarrollo es la codificación de un diseño:
  - ➢ Si el diseño y la arquitectura no es adecuado, el desarrollador se tendrá que tomar muchas decisiones.
  - ➢ Si no hay políticas y estándares suficientes, el desarrollador tendrá que tomar aún más decisiones.
- ✓ **Tutorial del código.** Su propósito no es realizar una revisión de código, sino comprender a alto nivel el flujo, el diseño y la estructura del código que conforma la aplicación.
- ✓ **Revisiones de código**. El tester puede examinar el código real en busca de defectos de seguridad. La revisión de código estático produce un retorno de calidad mucho más altos que cualquier otro método de revisión de seguridad y dependen menos de la habilidad del revisor.

Fase 4 Durante la Implementación.

- ✓ **Testing de penetración de aplicaciones.** Suministra una verificación adicional para garantizar que no se haya pasado nada por alto.

- ✓ **Prueba de gestión de la configuración.** Hay que revisar todas todos los elementos de configuración para que no queden con una configuración predeterminada y puedan ser vulnerable a explotación.

Fase 5 Durante el Mantenimiento y las Operaciones.

- ✓ **Realizar revisiones de la gestión operativa.** Se debe implementar un proceso que detalle cómo se gestiona el componente operativo de la aplicación e infraestructura.
- ✓ **Realizar controles de salud periódicos.** Realizar controles periódicos del estado de la aplicación y la infraestructura para garantizar que no hay nuevos riesgos de seguridad y se tiene el mismo nivel de seguridad.
- ✓ **Garantizar la verificación de cambios.** Cada vez que se aprueba un cambio en cada entorno, calidad, producción, se debe verificar el cambio para que el nivel de seguridad no haya sido afectado e integrarlo al proceso de gestión de cambios.

En la sección 4 de la guía se describen, en la práctica y en detalle, las pruebas de seguridad de aplicaciones web, de igual manera, explica como probar la evidencia de las vulnerabilidades.
Está dividida en 12 categorías, que se enfocan en pruebas de caja negra, donde interactúa, el tester, herramienta y tecnología y la aplicación. Las pruebas se clasifican en pruebas pasivas y activas:

1. Recopilación de información.
2. Pruebas de gestión de configuración e implementación.
3. Pruebas de gestión de identidad.
4. Pruebas de autenticación.
5. Prueba de autorización.
6. Pruebas de gestión de sesiones.
7. Pruebas de validación de entrada.
8. Manejo de errores.
9. Criptografía.
10. Pruebas de lógica empresarial.
11. Pruebas del lado del cliente.
12. Pruebas API.

6.3.9 Integrando WSTG a SAMM Ágil y ASVS.

Se utilizará esta guía adaptada al proceso de desarrollo de la empresa SAC, este contará con las técnicas, procedimientos y recomendaciones propias de este marco en cada una de las fases de SDLC asociadas a los requisitos de seguridad de la ASVS definidas en cada una de las Historias de usuario. Es posible que debamos crear una HU para requisitos de seguridad específicos según la recomendación de la guía, así mismo se irán adicionando en cada iteración como se definió para la ASVS.

Se inicia con la primera actividad de este componente, el paquete de trabajo de capacitación que tiene como objetivo reconocer la Guía de Pruebas de Seguridad Web de OWASP.

Figura 45. Material de capacitación WSTG-v4.2

Fuente: elaboración propia.

Recordemos que el objetivo definido en la FASE I del modelo SAMM, incluye los controles de seguridad del nivel L1 del ASVS, donde casi todos se pueden probar con test de penetración. Por esta razón, la principal herramienta utilizada por SAC para este fin es OWASP ZAP y en una menor medida, solo para esta fase, revisiones manuales.

Siguiendo con el pensamiento ágil (SAMM Ágil) y el principio de la seguridad hacia la izquierda, el enfoque adoptado por SAC estuvo en las fases del SDLC, Gobernanza (Definición, planeación), Diseño y Verificación (Pruebas).

6.3.9.1 Antes de Desarrollo (Gobernanza).

Ya se ha definido el modelo S-SDLC ágil, apoyado del framework Scrum y su base principal de seguridad es el SAMM V2.

En SAC, no se tiene políticas específicas de su SDLC, lo cual no les permite apoyarse en estas, pero se vienen estableciendo las políticas inherentes recomendadas de los marcos de trabajo adoptados en este proyecto.

2. Durante la Definición y el Diseño.

Los objetivos de las pruebas están orientados a los controles de seguridad definidos en la HU, los cuales vienen del entorno del desarrollo, la necesidad de mitigación de amenazas del modelado de amenazas y/o de los controles de seguridad del ASVS (para el nivel L1), es decir, se definieron desde la perspectiva del riesgo.

Igualmente, el enfoque es probar los controles de seguridad orientados a:

- ✓ Autenticación.
- ✓ Validación, desinfección, y Codificación.
- ✓ Protección de datos.
- ✓ Archivos y recursos.
- ✓ API y servicios web.

En cada HU se adicionó un escenario de amenaza detallando la función que debería realizar según los controles del ASVS para mitigar la amenaza.

SAC no cuenta con documentación de la arquitectura, ni UML, pero a partir de la documentación de la funcionalidad de cada módulo (Requisitos del sistema, manuales y entregas a QA), entrevistas de personal, se creó el modelado de amenazas que se definió revisar en esta etapa.

3. Durante el desarrollo.

Aunque es claro que la revisión de código estático produce mejores resultados que cualquier otra técnica, se requiere el compromiso de más recursos, el cual el muy limitado en SAC. No se realizaron pruebas de este tipo, de acuerdo al objetivo de la Fase I (SAMM) y se define incluirlo en el objetivo de la Fase II.

4. Durante la Implementación.

Para la fase del S-SDLC, Verificación, las pruebas estuvieron basadas en requisitos de seguridad, como se comentó anteriormente, aunque la recomendación es incluir las pruebas manuales para una mejor eficiencia al evaluar vulnerabilidades que con el test automático no se podría “las pruebas de penetración enfocadas (es decir, pruebas que intentan explotar vulnerabilidades conocidas detectadas en revisiones anteriores) pueden ser útiles para detectar si algunas vulnerabilidades específicas realmente están solucionadas en el código fuente implementado”.

Conociendo las limitaciones de las pruebas de caja negra con herramientas automatizadas como ZAP, se crearon contextos y alcances específicos para probar los requisitos de

seguridad de la HU cada vez que se hacía una entrega de Desarrollo a la Fase de Verificación (Pruebas).

Entonces, se realiza las pruebas, las básicas o iniciales que se definieron para realizar en esta Fase I. Para este ejemplo:

- ✓ Autenticación.
- ✓ Validación, desinfección, y Codificación.
- ✓ Protección de datos.

Y las de entorno del desarrollo, es decir, para este ejemplo:

- ✓ Archivos y recursos.

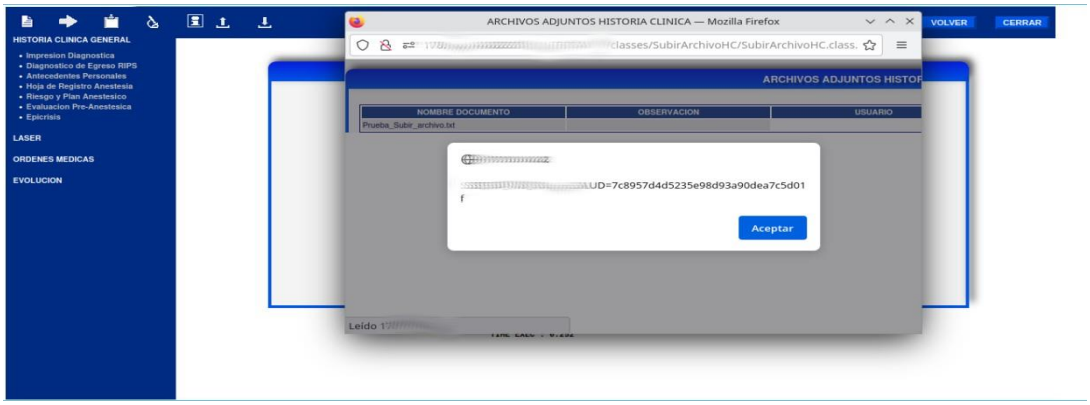
6.3.9.5 Prueba de Validación y carga de archivos.

Test de Cross Site Scripting (XSS) almacenados.
ID: WSTG-INPV-02
Las aplicaciones web permite a los usuarios almacenar datos que pueden ser maliciosos para su posterior uso. En este caso se realiza la prueba ingresando un script:

```
<script>alerta(document.cookie)</script>
```

La entrada se envía a través de la aplicación al modificar la solicitud HTTP con ZAP para ser reutilizado después, en este caso al recargar la opción de listar los archivos cargados. Para este ejemplo, la ventana emergente muestra la información de las cookies.

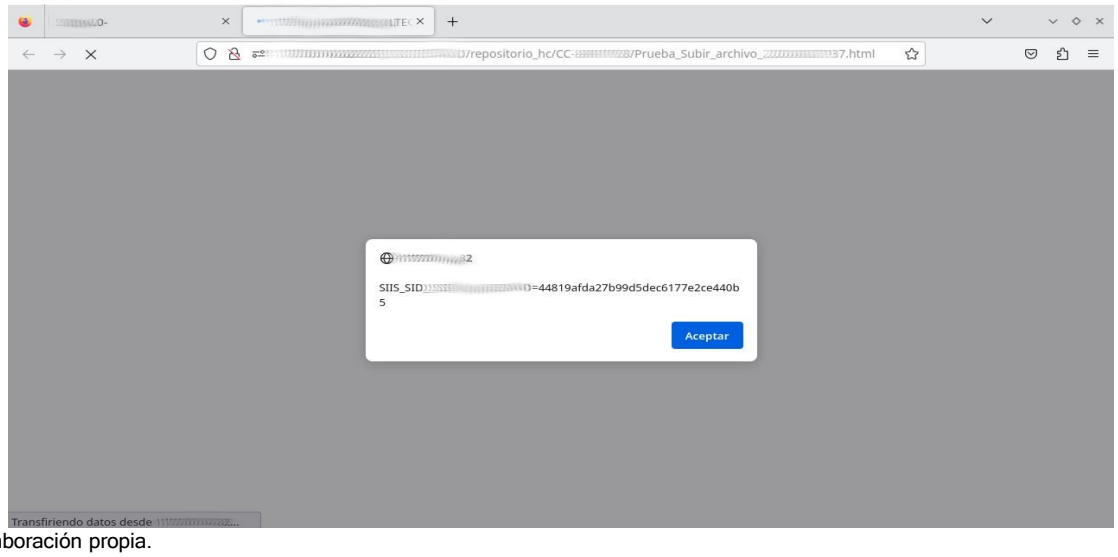
Figura 46. Test de XSS almacenado



Fuente: aplicación SAC.

Otra prueba realizada al subir el archivo, fue verificar si permite cargar archivos HTML, en estos se puede inyectar cargar XSS, Para este ejemplo al ingresar a la opción para ver el archivo, genera la ventana emergente con la información de las cookies, pues contiene el mismo script de la prueba anterior.

Figura 47. Inyectar carga XSS almacenado en archivo

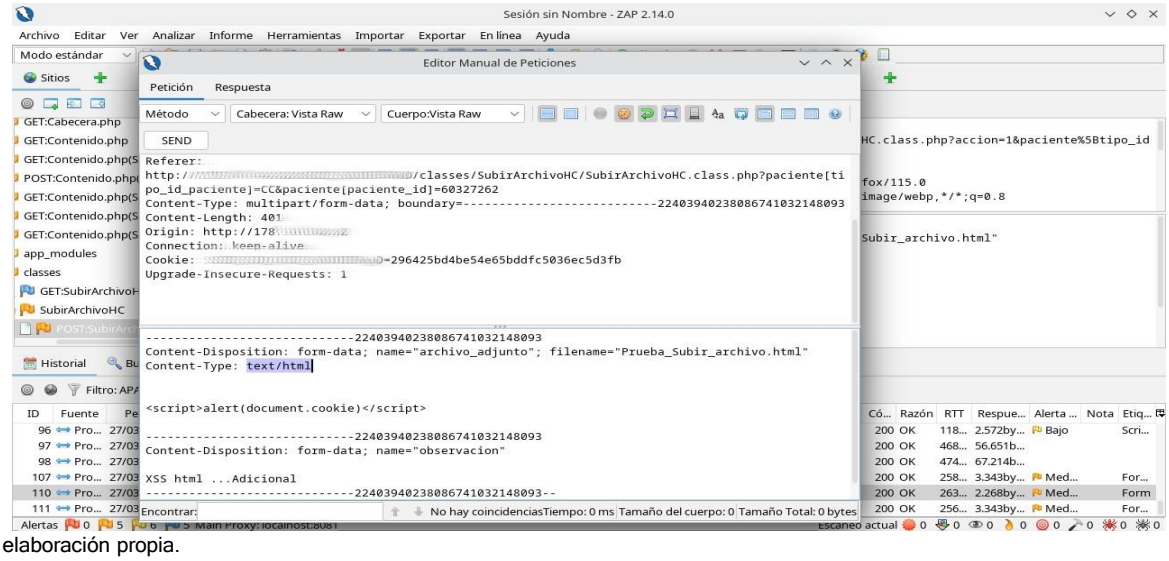


Fuente: elaboración propia.

De igual manera se realiza la prueba con ZAP modificando la entrada a través de su herramienta para modificar la solicitud.

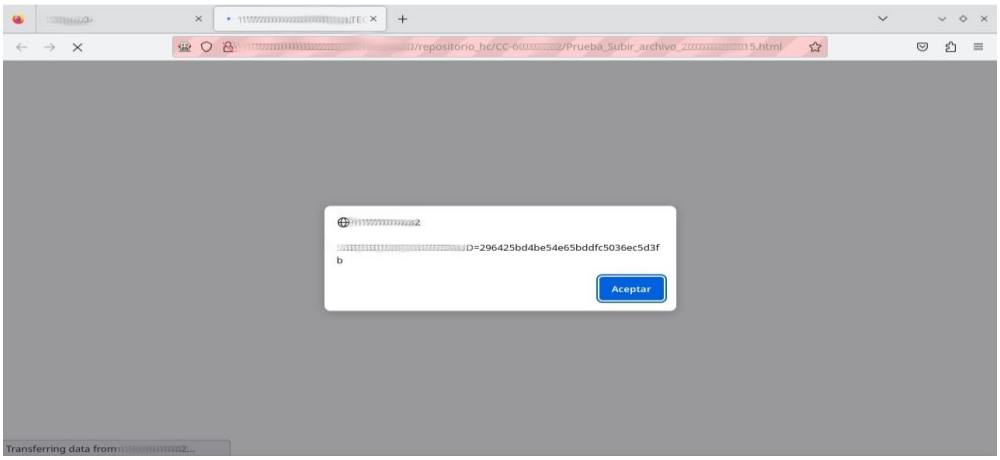
La imagen muestra la petición que fue falsificada para cargar el archivo html.

Figura 48. Solicitud falsificada



Fuente: elaboración propia.

Figura 49. XSS almacenado



Fuente: elaboración propia.

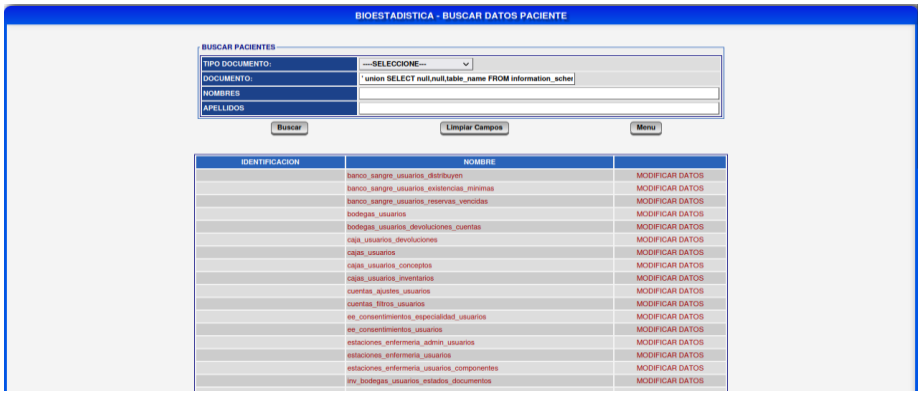
Test para PostgreSQL.
ID: WSTG-INPV-05

Ahora, para comprobar la validación de entradas(campos), se realiza un test de inyección SQL para uno de los campos del formulario que contiene los parámetros de búsqueda del paciente:

En este test, previamente se identifica el campo del formulario que permite “SQL Injection” y la cantidad de campos que genera el Query, y con una consulta SQL con UNION ejecutada en el campo vulnerable, permite identificar las tablas de la base de datos cuyo nombre contengan ‘usuario’:

```
' union SELECT null,null,table_name FROM information_schema.tables WHERE table_schema='public' AND table_type='BASE TABLE' AND table_name like '%usuario%'–
```

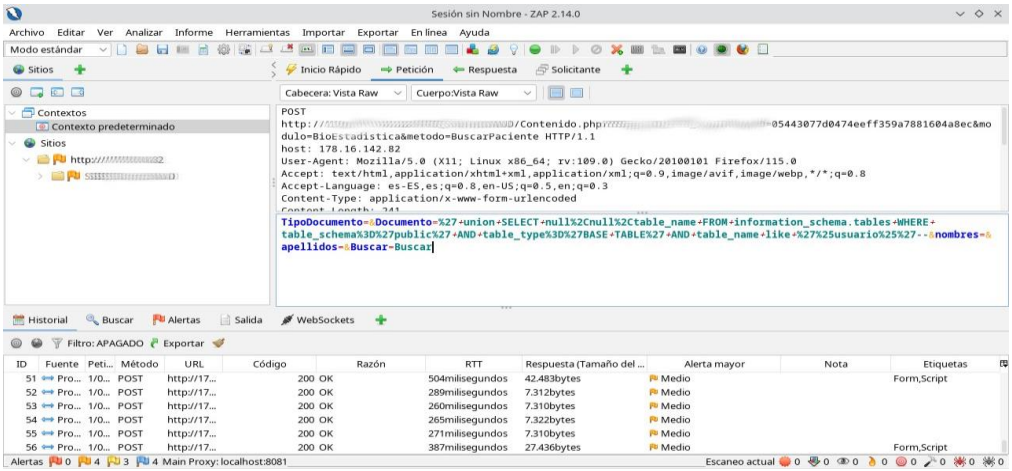
Figura 50. Inyección SQL manual



Fuente: elaboración propia.

Con ZAP también se ejecuta un contexto creado poder automatizar la prueba.

Figura 51. Inyección SQL con ZAP



Fuente: elaboración propia.

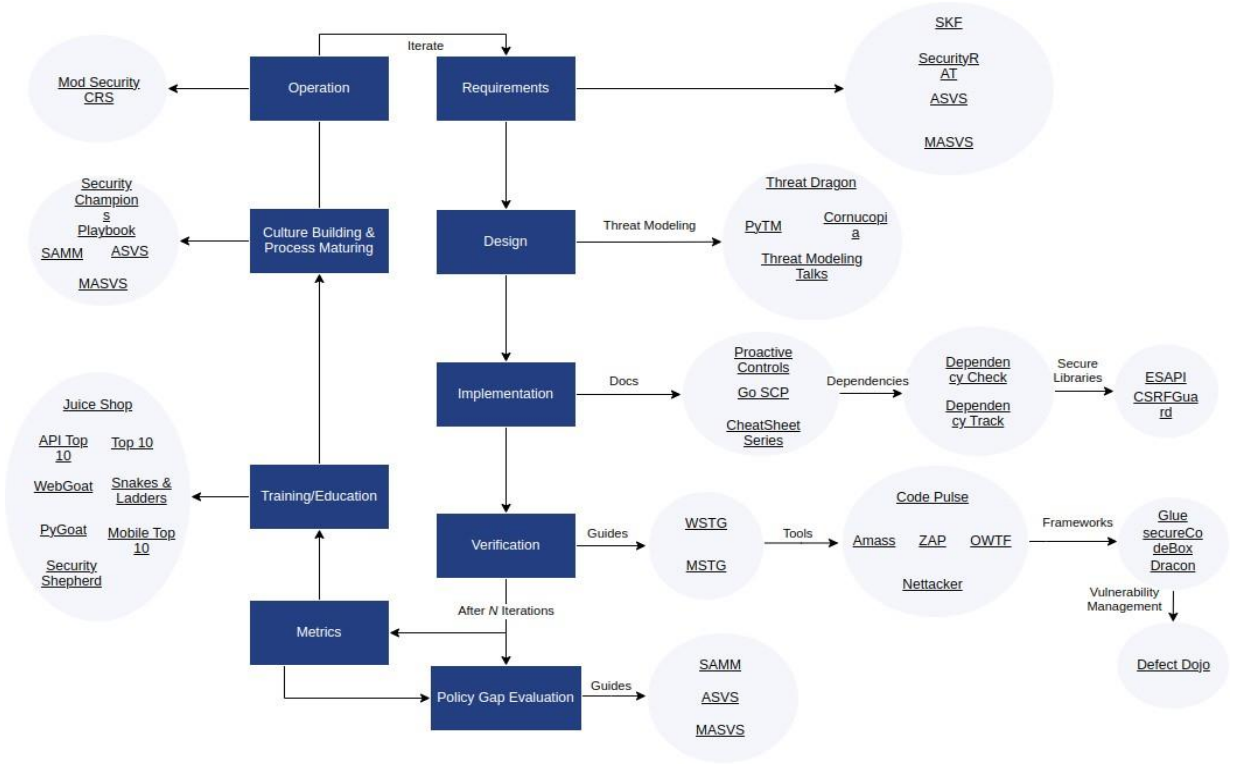
Aunque estas pruebas están orientadas específicamente a la HU, no podemos olvidar que se debe crear el plan de Testing general en la fase de Mantenimiento.

Es necesario realizar una imagen del estado actual de la evaluación de vulnerabilidades de la aplicación.

6.3.10 Ruta de S-SDLC con OWASP.

La fundación OWASP es una comunidad que ofrece una serie de proyectos para mejorar la seguridad de aplicaciones web, en la siguiente imagen muestra el diagrama que relaciona los diferentes proyectos para alcanzar este objetivo, son marcos de trabajo, guías, modelos de buenas prácticas y herramientas que se deben aplicar en cada una de las fases del ciclo de vida de desarrollo seguro según OWASP.

Figura 52. Rutas de aplicaciones de seguridad en S-SDLC



Fuente: OWASP. Projects | OWASP Foundation [imagen]. [Consultado el 10, marzo, 2023]. Disponible en Internet: <https://owasp.org/projects/>.

Para este caso se evaluarán los proyectos emblemáticos de OWASP que se ha comprobado aportan valor a la seguridad de aplicaciones web y que se adaptan más al SDLC de SAC.

Aunque proyectos nuevos como OWASP Devsecops Maturity Model, OWASP DevSecOps Guideline y el OWASP DevSecOps Verification Standard deben revisarse para integraciones futuras.

Para OWASP el SDLC cuenta con 6 fases, en las cuales ofrece una lista de herramientas y marcos gratuitos que se deben integrar para tener un SDLC seguro (S-SDLC)

1. Planificación – Análisis.

Durante la recolección de los requerimientos podemos definir cuál es la funcionalidad de la aplicación, la organización puede usar SAMM para asociar conceptos fuertes de seguridad en el inicio de S-SDLC, estos se pueden auditar con los check list de ASVS y relacionar el Sprint en SKF o gestionar automatizar estos requisitos de seguridad con SecurityRAT.

2. Diseño.

En esta fase se define el flujo de los datos y la arquitectura del sistema, insumos que permiten generar el modelado de datos de la épica. Continua ***SAMM*** y aprovechando ***SFK***, con ***TheartSpec*** y ***PyTM*** se pueden crear los modelados de amenazas. Así estarían en camino de tener diseño y consideraciones seguras.

3. Desarrollo (Implementación).

Esta etapa se aborda de acuerdo a algunas consideraciones propias de la organización, experiencia del equipo, cultura de ingeniería, madurez del marco de seguridad. En esta fase se utilizan los componentes de patrones y antipatrones de código y herramientas para realizar test de abusos de seguridad. Para asegurar código consistente se integra la corrección de código con algún **linters** que comprueben el código rápidamente. Con ***Cheatsheets*** ayuda a definir código de forma segura, asi mismo ***SFK*** con su extensa biblioteca de patrones de código. Para frontend ***ReactJS*** y bases de datos Query builders.

4. Pruebas e Integración.

Se validan las correcciones de la aplicación y sus resultados para tomar decisiones sobre fases anteriores, se realizan pruebas manuales y automáticas; con ***SAMM*** se revisa la arquitectura y los requisitos de seguridad. Se recomienda la automatización para identificar vulnerabilidades sin la interacción humana. Para este caso se puede utilizar el ***ZAP*** para automatizar los ataques a aplicaciones web, ofrece su API REST, una secuencia de comandos Zest que se pueden importar en la plataforma ***Defect Dojo***. Para las pruebas manuales, ***Web Security Testing Guides (WSTG).***

5. Lanzamiento.

En esta etapa las propiedades de la aplicación deben estar correctamente diseñadas, codificadas y probadas, está relacionada a la configuración y resistencia, ***Mod Security Core Rule Set*** detecta y bloquea ataques contra la plataforma. Con ***Open Policy Agent*** se debe aplicar estándares de configuración seguro, en cuanto a la observabilidad ***ELK stack, Grafana*** y ***Prometheus*** pueden ayudar con este objetivo. Por último, ***SAMM*** define una sección para manejo de incidentes.

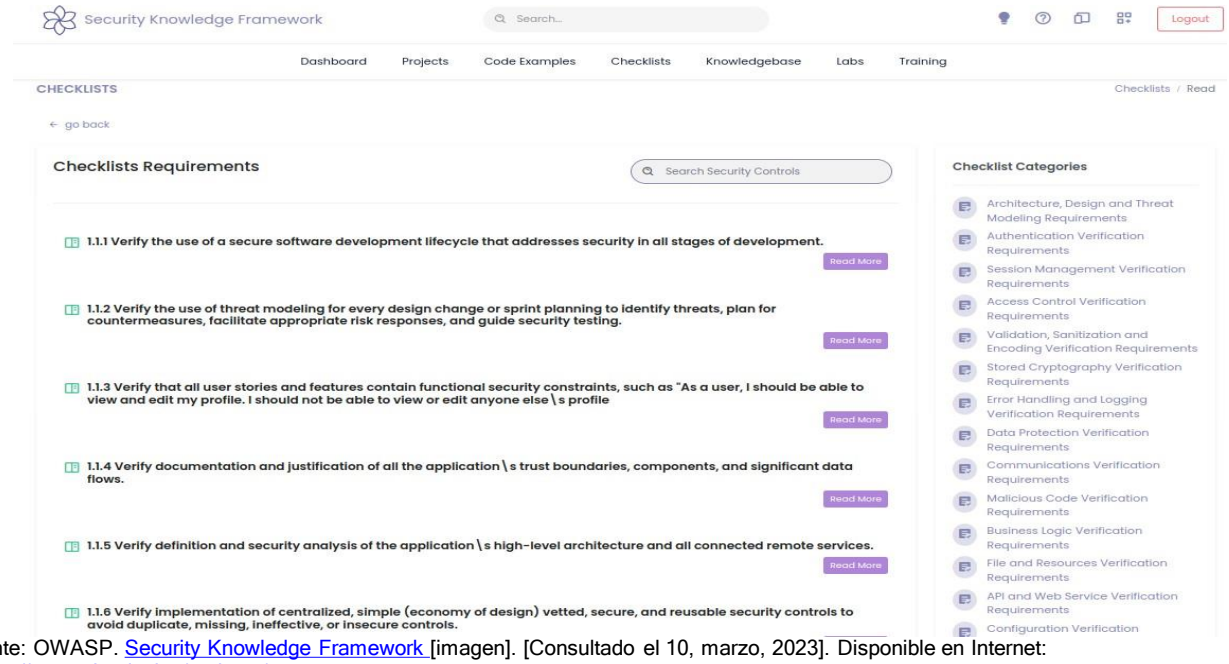
11. Fase Requerimientos:

1. Security Knowledge Framework (SKF).

Contiene información relacionada con seguridad, el desarrollo seguro y verificación de Seguridad. Uno de los proyectos más importantes, es el Security Knowledge Framework, útil para capacitarse, define unos requisitos para **crear aplicaciones seguras por diseño** en cada una de las fases del SDLC, ofrece una base de datos de conocimientos de vectores de

ataque, mejores prácticas y mitigación, laboratorios en línea, enlazado con otros proyectos como *ASVS* y *Cheatsheets*. Por ejemplo, al ingresar al laboratorio en línea se podrá tener el checklist de requisitos de seguridad de la fase de Requerimientos; para los patrones de diseño de ASVS nivel 2. En la siguiente ilustración podemos ver la categoría Requisitos de verificación de autenticación:

Figura 53. SFK-ASVS



Fuente: OWASP. Security Knowledge Framework [imagen]. [Consultado el 10, marzo, 2023]. Disponible en Internet: <https://secureby.design/projects/manage >.

Se puede crear un proyecto en el que se esté trabajando y obtener los requisitos de seguridad en función de lo que se está desarrollando apoyado en ASVS.
Una función muy interesante es la *generación de requerimientos* que contiene una pipelines o flujo que, de acuerdo a una serie de preguntas, se generan el grupo de requisitos de seguridad según las categorías en ASVS para lo que se definió que se va a desarrollar. Asi mismo tendrá la opción de ver un ejemplo del código que debería tener para cumplir con dicho requisito y mitigar el riesgo.

6.3.11.2 SecurityRAT.

Esta herramienta permite gestionar los requisitos de seguridad en los proyectos de desarrollo ágil, especificando que es lo que se espera en el desarrollo, automatizando el proceso.

Ofrece una interfaz https://securityrat.org/ para crear el Artefacto (sprint backlog) en el cual se debe definir una serie de parámetros como el nivel (niveles ASVS), tipo de autenticación, manejo de sesiones, entre otros. Esta herramienta busca la ruta de seguridad para esos parámetros y generando los requisitos específicos según ASVS.

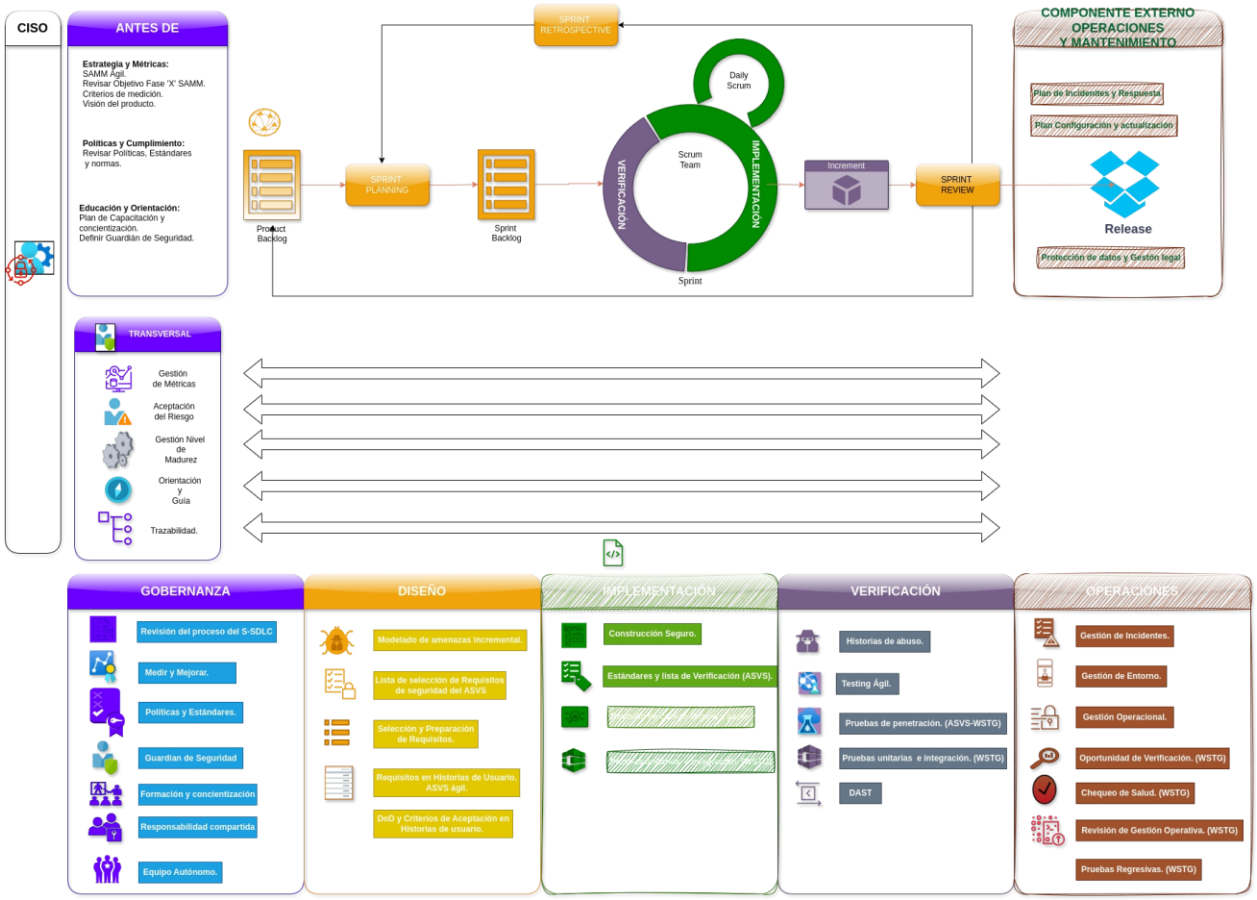
Tiene la opción para exportar este checklist a Excel y/o cargarla a una herramienta de apoyo al desarrollo ágil como Jira, de igual manera se afectará esta actividad (Historia de usuario) en SecurityRAT cambiando de estado.
Permite modelar capacitaciones a partir de los requisitos de seguridad, generando diapositivas.

6.4 PROPONER UN MODELO S-SDLC QUE INTEGRE LAS ACTIVIDADES DE SEGURIDAD, PRINCIPIOS Y PRÁCTICAS DE LA GUÍA OWASP Y EL FRAMEWORK DE SCRUM PARA OBTENER APLICACIONES ROBUSTAS Y CONFIABLES

Finalmente, podemos definir qué actividades se deben priorizar y relacionar de manera organizada en cada fase del S-SDLC, estas son necesarios para que se pueda implementar un modelo ágil y eficiente para el desarrollo de aplicaciones web, que también permita mitigar los riesgos de seguridad más críticos de estas.
A Continuación, se propone el modelo del Ciclo de vida de desarrollo seguro integrando los framework o proyectos más emblemáticos de la fundación OWASP al Scrum:

Recordemos que se basa en la guía ágil de SAMM-v2.0.3 y el objetivo en la FASE I del SAMM, por esta razón se desenfoca algunas prácticas y funciones de negocio.

Figura 54. Fase I SAMM ágil



Fuente: elaboración propia.

Ver en:
https://drive.google.com/file/d/1XC6uaL-iJ7_Co6AKlw1qSS6WVgOTWwz3/view?usp=sharing

6.4.1 Modelo de ciclo de vida de desarrollo seguro SSAWA.

El modelo sigue la ruta de S_SDLC de OWASP, tomando como base sus 3 proyectos más importantes:

- ✓ **SOFTWARE ASSURANCE MATURITY MODEL. (SAMM) Versión 2.0.3 (2022) - GUÍA ÁGIL.**
- ✓ **APPLICATION SECURITY VERIFICATION STANDARD 4.0.3 (ASVS).**
- ✓ **WEB SECURITY TESTING GUIDE V4.2 (WSTG).**

Estos fueron integrados en el framewrok de Scrum tomando como base las prácticas de seguridad de la guía ágil de SAMM, como listado de control de requisitos de seguridad, el ASVS y para la verificación, la WSTG.

Integra y/o adapta las prácticas, corrientes, recomendaciones y/o buenas prácticas de seguridad de cada proyecto en cada una de las fases del ciclo de vida de desarrollo (SDLC), antes del inicio del desarrollo hasta el mantenimiento y operaciones; enfocándose en las funciones de la guía ágil de SAMM, Gobernanza, Diseño y Verificación; sobre todo en otra de las premisas del OWASP, “seguridad hacia la izquierda”, es decir, su prioridad estaría en ese orden.

Por último, sigue el camino de la hoja de ruta que se define para el objetivo de seguridad de cada FASE (I,II,III,IV) del SAMM.

El especialista en seguridad (o CISO) se encargará de planear y enfocar la estrategia de seguridad adoptada por la organización durante todo el S-SDLC. Su enlace principal con el Equipo de trabajo (Scrum team) es el *Guardián de la seguridad.*

Antes que inicie el desarrollo, para la Función de GOBERNANZA y su práctica Estrategia y Métricas:

Crear y promover:

1. Revisión del proceso del S-SDLC (WSTG).

- ✓ Crear/Revisar el modelo de S-SDLC, en este caso se tiene como estrategia de seguridad el SAMM-v2.0.3 en su guía ágil.

- ✓ Presentar a los stakeholters el alcance del plan estratégico de seguridad, el cual contenga el estado actual de prácticas de seguridad y su riesgo, con los beneficios y oportunidades que este modelo de S-SDLC proporciona.

- ✓ **Revisar objetivo de la Fase en curso.** Para cumplir con la estrategia y alcanzar el objetivo de seguridad, alinear con las actividades definidas en la Fase actual de la hoja de ruta SAMM.

2. Medir y Mejorar.

- ✓ **Desarrollar criterios de medición.** Datos para poder medir con frecuencia y poder mejorar. Los básicos podrían ser:

1. Registro de las horas de capacitación.
2. Registro de horas dedicadas al desarrollo de requisitos de seguridad.
3. Datos generados del escaneo y modelado de amenazas, cantidad de amenazas mitigadas y pendientes por mitigar.
4. Estado de prácticas de seguridad SAMM según fase.
5. Cantidad Controles de seguridad realizados vs pendientes.

- ✓ **Garantizar la trazabilidad.**
- ✓ **Dashboard** con las prácticas de seguridad y adicionar los resultados de estas métricas que le sirva a los stakeholter para la toma de decisiones.

3.Políticas Y Cumplimiento. Revisar Políticas, Estándares y normas orientadas con el entorno del producto y objetivos de seguridad para la Gestión de cumplimiento, asegurando que existan políticas, estándares y documentación adecuados.

En la práctica **EDUCACIÓN Y ORIENTACIÓN:**

4.Guardián de Seguridad. Definir y asignar las responsabilidades al guardián de la seguridad del equipo de trabajo (para este caso Scrum Master).

5. Formación y concientización.
Definir el plan de capacitación y/o concientización según el objetivo de seguridad de la Fase en curso. Actividad periódica. Enfocada en cumplir con prácticas de aseguramiento SAMM; Clasificación, evaluación del riesgo y mitigación de amenazas; el análisis de selección de requisitos más relevantes y pruebas basada en requisitos de seguridad.

6.Responsabilidad compartida. Fomentar la responsabilidad de la seguridad compartida por todos los miembros del equipo: Guardián de seguridad y/o Scrum master, Product Owner, Developmet team (Desarrollo y Test) y stakeholters.

7.Equipo Autónomo. Se prioriza la autonomía del equipo en temas de seguridad, presentes en el refinamiento del backlog y la retrospectiva. para realizar una selección de los requisitos más relevantes

Visión del producto. El Scrum team conoce las necesidades del cliente y objetivos del producto.

Durante el Diseño, en la práctica de Evaluación de amenazas:

Se definen las *Épicas,* según la visión y objetivo del producto.

8.Modelado de amenazas Incremental. Crear modelado de amenazas capa cero y capa 1, iniciando con los procesos afectados con desarrollos nuevos, que se van actualizando con cada Sprint, cumpliendo con el modelado incremental.

9.Lista de selección de requisitos de seguridad del ASVS. Integrar el listado de controles de seguridad del ASVS-v4.0.3 para la selección de requisitos.

10. Selección y Preparación de Requisitos.

- ✓ Capacitar y concientizar y al *Scrum team* para realizar una selección de los requisitos más relevantes.
- ✓ Definir nivel.
- ✓ Analizar recursos de vulnerabilidades como CWE, NVD y de amenazas como el MITRE, CAPEC y sus recomendaciones de mitigación al seleccionar los requisitos de seguridad.
- ✓ Seleccionar requisitos del ASVS asociados a mitigar amenazas halladas en el modelado y del entorno del desarrollo, estos debían ser los más relevantes, con mayor impacto en la reducción del riesgo y menor recurso.

11. Requisitos en Historias de Usuario, ASVS ágil.
En la creación del *Product backlog* y el *refinamiento*, en cada HU:

- ✓ Identificar un conjunto mínimo de requisitos de seguridad según el tipo de desarrollo y el modelado de amenazas.
- · Generar las Historias de usuario donde su formato identifique claramente el escenario de abuso del requisito funcional de seguridad ASVS (ASVS ágil).

12. DoD y Criterios de Aceptación en Historias de usuario.

- ✓ Definir los DoD, como requisitos generales de las historias de usuario del spring.
- ✓ Adicionan los requisitos de seguridad específicos para cada HC, según el DoD y el entorno del desarrollo, registrándose como criterios de aceptación.
- ✓ Actualizar en el refinamiento.

Con estas actividades se ha "planificado" la seguridad del producto, ahora el *Scrum team*, "prepara" la seguridad en el *sprint planing* y crea el *sprint Backlog* para que el *Develoment team* "implemente" la seguridad considerando estos conceptos en su evento Daily Scrum y sus artefactos como Impediment log.

Durante el Desarrollo, en la práctica de Construcción Seguro.

13.Construcción Segura. Definir proceso de construcción, describir todo el proceso de principio a fin.

14. Estándares y lista de Verificación (ASVS).

✓Proveer un conjunto de listas de verificación, políticas y/o Estándares de codificación segura.
✓ **V**alidar que el código se está desarrollando de conformidad con los requisitos de los estándares de codificación segura.
✓ Análisis de código fuente.

15.Revisión de código seguro x pares. El desarrollador senior debe liderar revisión de código seguro.

16.Pruebas unitarias e integración. (WSTG). Verificar estática y dinámicamente (SAST, DAST) que el código fuente desarrollado no incluye vulnerabilidades potenciales y cumple con los estándares de codificación segura.

Durante la Verificación, para la práctica Pruebas Basadas en Requisitos.

13.Historias de abuso. Probar los escenarios de abuso asociados a los requisitos seguridad definidos en las historias de usuario como criterios de aceptación.

Para la corriente Testing de seguridad.

14. Testing Ágil.
✓ Automatizar las pruebas SAST, DAST Y SCA.
✓ Enfocar pruebas manuales en el código modificado o nuevo.

15. Pruebas de penetración. (ASVS-WSTG).

✓ Realizar pentesting en la funcionalidad nueva o modificada.
✓ Documentar como guías los procedimientos de pruebas. Lista de verificación de casos de prueba.

16.DAST. Priorizar pruebas para requisito de nivel L1. La mayoría de los controles de seguridad de nivel L1 se pueden verificar con estas pruebas.

17.Pruebas de integración. (WSTG). Validar si las vulnerabilidades son reales y pueden ser aprovechadas por los atacantes. Actividad periódica.

El éxito de la verificación de estas pruebas de seguridad debe darse a conocer en el sprint review con el cliente.

Como ya se dijo, el campeón de la seguridad tiene un papel muy importante que viene del componente de colaboración del Scrum. En SAMM ágil es el enlace entre Development team y el especialista o equipo de Seguridad, eliminará cualquier impedimento que afecte el correcto funcionamiento del S-SDLC, apoyará cuando sea posible, pero el equipo no debe

dejar de ser autónomo y responsable de la seguridad. En este modelo, además de las mencionadas actividades, es quien apoyara algunas actividades durante todo el spring.

TRANSVERSAL:

Gestión de Métricas. Promueve y realiza seguimiento al registro de las métricas para cumplimiento de los criterios de medición, datos que ayudaran a la gestión del proceso, análisis del riesgo y toma de decisiones.

Aceptación del Riesgo. Gestionar las métricas asociadas al riesgo para tomar decisiones sobre aceptarlos, mitigarlos o transferirlos.

Gestión Nivel de Madurez. Verificar que las actividades y los requisitos de seguridad en las Historias de usuario estén alineados con el nivel de madurez objetivo del modelo SAMM y el nivel definido en ASVS.

Orientación y Guía. Orientar al equipo en cada una de las actividades del modelo, por ejemplo, Técnicas de modelado de amenazas, de desarrollo seguro, Herramientas de pruebas DAST. Conceptos específicos sobre los marcos integrados Scrum, SAMM, ASVS y WSTG y los relacionados del OWASP. Fortalecer la seguridad compartida.

Trazabilidad. Seguimiento a los requisitos de seguridad desde su selección hasta las pruebas realizadas. Datos relacionados como estado del requisito (diferido, aplazado, desarrollado), Id de pruebas realizadas, reportes de análisis de vulnerabilidades.

Durante el Mantenimiento y las Operaciones (Componente Externo).

- ✓ **Realizar revisiones de la gestión operativa. I**mplementar un proceso que detalle la gestiona operativa de la aplicación e infraestructura.
- ✓ **Realizar controles de salud periódicos.** Realizar controles periódicos del estado de la aplicación y la infraestructura.
- ✓ **Garantizar la verificación de cambios.** Verificar los cambios para que el nivel de seguridad no haya sido afectado e integrarlo al proceso de gestión de cambios.

Se logró integrar las metodologías, procesos, buenas prácticas, recomendaciones y herramientas propuestas por la fundación OWASP, tomando como base las prácticas del modelo SAMM ágil, como lista de requisitos de seguridad, el estándar ASVS y para las pruebas, la guía WSTG, estas se aplicaron en todas las fases del S-SDLC desde antes del inicio del desarrollo hasta la fase de verificación y en cada sprint del Scrum

Se podría decir que se definieron procesos generales como cimientos sólidos del modelo SAAWA propuesto:

- ✓ Adoptar un modelo de S-SDLC.
- ✓ Identificar el estado actual de las prácticas de seguridad y definir un objetivo.
- ✓ Concientización y entrenamiento.
- ✓ Modelado de amenazas incremental.
- ✓ Análisis eficiente de la selección de requisitos de seguridad en HU.
- ✓ Construcción y Revisión de código seguro.
- ✓ Automatización de pruebas (SAST,DAST y SCA).
- ✓ Definir enfoque y plan de testing de seguridad.

El mayor impacto lo tendrá el apetito de los interesados por gestionar el riesgo, la guía del especialista en seguridad y el liderazgo del guardián de la seguridad.

La visibilidad de todo el modelo (heredado de Scrum) permitió el seguimiento de cada fase del S-SDLC, solucionar los impedimentos que se presentaron e identificar actividades que se pueden mejorar.

Al iniciar, se pudo identificar como era el estado de las actividades que realizaba la casa de desarrollo, en materia de seguridad en su ciclo de vida de desarrollo actual y su gestión de desarrollo de software con Scrum.

Se puede afirmar que la organización SAC y su equipo de trabajo (Scrum Team) tiene el entrenamiento para el manejo de los recursos que les permiten estar actualizados con las vulnerabilidades más críticas asociadas a las aplicaciones web y como identificarlas. Un mejor grado de concientización sobre los riesgos generados por la materialización de una amenaza al explotar una vulnerabilidad; y la complejidad de corrección de fallos de seguridad en etapas finales del desarrollo.

Se realizó la descripción del proceso de integración de los 3 proyectos principales del OWASP con Scrum, analizando cada uno y detallando la relación de las actividades de cada framework con cada fase del S-SDLC. El equipo tiene la capacidad y la autonomía para seguir las actividades definidas en la propuesta del modelo.

Finalmente, se definió la propuesta del modelo de S-SDLC SSAWA, donde se representa gráficamente la integración de actividades de los framework Scrum, SAMM, ASVS y WSTG en su forma ágil.

7 CONCLUSIONES

Esta revisión permitió detallar la implementación del modelo de desarrollo seguro de OWASP con sus proyectos más importantes, SAMM-v2.0.3, ASVS-v4.0.3 y WSTG-v4.2, adaptados al pensamiento ágil en cada caso, para integrarlo con el framework Scrum.

Inicialmente, se determinó el estado actual de la seguridad en las fases de su SDLC, que se integrarían al framework SCRUM en este caso el Análisis, Diseño, Desarrollo, Pruebas, Despliegue y Mantenimiento, evidenciando la ausencia de estrategias en cuanto a la seguridad informática en la organización SAC, destacando la carencia de cultura organizacional sobre este tema y la falta de un marco de buenas prácticas de desarrollo. Pudimos confirmar que no se incorpora la seguridad en su SDLC.

Se analizó la información de los recursos que describen las vulnerabilidades más comunes que convierten una aplicación web en un vector de ataque si no cuenta con buenas prácticas de desarrollo de software seguro para mitigar el riesgo que supone un evento de seguridad generado por una amenaza. Así mismo, se revisó los tipos de prueba que permiten encontrar estas brechas de seguridad. Estos recursos fueron fundamentales para planificar el componente de concientización y capacitación del proyecto. Que el equipo de trabajo de la organización conozca y este consciente sobre los riesgos a los que están expuestos sus aplicaciones puede ser la base más sólida de cualquier modelo, buenas prácticas, recomendaciones o estándar sobre desarrollo de software seguro que se quiera implementar.

Se evidenció concientización de los líderes asignados por SAC para el proyecto, así como la de gerencia respecto a la promoción y definición de políticas para integrar prácticas de desarrollo seguro en sus aplicaciones y de los beneficios de esto para la organización y sus stakeholders.

Se logró un entendimiento profundo de los framework de OWASP y su relación con el modelo de desarrollo ágil, lo cual permitió identificar la dinámica entre los componentes y actividades específicas para formar un solo modelo de desarrollo de software seguro ágil.

Se integró los marcos de trabajo (SAMM ágil, ASVS y WSTG) seleccionados del modelo de desarrollo seguro de OWASP con el framework SCRUM, cumpliendo con el compromiso ético de iniciar con una estrategia para incluir en su marco de trabajo ágil, la seguridad en cada una de las Fases de su SDLC. La integración entre un modelo de desarrollo ágil soportado en el framework SCRUM a una modelo como el de OWASP, mejora el componente de seguridad en el S-SDLC sin perder sus ventajas de entrega continua, iteraciones cortas, retroalimentación inmediata, control de riesgo y adaptación al cambio, mejora la calidad del software, reduce el tiempo de desarrollo de controles de seguridad y reduce los hallazgos de vulnerabilidades.

Se logró contribuir al desarrollo de software seguro integrando el modelo de OWASP al framework SCRUM porque se tendrá un procedimiento para planificar el inicio de un S-SDLC

Ágil, que se podrá implementar de manera rápida y contribuir al cambio de cultura hacia el desarrollo de software seguro en cualquier organización del sector de desarrollo de software.

8 RECOMENDACIONES

Necesitaríamos tener un componente de capacitación y concientización mucho más robusto con algún componente psicológico para mejorar entrega y recepción de conocimientos técnicos y de gobernanza.

Este estudio resalta la importancia de tener políticas de desarrollo básicas, estándares de desarrollo de acuerdo con las tecnologías utilizadas, documentación de diseño y arquitectura, casos de uso, UML y pruebas unitarias e integradas.

Profundizar en conocimientos de métricas de gestión (Esfuerzo, Resultado y entorno) y los demás KPI asociados a los objetivos de los framework implementados para integrarlos.

Este procedimiento puede ser utilizado por organizaciones con pocos recursos que quieran iniciar con su S-SDLC Ágil.

Futuras investigaciones para automatizar la integración de los framework estudiados, otros relacionados y sus herramientas, sería de gran ayuda para reducir tiempos de implementación del S-SDLC

Un posible enfoque sería conocer el impacto que tendría para las casas de software una política obligatoria en la adopción de un modelo de S-SDLC por parte del estado.

Será conveniente seguir con la investigación para incluir en el modelo S-SDLC Ágil las fases que no se abordaron en este proyecto.

9 BIBLIOGRAFÍA

AHMAD, Areeba; HILL, Richard; LU, Juan; MACCLUSKEY, Lee y WADE, Steve. Un análisis sobre la metodología Scrum en el desarrollo global de software – GSD. [En linea]. Lugar de publicación: 2020. [consultado el 02 de Enero de 2023]. Disponible en: https://ieeexplore- ieee-org.bibliotecavirtual.unad.edu.co/document/9457918

BORGES FRANCIA, Mauro; CARDOSO, Alejandro; LAMOUNER, Junior y TAVARES MOTA. Camila . Agile Short Unified Process - ASUP: A hybrid methodology supported by the adaptation of the Scrum framework and the Unified Process mode. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação. [En línea]. 2022, Vol.46. [consultado el 02 de enero de 2023]. ISSN 1646-9895. Disponible en: http://www.scielo.pt/scielo.php?script=sci_arttext&pid=S1646-98952022000200071&lang=es#B17

BORGES FRANCIA, Mauro; CARDOSO, Alejandro; LAMOUNER, Junior y TAVARES MOTA. Camila . Agile Short Unified Process - ASUP: A hybrid methodology supported by the adaptation of the Scrum framework and the Unified Process mode. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação. [En línea]. 2022, Vol.46. [consultado el 03 de enero de 2023]. ISSN 1646-9895. Disponible en: http://www.scielo.pt/scielo.php?script=sci_arttext&pid=S1646-98952022000200071&lang=es#B17

CAÑO QUINTERO, Jose Joaquin. DevSecOps: integración de herramientas SAST, DAST y de análisis de Dockers en un sistema de integración continua. [en línea]. Trabajo de Grado. Universitat Oberta de Catalunya, Catalunya.: 2019. [Consultado 10, marzo,2023]. Disponible en: https://openaccess.uoc.edu/handle/10609/95987

COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Decreto 1377. (27, junio, 2013). Por el cual se reglamenta parcialmente la Ley 1581 de 2012. En: Sistema único de Información Normativa. Bogotá D.C.. 2013. 28 p.

COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Ley 1273. (05, enero, 2009). por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado - denominado “de la protección de la información y de los datos”- y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones. En: Sistema único de Información Normativa. Bogotá D.C.. 2009. 4 p

COLOMBIA. MINISTERIO DE JUSTICIA Y DEL DERECHO. Ley 1928. (24, julio, 2018). por medio de la cual se aprueba el “Convenio sobre la Ciberdelincuencia”, adoptado el 23 de noviembre de 2001, en Budapest.. En: Sistema único de Información Normativa. Bogotá D.C.. 2018. 20 p.

CORTES RUGELES, Diana Marcela; ARDILA GUZMAN, Alix Victoria. Metodología para la implementación de un sistema integrado de gestión con las normas ISO 9001, ISO 20000 e

ISO 270001 [en línea]. Tesis especialización. Bogotá: Universidad EAN, 2019 [consultado el 26, enero, 2023]. 72 p. Disponible en Internet: <http://hdl.handle.net/10882/2779>

CVE®. cve-website. cve-website [página web]. [Consultado el 2, enero, 2023]. Disponible en Internet: <https://www.cve.org/About/Overview>

CWE - 2022 CWE top 25 most dangerous software weaknesses [Anónimo]. 3, agosto, 2022. [Consultado el 3, marzo, 2023]. Disponible en Internet: <https://cwe.mitre.org/data/definitions/918.html>

DEPARTAMENTO DE SALUD Y PROTECCIÓN SOCIAL. [en línea] Bogotá: La entidad. [consultado el 26 de enero de 2023]. Disponible en: https://www.minsalud.gov.co/Ministerio/Institucional/Procesos%20y%20procedimientos/CVSS01.pdf

DEPARTAMENTO NACIONAL DE PLANEACIÓN. POLÍTICA NACIONAL DE SEGURIDAD DIGITAL. [en línea] Bogotá: La entidad. [consultado el 04 de enero de 2023]. Disponible en: https://colaboracion.dnp.gov.co/CDT/Conpes/Econ%C3%B3micos/3854.pdf

El tiempo. INVIMA, en alerta por ataque cibernético. [sitio web]. Bogotá: la entidad. [04, diciembre, 2022]. Disponible en: https://www.eltiempo.com/salud/invima-esta-en-alerta-por-ataque-cibernetico-707158

ESET. ESET-security-report-LATAM2022. [Sitio WEB]. Bratislava. La entidad. [30, Noviembre, 2022]. Disponible en: https://www.welivesecurity.com/wp-content/uploads/2022/10/ESET-security-report-LATAM2022.pdf

ESTRADA-VELASCO, Marco Vinicio, et al. Revisión sistemática de la metodología scrum para el desarrollo de software. En: Dominio de las ciencias [en línea]. 2021. vol.7, no.4 [consultado el 9, febrero, 2023]. Disponible en Internet: <https://dominiodelasciencias.com/ojs/index.php/es/article/view/2429/0>.

FEDESOFT. Llega la décima versión de los premios más importantes del mundo del software y las TI en Colombia. [Sitio WEB]. Bogotá. La entidad. [04, diciembre, 2022]. Disponible en: https://fedesoft.org/llega-la-decima-version-de-los-premios-mas-importantes-del-mundo-del-software-y-las-ti-en-colombia/

GOMEZ FUENTES, Maria Del Carmen; CERVANTES OJEDA, Jorge y GONZALEZ PEREZ, Pedro Pablo. Fundamentos de ingeniería de software [en línea]. Cuajimalpa: México : UAM, Unidad Cuajimalpa, 2019 [consultado el 22, enero, 2023]. 277 p. Disponible en Internet: <http://ilitia.cua.uam.mx:8080/jspui/handle/123456789/1000 >. ISBN 978-607-28-1659-6.

IBM. ¿Qué es el desarrollo de software?. [En línea]. Bogotá. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://www.ibm.com/co-es/topics/software-development

INCIBE. Vulnerabilidad. [En línea]. Madrid. La entidad. [consultado el 22 de diciembre de 2022]. Disponible en: https://www.incibe.es/aprendeciberseguridad/vulnerabilidad

INTRODUCCIÓN - OWASP Top 10:2021 [Anónimo]. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [página web]. (2022). [Consultado el 8, marzo, 2023]. Disponible en Internet: <https://owasp.org/Top10/es/A00_2021_Introduction/>

JUNTA DE ANDALUCÍA. Metodología y Frameworks de testeo de la seguridad de las aplicaciones. [Sitio WEB]. Andalucía. La entidad. [01, diciembre, 2022]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/sites/default/files/historico/1.3.1/contenido-recurso-216.html

LERMA GONZALES, Hector Daniel. Metodología de la investigación: propuesta, anteproyecto, y proyecto. Bogotá D.C.: Ecoe Ediciones, 2009. 72 p. ISBN 978-958-648-602-6.

LÓPEZ ALVAREZ, Diana Maria. Método para el desarrollo de software seguro basado en la ingeniería de software y ciberseguridad. Universidad Piloto de Colombia. Bogotá D.C.:[En línea]. [consultado el 20 de enero de 2023]. Disponible en: http://polux.unipiloto.edu.co:8080/00003022.pdf

LÓPEZ RODRIGUEZ, Michelle. Análisis de contenedores Docker e integración de herramientas SAST y DAST. [en línea]. Trabajo de Grado. Universidad Autónoma de Barcelona, Barcelona.: 2020. [Consultado 28, marzo,2023]. Disponible en: https://ddd.uab.cat/record/226837

MINISTERIO DE HACIENDA Y ADMINISTRACIONES PÚBLICAS. Magerit versión 3.0: metodología de análisis y gestión de riesgos de los sistemas de información. [en línea]. Magerit Libro I: Método. Madrid: La entidad. 127 p. [Consultado el 2, enero, 2023]. Disponible en Internet: <https://www.ccn-cert.cni.es/documentos-publicos/1789-magerit-libro-i-metodo/file.html>.

OWASP. Application Security Verification Standard 4.0.3. [Sitio WEB]. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

OWASP. Introduction. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/

OWASP. OWASP Application Security Verification Standard. [Sitio WEB]. La entidad. [consultado el 29 de Enero de 2023]. Disponible en: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-es.pdf

OWASP. OWASP SAMM versión 2. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 03 de enero de 2023]. Disponible en: https://drive.google.com/file/d/1cl3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view

OWASP. OWASP Web Security Testing Guide. [Sitio WEB]. Edgewater Place. La entidad. [30, noviembre, 2022]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

OWASP. OWASP Web Security Testing Guide. [Sitio WEB]. Edgewater Place. La entidad. [consultado el 02 de enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

OWASP. SAMM AGILE GUIDANCE. [Sitio WEB]. La entidad. [consultado el 17 de marzo de 2023]. Disponible en: https://owaspsamm.org/guidance/agile/#General

OWASP. The OWASP Testing Project. [Sitio WEB]. Eoin Keary. [consultado el 30 de junio de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/

OWASP. Who is the OWASP® Foundation?. [En línea]. Maryland. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://owasp.org/

OWASP. WSTG-v4.2. [Sitio WEB]. La entidad. [consultado el 29 de Enero de 2023]. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

PARDO, César, et al. Modelo de referencia para la adopción e implementación de Scrum en la industria de software. En: Investigación e innovación en ingenierías [en línea]. 2020. vol. 8, no.3 [consultado el 6, febrero, 2023], p.15. Disponible en Internet: <https://dialnet.unirioja.es/servlet/articulo?codigo=7799075>. ISSN 2344-8652.

PIÑA TABORDA, Jessica, et al. Análisis prospectivo de la industria de desarrollo de software en Colombia. En: Punto De Vista [en línea]. 2020. vol.10, no.2(16) [consultado el 26, enero, 2023], p. 23. Disponible en Internet: <https://doi.org/10.15765/pdv.v11i16.1415>

RedHat. ¿Qué es la metodología ágil?. [En línea]. Bogotá. La entidad. [consultado el 21 de diciembre de 2022]. Disponible en: https://www.redhat.com/es/devops/what-is-agile-methodology

ROJAS PEÑA, Hernán Mauricio. Aplicación de la metodología Magerit para el análisis de riesgos de los sistemas de control en la estación Tenay del Oleoducto Alto Magdalena [en

línea]. Trabajo de grado. Neiva: Universidad Nacional Abierta y a Distancia UNAD, 2019 [consultado el 11, enero, 2023]. 97 p. Disponible en Internet: <http://repositorio.espe.edu.ec/xmlui/handle/21000/20405>

Scrum.Org. What is Scrum?. [En línea]. Andalucía. La entidad. [consultado el 22 de diciembre de 2022]. Disponible en: https://www.scrum.org/resources/what-is-scrum

scrumguides.org. La Guía de Scrum. [Sitio WEB]. La entidad. [consultado el 28 de Enero de 2023]. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

SEGURIDAD EN el ciclo de vida del software - SSHTeam [Anónimo]. SSHTeam - Expertos en ciberseguridad [página web]. (2020). [Consultado el 30, enero, 2023]. Disponible en Internet: <https://sshteam.com/ssdlc-y-desarrollo-seguro/>

STATE OF agile report [Anónimo] [en línea]. [s.l.]: stateofagile, 2022 [consultado el 1, marzo, 2023]. 22 p. 2022-16th. Disponible en Internet: <https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf>

TASSEY, Gregory. The economic impacts of inadequate infrastructure for software testing final report [en línea]. Gaithersburg: RTI, 2002 [consultado el 10, enero, 2023]. 309 p. 97. Disponible en Internet: <https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

VANEGAS, BARRERO, Alejandro. SEGURIDAD PARA MINIMIZAR RIESGOS EN EL DESARROLLO DEL SOFTWARE. Universidad ECOTEC. Bogotá D.C.:[En línea]. [consultado el 20 de enero de 2023]. Disponible en: http://polux.unipiloto.edu.co:8080/00003022.pdf

Welivesecurity. Ransomware Hive atacó a más de 1300 compañías en el mundo. [Sitio WEB]. Bratislava. La entidad. [30, noviembre, 2022]. Disponible en: https://www.welivesecurity.com/la-es/2022/11/22/ransomware-hive-ataco-1300-companias-mundo/

WHAT IS Scrum? [Anónimo]. Scrum.org [página web]. [Consultado el 30, noviembre, 2022]. Disponible en Internet: <https://www.scrum.org/resources/what-is-scrum>

10 ANEXOS

Anexo A. AUTORIZACIÓN EJECUCIÓN PROYECTO.

Ver en el link: https://drive.google.com/file/d/1D67Rov20svnMqMiOgDDt2RxqOW8k_JMu/view?usp=sharing

Anexo B. ACUERDO DE CONFIDENCIALIDAD.

Ver en el link: https://drive.google.com/file/d/1OBz6Jzhb93z-e72RT1cuRidch-jJfwK2/view?usp=sharing