LA SEGURIDAD INFORMÁTICA EN EL DESARROLLO DE APLICACIONES WEB MEDIANTE EL USO DE LA METODOLOGÍA OWASP

TANIA SIERRA HUERTAS

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ
2022

LA SEGURIDAD INFORMÁTICA EN EL DESARROLLO DE APLICACIONES WEB MEDIANTE EL USO DE LA METODOLOGÍA OWASP

TANIA SIERRA HUERTAS

Proyecto de Grado – Monografía presentado para optar por el título de
ESPECIALISTA EN SEGURIDAD INFORMÁTICA

Director
Ingeniero Danny Fernando León Jaramillo

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D. C.
2022

NOTA DE ACEPTACIÒN

Firma del Presidente de Jurado

Firma del Jurado

Firma del Jurado

Bogotá., Fecha sustentación

DEDICATORIA

Con cariño y amor dedico este trabajo a mi familia, que ha soportado mi ritmo de trabajo y estudio siempre resguardado a altas horas de noche y a la madrugada ya que el silencio permite mi concentración. Y en especial a Dios y a mí que muchas veces quise claudicar y dejar todo a un lado. Dios, mi persistencia y la de mi familia aun que me he demorado años en lograr un título y a pesar de todas las dificultades económicas y emocionales. Siempre esta Dios y mi familia para decirme animo tú vas a lograrlo. Gracias a los profesores de la UNAD que no han sido indiferentes a mis preguntas y mis opiniones. Gracias a la vida que me ha permitido estudiar que ha sido mi mayor deseo desde niña.

AGRADECIMIENTOS

Agradezco a todos los tutores que han hecho parte de mi proceso de aprendizaje a distancia y a la Universidad Nacional Abierta y a Distancia UNAD, que me ha permitido ser parte de su comunidad educativa.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE CUADROS

pág.

GLOSARIO

AMBIENTE DE DESARROLLO: Es un entorno interactivo de una aplicación informática que proporciona diferentes servicios para facilitar la programación.

ANIDACIÓN: Estructuras o funciones dentro de otra función o estructura.

API: Application Programming Interfaces, Interfaz de programación de aplicaciones, conjunto de protocolos que se utilizan para desarrollar e integrar el software utilizando reglas determinadas.

BACK END: Entorno donde se encuentra las configuraciones, acciones y conexiones entre la base de datos y el servidor.

DESPLIEGUE: Es cuando un sistema o software ya se encuentra disponible para su uso.

CODIFICACIÓN: Transformar un código fuente gracias a un lenguaje de Programación en alguna aplicación o programa.

FRAMEWORK: Entorno de trabajo específico para realizar un proyecto de software

FRONT END: Entorno al lado de cliente el cual permite la interacción entre los usuarios y la aplicación (sitio web).

FUNCIÓN EN PROGRAMACIÓN: Es una parte de un programa que permite calcular un valor realizando una tarea específica.

JSON: JavaScript Object Notation (Notación de Objeto JavaScript): Formato de almacenamiento para trasladar datos del servidor al cliente

INTEGRIDAD DEL SOFTWARE: Es la capacidad de garantizar que los datos no han sido modificados ni el código que compone la aplicación

LIBRERIAS: Conjunto de archivos que permiten facilitar la programación

PRODUCCIÓN: Cuando un producto o aplicación cumple con los requisitos del cliente

VALIDACIÓN DE DATOS: Cumplir con ciertas características y realizar verificación de los datos, permitiendo garantizar su integridad y su entrega segura.

PARAMETROS: Variable que es usada para recibir un valor de entrada.

SERIALIZACIÓN: Codificación de objeto el cual será trasmitido por medio de una red.

VARIABLE: Elemento que almacena un dato con un valor

TESTING: Prueba realizada al software para verificar su funcionamiento.

RESUMEN

Gracias a los avances del Internet y las necesidades de consulta y almacenamiento, se han podido utilizar diferentes aplicaciones web que permiten brindar información de forma rápida y eficiente; para mantener la integridad de los datos y la seguridad de las aplicaciones es indispensable que desde su definición, diseño, desarrollo, implementación y mantenimiento (Ciclo de vida de desarrollo de software) se determine las posibles vulnerabilidades o riesgos utilizando la metodología OWASP que permite implementar principios y buenas prácticas de seguridad.

PALABRAS CLAVE: Metodología, Seguridad, Software, Vulnerabilidad.

ABSTRACT

Thanks to the advances of the Internet and the consultation and storage needs, it has been possible to use different web applications that allow information to be provided quickly and efficiently; To maintain the integrity of the data and the security of the applications, it is necessary that from their definition, design, development, implementation and maintenance (Software development life cycle) possible vulnerabilities or risks are determined using the OWASP methodology that allows to implement principles and good security practices.

KEY WORDS: Methodology, Security, Software, Vulnerability.

INTRODUCCIÓN

La seguridad en las aplicaciones web se ha convertido en un requisito obligatorio para evitar pérdida o fuga de información; es ineludible empezar a aplicar los conocimientos y habilidades para desarrollar aplicaciones seguras que cumplan con las metodologías que permiten dar como resultado software seguro.

La metodología Owasp puede emplearse en todas las etapas del siclo de vida del desarrollo de software, si es aplicada correctamente es posible mejorar la seguridad de diferentes tipos errores y ataques; al no considerar esta metodología el proyecto de desarrollo de las aplicaciones web puede estar más expuesto a múltiples vulnerabilidades. La variedad de la metodología Owasp como sus proyectos de seguridad, documentación y herramientas permite exponer, divulgar e identificar las vulnerabilidades y ataques más comunes que sufren las aplicaciones web en los diferentes entornos que se han implementado.

1. DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES DEL PROBLEMA

La ausencia de buenas prácticas, preparación y metodologías abre un riesgo de seguridad o vulnerabilidad a amenazas, posibles errores de programación como por ejemplo uno que se presenta con mayor frecuencia con la validación de datos en la cual se puede ser víctima de inyección SQL; tiene una frecuencia máxima del 19% según el reporte de Owasp top 10 del 2021. Esto acontece cuando los datos no se validan, no son parametrizados. Estas inyecciones de código malicioso pueden estar dentro de las consultas, bibliotecas de navegación de gráficos. Es ineludible revisar el código fuente;[1] estas fallas de seguridad pueden ser más habituales en aplicaciones.

Se requiere la identificación de las posibles amenazas en el ciclo de vida del software ya que pueden existir peligros no identificados.

Esta problemática se evidencia según la estadística de la revista digital IT Digital Security en la cual muestra que “el 82% de las vulnerabilidades de las aplicaciones web están en el código fuente; estadística realizada en el año 2019. Este porcentaje tan alto pone en manifestación que los desarrolladores no le dan la suficiente importancia a la seguridad”[2].

También se demuestra en el artículo de IT Governance. “El 65% de las vulnerabilidades encontradas provienen de un mal desarrollo en las aplicaciones, como fallos en la codificación o configuraciones incorrectas. Las organizaciones deben encontrar un sistema de defensa por capas con el objetivo de detectar errores antes de que sea demasiado tarde”[3]

Sin contar que es posible que exista otras problemáticas como la falta de motivación para que haya la participación y el compromiso tanto de los desarrolladores como del cliente lo cual hace que se vea reflejado en levantamiento de requerimientos incompletos o variables.

La falta de recursos y los tiempos de entrega limitados conlleva a que el desarrollo no tenga un ciclo de tiempo adecuado para incluir la seguridad al proyecto. El afán por cumplir los requerimientos tanto en funcionabilidad como en operabilidad hace que solo se realice un testeo de funcionamiento y no de seguridad. Sumado la carencia de competencia tecnológica por parte del equipo de desarrollo, no tener claridad en los objetivos, no utilizar la metodología adecuada pueden llevar a un

1 A03:2021 – Injection Owasp Top 10-2021
2 IT Digital Security, El 82% de las vulnerabilidades de las aplicaciones web están en el código fuente. España. 2020.
3 IT Governance. El 52% de las aplicaciones web tiene importantes fallos técnicos. España.2018.

resultado de una aplicación vulnerable la cual puede presentar riesgo en la integridad de los datos e información que la compone, con frecuencia solo se considera el papel de “atacado o victima” al instante de realizar pruebas (se debe considerar también el papel de atacante) para tener ambas percepciones de posibles situaciones de vulnerabilidad.

Esto demuestra que no se implementan las metodologías y las buenas prácticas de seguridad, dejando una puerta abierta a cualquier tipo de intruso, tanto el inexperto que quiere buscar información para su provecho o curiosidad, el experto que quiere cobrar alguna recompensa por cualquier dato que pueda capturar, esto es posible a las diferentes herramientas que se encuentran en la web y que son de fácil acceso.

Es apremiante generar la conciencia y la responsabilidad a la hora de iniciar un proceso de desarrollo para que los productos a entregar sean confiables y seguros lo cual da un nivel de profesionalismo más ético y responsable.

1.2 FORMULACIÓN DEL PROBLEMA

¿Cómo determinar a partir de la metodología OWASP la seguridad informática en el desarrollo de aplicaciones web?

2 JUSTIFICACIÓN

Gracias a que la web es el sistema de gestión más frecuente y popular, convirtiéndose en una fácil herramienta para prestar servicios, mostrar productos, entretenimiento, negocios empresariales y financieros. Este crecimiento también ha hecho que crezcan las vulnerabilidades y los riesgos de exponer la información.

Es preciso evaluar desde cualquier punto del ciclo de vida del software, si se cumplen ciertos requerimientos para el buen funcionamiento acompañado de la seguridad, buscando siempre la integridad, la confidencialidad y la disponibilidad de la información.

Para tener claro el control de seguridad es esencial realizar un análisis inicial de requerimientos (posibilidades, objetivos a cumplir), con este análisis se debe considerar la arquitectura de seguridad, la infraestructura tecnológica solicitada y la que se tiene en el tiempo, acompañada de una de las mejores metodologías en desarrollo como es OWASP, esta metodología permita mejorar la seguridad y brindar nociones al grupo de desarrollo enseñándole técnicas que permitan generar una cultura de desarrollo responsable como un software de alta calidad brindado seguridad y conocimiento al equipo de trabajo.

3 OBJETIVOS

3.1 OBJETIVOS GENERAL

Establecer cuáles son las vulnerabilidades de seguridad en el desarrollo de aplicaciones web a partir de la metodología OWASP.

3.2 OBJETIVOS ESPECÍFICOS

- Inspeccionar los acontecimientos más importantes sobre ataques y amenazas de seguridad en aplicaciones web.

- Evaluar las prácticas deficientes presentes en el desarrollo de aplicaciones web en sus fases de configuración y codificación.

- Estimar dentro del ciclo de vida de desarrollo de software donde se presentan las mayores vulnerabilidades en la seguridad del desarrollo de aplicaciones web.

- Proponer una metodología para el desarrollo seguro de aplicaciones web, acompañado de buenas prácticas basadas en OWASP para los desarrolladores de aplicaciones web.

4 MARCO REFERENCIAL

Las metodologías de desarrollo de software permiten utilizar un ambiente controlado con objetivos para el trabajo en equipo; el objetivo es construir una aplicación que desempeñe los requerimientos cumpliendo con ciertas exigencias de seguridad donde se debe tener presente posibles errores de seguridad y su impacto en el producto solicitado.

Los tres pilares de la seguridad de la información se encuentran enfocados en la integridad de la información, la confidencialidad y la disponibilidad

Para cumplir estos requisitos y mejoras la seguridad en los desarrollos de software se ha formado una fundación sin ánimo de lucro que presenta proyectos de código abierto para que la comunidad participe y forme parte de su fundación para mejora y protección de la Web; esta fundación es llamada OWASP (Open Web Application Segurity Project.

El proyecto que más se recomienda para la codificación segura, detección de riesgos y evaluación de proceso es OWASP top 10, en los cuales se considera las 10 amenazas más frecuentes que pueden sufrir las aplicaciones web.

Gracias a la recopilación de datos de vulnerabilidades en aplicaciones web procedentes de diferentes fuentes como proveedores, consultorías de seguridad recompensas por errores, junto con contribuciones de la empresa/organización[4], a más información es más preciso el análisis, una de las fuentes de información son de las CWE que es un sistema que enumera y categoriza vulnerabilidades, el programa CVE es una comunidad que identifica y cataloga vulnerabilidades de seguridad divulgadas públicamente en el cual se registran a la fecha 174,789 registro CVE (Common Vulnerabilities and Exposeres) Vulnerabilidades y Exposiciones Comunes.[5]

OWASP también presenta entre sus múltiples proyectos la guía de prueba de seguridad Web de OWASP; esta guía realiza pruebas tanto a la aplicación web como a los servicios web, desde antes de iniciar el desarrollo, en el momento del desarrollo, durante el despliegue, el mantenimiento y la operación.

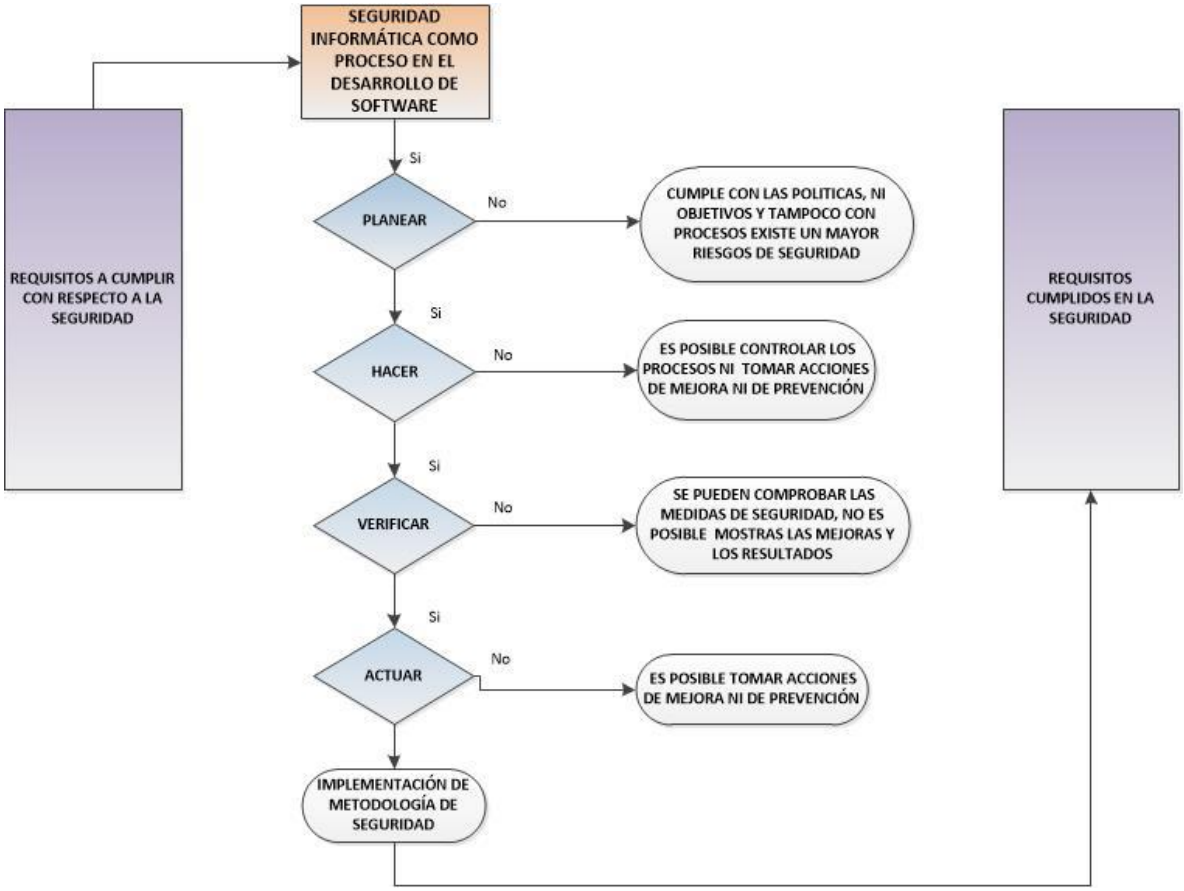
4.1 MARCO TEÓRICO

Las metodologías están pensadas para que se aplique las técnicas más adecuadas según las necesidades de un desarrollo seguro, estas metodologías están fundadas

4 Los diez mejores de OWASP. Plan de análisis de datos OWASP Top 10 2020
5 CVE. Misión del Programa CVE

en procesos para poder ser más eficientes. Su objetivo principal es mejorar la calidad del software en cada una de sus fases.

La gran mayoría de metodologías están basadas en cuatro procesos del Sistema de Gestión de Seguridad Informática y que son principios esenciales de cada estructura metodológica del:
modelo PHVA
Figura 1. Modelo PHVA aplicado al Desarrollo de Software

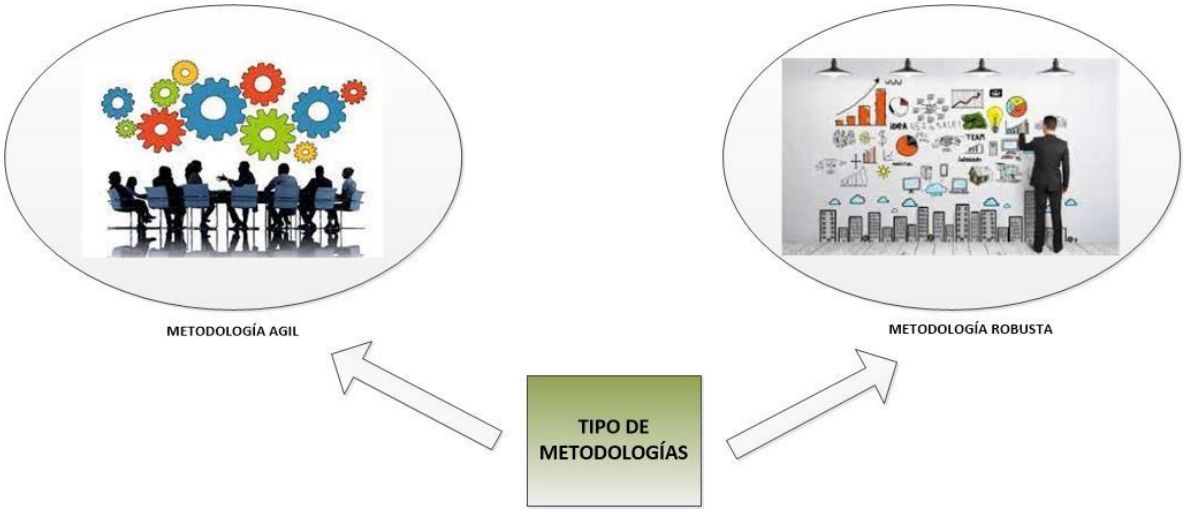


Fuente: Propia.

Las metodologías para el desarrollo de software se pueden dividir en Metodologías Agiles y Metodologías robustas

La metodología ágil se desarrolla a la par entre el grupo de trabajo y el cliente mientras que la metodología robusta realizar una documentación juiciosa de cada proceso. A continuación, se representan en la siguiente figura 2:

Figura 2. Tipo de Metodologías



Fuente: Propia.

Las metodologías agiles son las que gestionan una comunicación constante y directa con el cliente, estableciendo los requerimientos a cumplir en el proyecto y trabajando casi exclusivamente sobre el producto y con un diseño mínimo de documentación.

Las metodologías robustas o tradicionales tienen como objetivo llevar una documentación completa del desarrollo cumpliendo el plan de proyecto.

4.1.1 Ciclo de Vida de Software

Gracias a las fases de desarrollo (Definición, análisis, desarrollo, pruebas, despliegue y mantenimiento. Estas fases deben ser documentadas para gestionar e integrar aspectos de seguridad.

Cada ciclo debe ser compuesto como lo demuestra la figura 3, inicialmente debe existir una estrategia, una definición y análisis, después de estos procesos inicial la fase de desarrollo, las pruebas y por último el despliegue.

Figura 3. Ciclo de vida seguro de desarrollo de software

Fuente: PEREIRA.Dani.Desarrollo seguro de software. ISO 27001 [En línea] disponible en: https://itfortechies.wordpress.com/2019/08/30/desarrollo-seguro-de-software-iso-27001/

- Formación y estrategia hace referencia a la importancia del conocimiento en el equipo de desarrollo, es importante instruir al grupo en conceptos de seguridad, buenas prácticas.

  El equipo de trabajo debe conocer los conceptos de seguridad, conceptos de diseño y modelado de amenazas.

- Definición y Diseño donde debe ir acompañado de la documentación, el levantamiento de características describiendo cada proceso de la siguiente manera:
  - Se identifican y se definen los requisitos
  - Confirmación de requisitos de seguridad
  - Modelación de amenazas
  - Definición de ataques
  - Diseño seguro

4.1.2 ¿Qué es OWASP?

Open Web Application Security es una fundación sin ánimo de lucro, fomenta el desarrollo de herramientas para evaluar seguridad en software de aplicaciones web; tiene como objetivo la mejora de la seguridad del software. Esta fundación permite encontrar herramientas, cursos de capacitación, comunidades de diversos temas de desarrollo como son: Seguridad en aplicaciones Web, seguridad en aplicaciones móviles, reglas básicas de Owasp Modsecurity Conjunto de reglas para la detección de ataques a aplicaciones web, modelo de madurez de garantía de software, entre otras destacados. Las aplicaciones Web no son seguras por si solas, son el resultado de la búsqueda, gestión de políticas de seguridad tanto en versiones como en configuración. Esto es posible apreciarlo en la guía que ha realizado OWASP

para construir aplicaciones y servicios web seguros. En esta guía se analizan las arquitecturas dependiendo del tamaño del proyecto a desarrollar.

En este documento se clasifican los desarrollos en pequeños, medianos y gran escala.

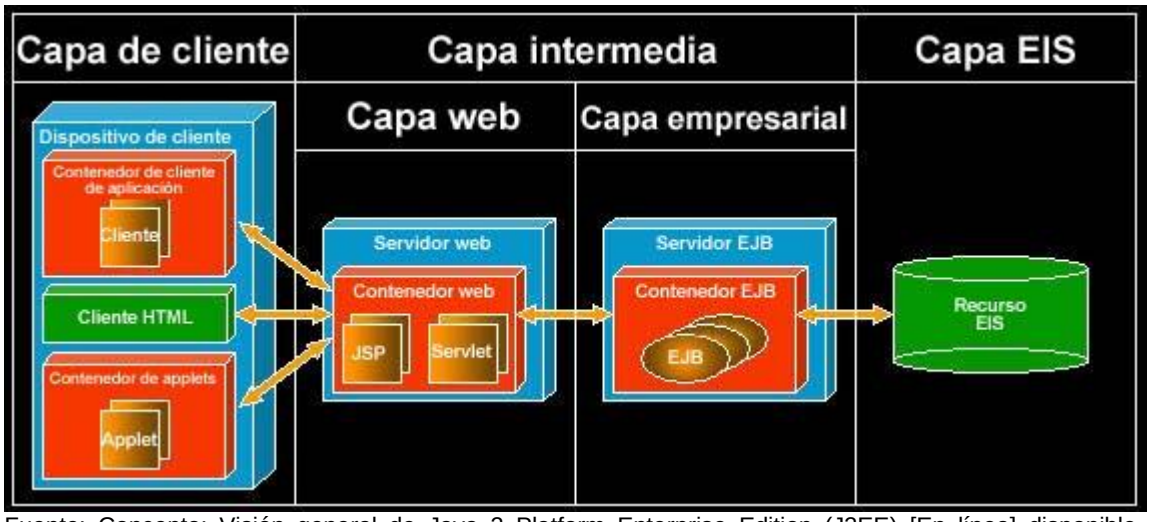
J2EE (java 2 platform, Enterprise Edition) brinda un estándar de desarrollo, un modelo de programación que mejora y estandariza el desarrollo de aplicaciones, escrita en lenguaje de programación java el cual se ejecuta desde un servidor de aplicaciones.
Esta arquitectura está basada en multiniveles.

Los niveles son:

- Nivel Cliente: Aplicaciones con java autónomas las cuales brindan interfaces dinámicas.
- Nivel Medio: Este nivel lo compone el servidor (Application Server), los servicios web los cuales encapsulan la lógica realizando acciones y almacenando en la base de datos.
- Nivel de Empresa: Esta almacenada en una base de datos relacional.

Las aplicaciones con J2EE realizan respuestas dinámicas, sus componentes alojan los servicios a los módulos web. Estas aplicaciones desarrolladas en J2EE son portables en servidores J2EE sin necesidad de modificar el código.
A continuación, son representadas en la figura 4.

Figura 4. Arquitectura de varias capas de J2EE

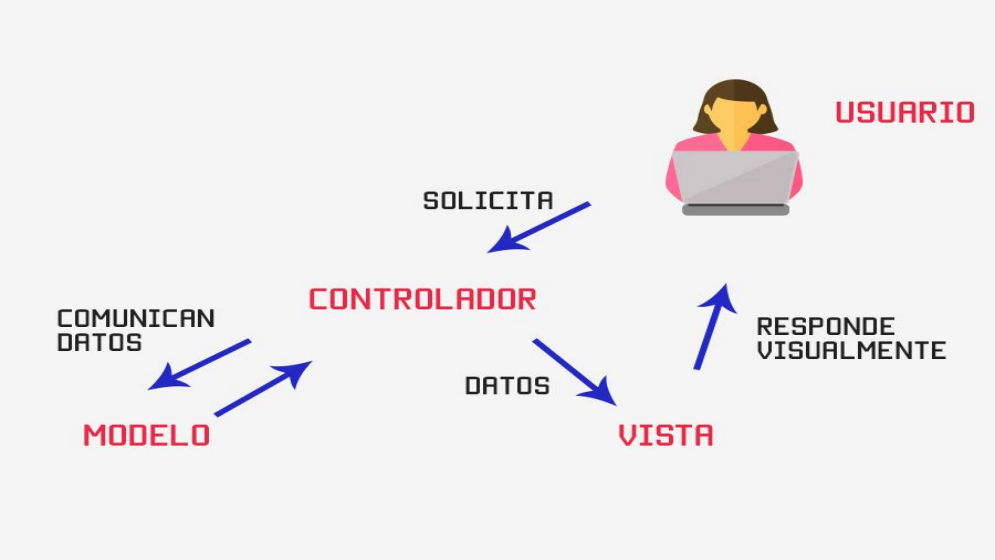


Fuente: Concepto: Visión general de Java 2 Platform Enterprise Edition (J2EE) [En línea] disponible en: https://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/tech.j2ee/guidances/concepts/java_2_platform_enterprise_edition_j2ee_overview_9A95BA45.html

Para las aplicaciones grandes o a gran escala se requiere de arquitecturas que permitan implementar y mantener funciones sin perder la calidad. Por lo general la arquitectura de aplicaciones escalables se divide en tres niveles para la reutilización de objetos; la división permite mejorar la capacidad de ajuste y respuesta.

Una de las arquitecturas más comunes es la de Modelo, Vista Controlador (MVC). Este esquema de diseño utiliza interfaces de usuario, lógica y control para separar la lógica de funcionamiento y su visualización (Figura 5).

Figura 5. Arquitectura MVC



Fuente: HERNÁNDEZ. Uriel. MVC (Model, View, Controller) Explicado [En línea] disponible en: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

- Modelo: Tiene la función de gestionar y actualizar los datos encapsulando los datos y comprobando los mismos.

- Vista: Presenta los datos al usuario o nivel de presentación llamado Front-end

- Controlador: Es encargado en dar toda la gestión a las instrucciones (procesa, las redirige y almacena) permitiendo la comunicación entre el modelo y la vista

4.2 MARCO CONCEPTUAL

Los conceptos fundamentales en el desarrollo de aplicaciones web son el SDLC que es el ciclo de vida del desarrollo de software; este tipo de distribución describe los procesos relacionados con el desarrollo, mantenimiento de una aplicación de software desde el levantamiento de requisitos hasta su implementación.
Su principal objetivo es evitar fallas o errores, por lo cual es necesario cumplir ciertos requisitos en cada fase para tener cierta garantía de seguridad. Los controles que se realizan pueden ser verificación de requisitos, administración de requisitos, control de versiones, informes etc. Hay que tener en cuenta que el ciclo de vida puede tener diferentes metodologías, pero una única norma ISO/IEC/IEEE 12207, esta determina una serie de procesos definidos, establecidos en común para precisar, controlar y mejorar los procesos del ciclo de vida del software.

CVE es una lista de información definida que es sostenida por The MITRE organización sin ánimo de lucro estadounidense, esta facilita investigaciones y soporte sobre tecnología informática al gobierno de Estados Unidad.

DEVSECOPS: Este método de desarrollo ágil combina el desarrollo con seguridad, su objetivo es automatizar y monitorear en todas las fases del desarrollo aplicando la seguridad.

Vulnerabilidades: debilidad que presenta algún sistema o aplicación, permitiendo al atacante ingresar sin autorización.

CVSS: Common Vulberability Scoring System (Sistema de vulnerabilidad común de puntuación) Es un marco de trabajo el cual comunica el nivel de gravedad de las vulnerabilidades, dándole a cada vulnerabilidad una puntuación según su importancia.

4.3 MARCO HISTÓRICO

La historia del desarrollo de software tiene como principios la lógica matemática que la cual permite el análisis lógico y el razonamiento matemático.

Gracias al razonamiento lógico es posible el planteamiento de algoritmos, el principio historio de los algoritmos y de la inteligencia artificial fue la máquina de Turing.

La segunda guerra mundial dio como principio a la encriptación de información, el inicio del lenguaje de alto nivel con la computadora z4 que llego hasta a su versión

z34; construida por el ingeniero alemán Konrad Zuse que había diseñado un almacenamiento magnético.[6]

Cada avance histórico desde la década de los años 50,60,70 y 80 han hecho cambios en la evolución del desarrollo de software y aunque su historia no es muy larga si contiene numerosos sucesos que han permitido su progreso.

4.3.1 Línea de Tiempo

La figura 6 presenta una línea de tiempo de los acontecimientos más importantes en la evolución de la historia del desarrollo de software.

Figura 6. Línea de Tiempo

[6] MERINO MARCOS, El primer lenguaje de programación de alto nivel data de la 2ª Guerra Mundial, pero no compiló ni una línea de código hasta los 70, junio 2022

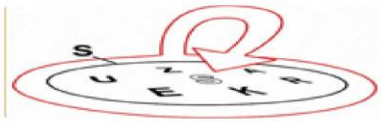
HISTORIA DEL DESARROLLO DE SOFTWARE

HITEHEAD – Principia Mathematica 1 +

1912

La Historia del Desarrollo de Software Inicia con Bertrand Russel y Alfred North con la presentación de la Lógica Formal o lógica matemática que estudiaba el razonamiento matemático implementado en análisis lógicos.

"Es una paradoja pregunta si, si S es el conjunto de conjuntos de todos los cuales no se tienen a sí mismos como miembro, S es miembro de sí mismo" 5



1930

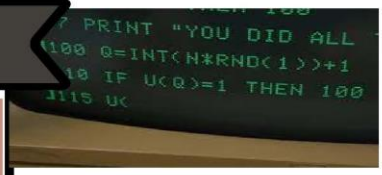
La Máquina de Turín fue unos de los inicios de la programación.

Dejo un gran aporte a la criptografía ya que realizo el análisis de los mensajes navales de los alemanes en la segunda guerra mundial.

1950

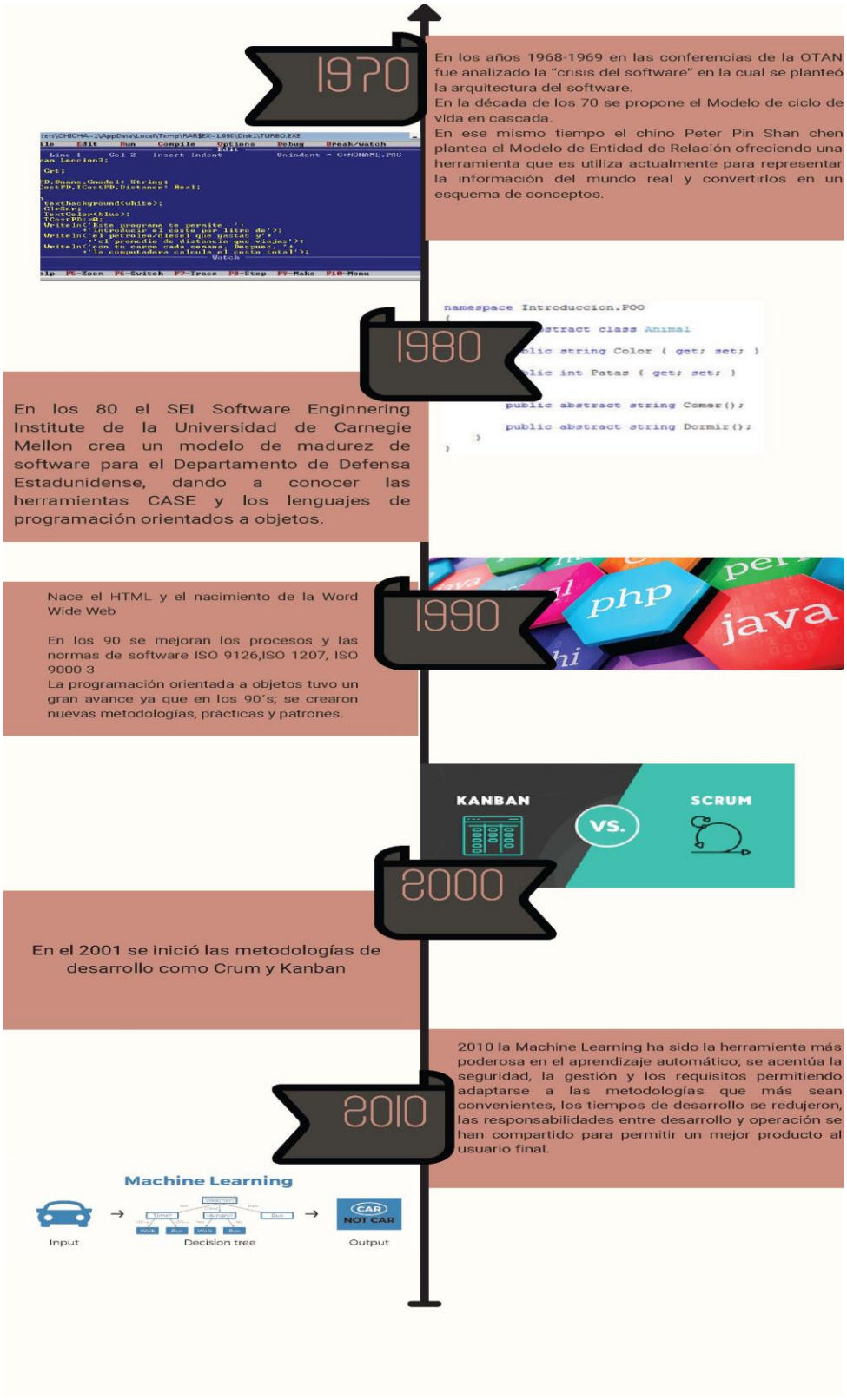
Alan Turing
Matemático de la Universidad de Cambridge, planteo algoritmos para soluciones de problemas, también planteo que hay problemas que no tienen solución. La máquina de Turing, la cual dejo todo su funcionamiento tanto teórico como operacional planteando problemas de parada o proceso infinito; su interés en la inteligencia artificial dejo como hecho la maquina computacional en 1950 la cual realizaba un test de turing.

1960



Los códigos de programación en los años 60 eran difíciles de interpretar llamados coloquialmente "Códigos espagueti".

En los años 50, 60 y 70 empezaron a desarrollarse leguajes como Fortran y SQL



Fuente: Propia

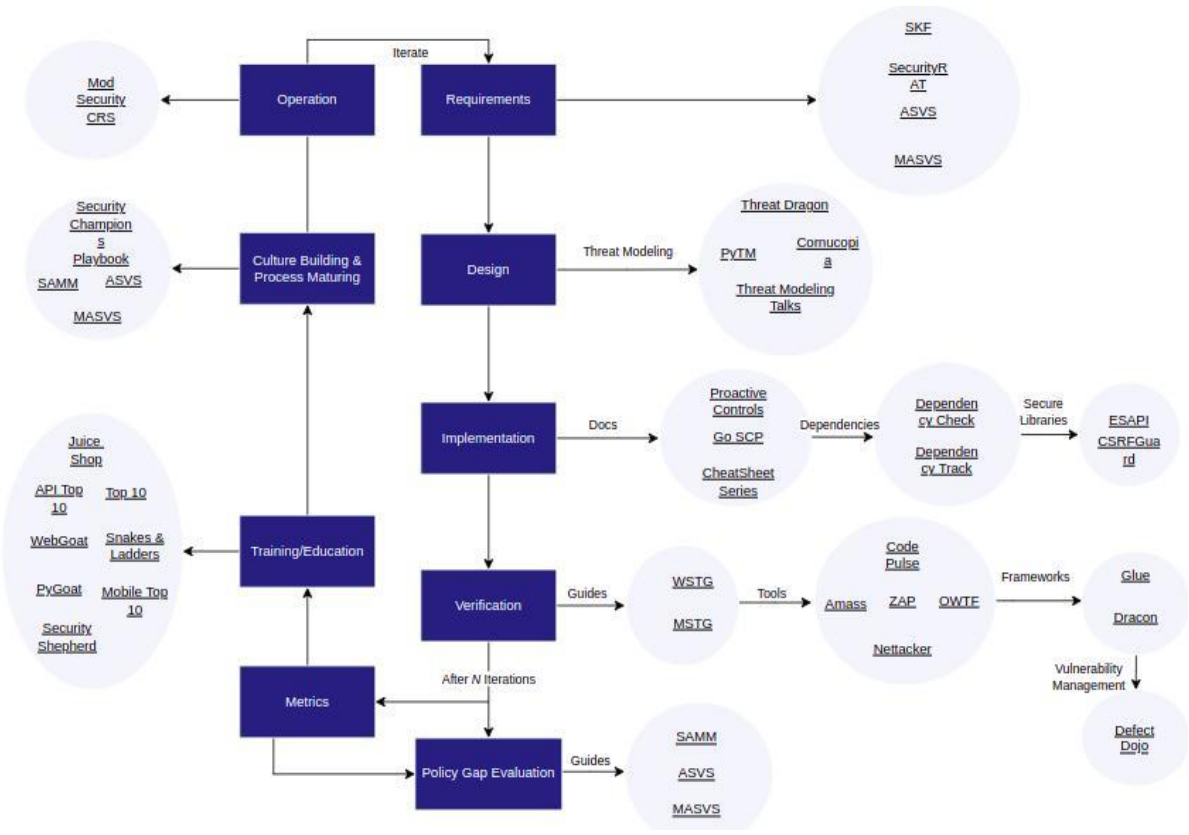
4.4 MARCO CIENTÍFICO O TECNOLÓGICO

Gracias a los avances que se han producido en los últimos tiempos en el desarrollo de software se han podido identificar las diferentes problemáticas que pueden presentarse desde la toma de requerimientos para cumplir objetivos o funciones y hasta la puesta en marcha o funcionamiento.

El proyecto de seguridad de aplicaciones web abiertas OWASP, ha trabajado para reconocer y tomar medidas de mejora o solución para encontrar las causas por las cuales el software es inseguro. Su comunidad AppSec que está conformada por organizaciones educativas, empresas y particulares que están en varias partes del mundo haciendo parte de reportes, opiniones, compartiendo documentación y brindado su conocimiento. Proponen un diagrama de Ciclo de Vida del Desarrollo de Software, para ayudar a los desarrolladores de software, arquitectos de seguridad, arquitectos de software o los que quieran profundizar en estos temas de seguridad; a conocer los estándares que se deberían considerar al momento de abordar un proyecto de desarrollo.

Los proyectos que integra OWASP se ven reflejados en el siguiente diagrama, el cual busca la seguridad ejecutando las diferentes aplicaciones, guías y demás herramientas de los proyectos OWASP en cada etapa del siclo de vida del desarrollo de software.

Figura 7 Diagrama de Ciclo de Vida del Desarrollo de Software según OWASP orientado a la seguridad



Fuente: OWASP [En línea] disponible en: https://owasp.org/projects/
Las etapas que presenta el diagrama del ciclo de vida del software se dividen en:

4.4.1 Requirements (Requisitos)

Estos requisitos son la base para el iniciar un proyecto, OWASP ofrece al usuario un entorno de aprendizaje en los lenguajes de programación más utilizados. Herramientas de verificación administrativa de seguridad, herramientas de verificación de requisitos dependiendo de los niveles de seguridad, guías y listas de verificación para aplicaciones web y aplicaciones móviles.

- SKF: es una aplicación web que brinda capacitación en python-flask/angular para la creación de aplicaciones web seguras. Sus principales servicios son la documentación, listas de verificación, plataforma de formación y los laboratorios que permiten la aplicación de pruebas a la configuración del código fuente.

- SecurityRAT: Herramienta que permite administrar las condiciones de seguridad. Define las características para establecer los requisitos de seguridad, dándole la importancia que sea necesaria según su estado.

- ASVS: Es un proyecto que permite realizar la verificación con listas de comprobación de requisitos y aplicaciones que permiten verificar la seguridad de las aplicaciones. Este es dividido en tres niveles dependiendo de la importancia de la aplicación. El nivel 1 de ASVS es para comprobaciones de seguridad de aplicaciones más bajo ya que es posible realizar por penetración casi sin necesidad de tener acceso al código fuente. El nivel 2 o standar, es diseñado para aplicaciones que contienen datos confidenciales o críticos como las aplicaciones que realizan transacciones de empresa a empresa, contienen datos comerciales críticos. Estas aplicaciones pueden ser atacadas por prácticas de explosión que encuentran las debilidades dentro de la aplicación. El nivel 3 es para aplicaciones más críticas o de alto valor que requieren un máximo de seguridad y verificación de la misma. Estas son aplicadas a organizaciones que realizan transacciones de alto valor, datos con un alto nivel de confidencialidad, funciones que no pueden ser interrumpidas, las aplicaciones de comprobación de seguridad del nivel 3 deben tener un buen diseño, una buena documentación. Su análisis debe ser más profundo, se deben hacer pruebas de separadas por módulos en todas las fases, físicas, de red, lógicas. Controles de integridad, autenticación, encriptación y manejo correcto de la carga.

- MASVS: Es un conjunto de guías de prueba y herramientas que permite la comprobación de la seguridad en aplicaciones móviles. Este proyecto incluye guías de pruebas, requisitos de seguridad que permite realizar la verificación tanto de la arquitectura como del diseño, listas de verificación, pruebas de seguridad. Cada guía de prueba se puede utilizar como recurso de aprendizaje, contiene procedimientos, tutoriales tanto del sistema móvil y técnicas avanzadas de ingeniería.

4.4.2 Design (Diseño)

En el diseño OWASP presenta herramientas de modelado, ayudas didácticas para identificar los requisitos de seguridad, al conocer las diferentes amenazas es posible abordarlas para formular soluciones.

- PyTM: Aplicación de modelado generando diagramas de flujo o diagrama de secuencia donde se describe las amenazas del sistema.

- Theat Dragon: Herramienta de modelado la cual es utilizada para crear modelos donde se puede describir las amenazas, mitigaciones y sus componentes.

- Cornucopia: Herramienta en forma de juego de cartas para identificar los requisitos de seguridad en el proceso de desarrollo ágil, convencional o

formal. Creado para el grupo de desarrollo donde se pueden recordar las validaciones, la autenticación, la gestión de sesiones, las autorizaciones y la criptografía.

- Thear Modeling Talks: Kit de herramientas de modelado la cual es explicada en una charla (Appsec California 2018) donde se describen los componentes básicos de los modelados de amenazas aplicándolo en el ecosistema de criptomoneda. Se analiza la seguridad en sus primeras fases, posibles ataques. Esta charla es ofrecida por Jonathan Marcil, Ingeniero de Seguridad de aplicaciones de Twitch, ex líder de Owasp Montreal.

4.4.3 Implementation (Implementación)

Al realizar la implementación OWASP ofrece una valiosa documentación donde se encuentran los listados de requisitos, listados técnicos de seguridad, repositorios de buenas prácticas y guías para desarrolladores donde se mencionan los errores más comunes en programación.

- Proactive: Es un listado de técnicas de seguridad que se deben incluir en cada proyecto de desarrollo de software, este listado cuenta con 10 items en el cual el primer ítem es el más importante (C1 Definir los requisitos de seguridad, C2 aprovechar las bibliotecas de seguridad, C3 acceso seguro a las bases de datos, C4 codificar con técnicas defensivas para detectar ataques, C5 validación de datos de entrada, C6 aplicar la equivalencia Digital (Representación del usuario y su proceso de autenticación), C7 cumplir los controles de acceso, C8 proteger los datos en todos los estados, C9 implementación de registros y monitoreo de seguridad, C10 manejo de errores y excepciones.

- Go SCP: Guía escrita en un libro de ayuda para desarrolladores donde describe los errores más comunes, aplicado a los que están aprendiendo a programar por primera vez en el lenguaje de programación GO (Lenguaje de programación concurrente creado por Google basado en C). También hace referencia a las prácticas de codificación seguras.

- CheatSheet Series: Es un proyecto donde se encuentran repositorios oficiales de OWASP, documentación de buenas prácticas de seguridad, archivos fuente, correcciones de código, códigos de prueba, nuevas propuestas de código, actualizaciones de códigos fuente, actualizaciones de hojas de referencia entre lo más destacado.

Dependencies (Dependencias)

Las dependencias son bibliotecas o paquetes utilizados cuando la aplicación o programa requiere su instalación para su funcionamiento.

- Dependency-Check: Herramienta de análisis de software que permite detectar vulnerabilidades que se han divulgado en componentes de código abierto. Estas vulnerabilidades por lo general son conocidas, al utilizar bibliotecas de terceros en aplicaciones propias son comunes las vulnerabilidades.

- Dependency-Track: Es una plataforma inteligente de análisis de componentes, realiza detección de vulnerabilidades, evaluación de políticas y análisis de impacto.

Secure Libraries (Bibliotecas Seguras)

Las bibliotecas seguras son un conjunto de funciones de código reutilizable previamente desarrollado que ya han sido probado y se encuentra listo para su uso.

- ESAPI CSRFGuard: Herramienta de seguridad gratuita que es muy conocida y popular, contiene un filtro JavaEE (Respuesta a una petición) que envía token de prevención sincronizado para mitigar los riesgos de falsificación de solicitudes en HTML. La última versión de esta herramienta permite él envió de solicitudes mediante Ajax (llamado a servidores desde el navegador sin refrescar la página), los token`s sincronizados de prevención son aleatorios correspondientes a la solicitud.

4.4.4 Verification (Verificación)

Las verificaciones son todos los procesos de comprobación y análisis para examinar que el desarrollo de la aplicación web cumpla con los especificaciones y necesidades del cliente. OWASP presenta guías de prueba, herramientas de verificación tanto del código como de red y diferentes pruebas de seguridad.

Guides (Guías)

- WSTG: Recurso de pruebas de seguridad para aplicaciones y servicios web.
- MSTG: Guía de pruebas para aplicaciones móviles, esta contiene casos de prueba, técnicas y herramientas para identificar que tan segura es una aplicación móvil.

Tools (Instrumentos)

- Amass: Esta herramienta permite realizar el mapeo de redes, detectando las posibles amenazas, la aplicación utiliza técnicas de reconocimiento activo y recolección de información de fuentes abiertas.
- Code pulse: Esta herramienta permite ejecutar en tiempo real un monitoreo
- donde recopila información sobre la actividad del código para identificar que petición hace y cuando, aunque es una aplicación de escritorio está enfocada actualmente a prueba de aplicaciones web.
- ZAP: Herramienta gratuita de pruebas de penetración de código abierto, está diseñada para probar aplicaciones web. Es llamado proxy intermediario ya que que se cruza entre las peticiones de comunicación del navegador y la aplicación web interceptando los mensajes, puede cambiar el contenido del mensaje si es necesario para que luego siga su destino.
- Nettacker: Es una herramienta que recopila información ya que realiza un escaneo de vulnerabilidades, utiliza diferentes métodos y generando informes en diferentes formatos (HTML, JSON, CSV) estos informes son de aplicaciones y redes, descubre puertos abiertos, servicios, errores de configuración entre otras. Es posible ejecutarlo en línea de comandos.
- OWTF: Este proyecto realiza pruebas de penetración más eficiente, encuentra la vulnerabilidad, realiza verificación en un tiempo eficiente integrando las mejores herramientas.

Frameworks (Marcos de trabajo)

Son aplicaciones o conjunto de modelos que permiten un desarrollo de aplicaciones web de forma ágil ya que contienen modulo, librerías y funciones creada con anterioridad para su uso.

- Glue: Proyecto de conjunto de herramientas para realizar análisis de seguridad de forma automática.
- Dracon: Proyecto que contiene herramientas flexibles que se permiten ejecutar en código abierto, permite el escaneo y análisis estático.

Vulnerability Management (Gestión de vulnerabilidades)

- Defect Dojo: Herramienta creada por DevOps de código abierto, es una herramienta líder ya que permite el rastreo, identificación y seguimiento de las pruebas de seguridad que se ejecuten, reportando el registro de usuarios, orígenes de la solicitud, ingresos entre las características más destacadas.

4.4.5 Policy Gap Evaluation (Políticas de Evaluación de brecha)

Estas políticas según OWASP son orientadas a la verificación de seguridad de las aplicaciones web utilizando modelos, programas y guías.

Guides (Guías)

- SAMM: Modelo de Madurez de Garantía de Software, es un modelo descriptivo, permite realizar en las organizaciones un análisis de las prácticas y actividades de seguridad.
- ASVS: Proyecto Estándar de verificación de seguridad de aplicaciones, permite probar los controles técnicos de seguridad de las aplicaciones web proporcionando una lista de requisitos de desarrollo seguro.
- MASVS: Proyecto de Guía de pruebas de seguridad móvil OWASP, define unas guías de pruebas de seguridad donde se cubren los procesos del software, las técnicas y herramientas estándar para las aplicaciones móviles.

4.4.6 Training/Education (Formación / Educación)

OWASP presenta diferentes alternativas pedagógicas para la enseñanza de amenazas y riesgos de seguridad en las aplicaciones web y aplicaciones móviles.

- Snakes & Ladders: proyecto educativo que consiste en un juego de mesa para aplicaciones web, permite capacitar en seguridad de una forma más divertida, el juego contiene los controles, riesgos y es de código abierto.
- Top 10: Documentación de los 10 principales riesgos de seguridad en aplicaciones. Contiene los riesgos más críticos e importantes para las aplicaciones web.
- Mobile Top 10: Top 10 de los riesgos de seguridad más importantes en aplicaciones móviles, desde el 2015 se está presentado este reporte a nivel mundial.
- Juice-Shop: Es una aplicación de juegos utilizada para capacitar en seguridad, en esta aplicación se ve reflejado todas las fallas de top 10 junto con otra que son encontradas en aplicaciones reales. Contiene desafíos donde el jugador debe hacer explotar las vulnerabilidades, es de código abierto.
- API Top 10: Son las clasificaciones de las categorías de vulnerabilidades y riesgos de seguridad que presentan las Interfaces de Programación de Aplicaciones (API).
- WebGoat: Es una aplicación que permite realizar comprobación de vulnerabilidades en las aplicaciones en Java. Esta aplicación funciona en un entorno seguro de enseñanza interactiva. Es necesario cuando se realice la instalación en el ordenado y se ejecute el programa desconectarse de internet ya que la configuración que realiza la aplicación pude dejar expuesto el ordenador. Las técnicas aplicadas en este programa son exclusivamente para fines de aprendizaje.

- PyGoat: Este proyecto consta de una aplicación desarrollada en phyton, que está diseñada para ser vulnerable a ataques de seguridad, su intención es realizar pruebas de cómo afecta las 10 principales amenazas de OWASP y cómo evitarlos.
- Security Shepherd: Plataforma robusta de capacitación en seguridad de aplicaciones web y móviles, donde se encuentran herramientas y pruebas de penetración de aplicaciones móviles y de aplicaciones web, módulos de juego para practicar las técnicas vistas, herramientas de ejemplos reales de riesgo de seguridad. Puede ser utilizado tanto local por un solo usuario como por varios usuarios en un servidor. Se puede personalizar según su asignación de permisos (Administradores, usuarios), es posible ver el avanza y dificultad de cada usuario en cada clase.

4.4.7 Culture Building & Process Maturing (Construcción de cultura y proceso de maduración)

Son proyectos de enseñanza y comprobación que contienes modelos, guías y juegos didácticos que permite reforzar el conocimiento de los patrones de seguridad.

- ASVS: Proyecto de verificación de Estándares de Seguridad de Aplicaciones de OWASP, es posible realizar la comprobación de controles técnicos de seguridad de aplicaciones web, revisar las listas de requisitos seguros para desarrolladores las cuales se encuentran disponibles en CSV y JSON, se puede verificar los flujos lógicos como la autenticación, la administración de sesiones y el control de acceso.
- MASVS: Guía de pruebas de seguridad para aplicaciones Móviles, cubre procesos, técnicas y herramientas utilizadas en la comprobación de seguridad. También contiene listas de verificación de seguridad que incluye el alcance de cada requisito.
- SAMM: Modelo de Madurez de Software Assurance, donde se realiza seguimiento en todo el ciclo de vida del software, se implementan prácticas de seguridad donde se define la madurez del software, evalúa la seguridad actual de software, verifica los objetivos, define la ruta de implementación y proyecta consejos de implementación.
- Security Champions Playbook: Plataforma de libro de juegos compartida por github donde se describe las principales características para establecer un programa de Security Champions. En cada capítulo del libro de juegos se estudia las recomendaciones generales, comentarios de los jugadores y enlaces de fuentes.

4.4.8 Operation (Operación)

Es la puesta en marcha de las aplicaciones las cuales deben cumplir con ciertos requisitos de seguridad para garantiza su funcionamiento e integridad de los datos.

- Mod Security CRS: Es un conjunto de reglas básicas de OWASP que permite la detección de ataques incluidas las mencionadas en el top 10 OWASP; está compuesto de guías, tutoriales y manuales para uso de la aplicación donde se ha mejorado a CRS 3 que incluye una reducción en alertas falsas, análisis de SQLi /XSS, reglas de inyección de código Java y PHP.

4.5 MARCO LEGAL

El software o aplicaciones desarrolladas a la medida están sujetas a derechos civiles, cumplen normas de propiedad intelectual.

También está sujeta a protección legal como de copyright, autorización de reproducción. Depende del cliente y de los derechos que se le dé al producto o al autor.

Legalmente se encuentran normas que cubren los riesgos de suplantación de identidad según la **Ley 1273 de 2009 Articulo 269G**: Suplantación de sitios web para captura de datos personales.

Acceso a un sistema sin autorización es castigado con el **artículo 269A de la misma Ley 1273 del 2009**; esta describe: "Acceso sin autorización en un sistema informática protegido o no o con una medida de seguridad, o se mantenga dentro del mismo en contra de la voluntad de quien tenga el legítimo derecho a excluirlo, incurrirá en pena de prisión de cuarenta y ocho (48) a noventa y seis (96) meses y en multa de 100 a 1000 salarios mínimos legales mensuales vigentes." [7]-[8]

En resumen, la **Ley 1273 de 2009** castiga además la obstaculización de un sistema informático o red de comunicaciones, la interceptación de datos, el daño informático (borrar, deteriorar etc.), usar software malicioso o software que genere daños, violación de datos personales, robo por medio informático, trasferencia de activos no autorizada. Todos estos delitos son castigados con multas o presión (el delito de intercepción de datos y violación de datos personales es castigado por entre mínimo 36-48 meses o 72-96 meses de cárcel)

Las metodologías permiten tener una visión que puede valorar las variables con la seguridad.

7 BERTRAND RUSSELL, La paradoja de Russell, Inglaterra 1903
8 DIARIO OFICIAL, Normatividad Ley 1273 de 2009. Colombia. 2009 p1.

Existen técnicas de recolección de datos según los estándares de metodologías a aplicar, estos deben ser llevados por escrito en documentos o bitácoras de trabajo.

El SDLC es un proceso metódico para la creación de software la cual permite darle calidad y permitir realizar correcciones al software desarrollado. Este proceso implica siete fases de proceso o ciclos de vida de desarrollo

- Definir las exigencias
- Planificación del Concepto
- Realizar el Diseño
- Desarrollo y Pruebas
  - Puesta en Marcha
  - Operaciones y Mantenimiento
  - Disposición

Para el desarrollo de un proyecto de software es necesario identificar primero los problemas más comunes tanto en el análisis, la planeación, desarrollo de la aplicación, pruebas y puesta en marcha.

5 AMENAZAS RECIENTES QUE AFECTARON LA SEGURIDAD EN LAS APLICACIONES WEB.

Desde que se inició la cuarentena por el covid-19, los trabajos que se realizaban en oficinas frente a un ordenador se trasladaron a los hogares, generando amenazas de ataques en algunas organizaciones. Los atacantes se han colado por el protocolo de escritorio remoto con técnicas como el relleno de credenciales o su ya tradicional phishing de ingeniería social para captar algún curioso que le suele dar clic a mensajes desconocidos.

Existen diversos ataques que pueden arremeter contra una aplicación web como es el **spoofing** que consiste en un taque de suplantación, **tampering** ataques a la integridad (modifican información), ataque de **Information Disclosure** (este ataque divulga información confidencial sin permiso) como el sucedido el 4 de mayo según el artículo del sitio DW Made for Minds FEW (EFE, AFP, El Tiempo) [9] con el ataque de Anonymous, realizado a la página del Ejército de Colombia el 5 mayo del 2021, publicando en redes sociales las contraseñas de correos electrónicos de 168 militares. Ataque de **Denial of Service** (Negación de servicio) denegación o disponibilidad de los servicios como el sucedido el 9 de mayo del 2021, documentado en el artículo digital BBC NEWS Mundo[10], narra el ciberataque que

9 DW Made for Minds. FEW (EFE, AFP, El Tiempo). Colombia: Anonymous se atribuye "hackeo" de la página del Ejército. Colombia. 2021

10 BBC News Mundo. EE.UU. declara estado de emergencia tras un ciberataque a la mayor red de oleoductos del país.EE.UU.2021

dejo en estado de emergencia a una red de oleoductos en Estados Unidos, impidiendo el servicio de administración del sistema para el trasporte de 2,5 millones de barriles de petróleo desconectando por completo cada oleoducto y robando más de 100 GB de información de los mismos; lo cual obligo a este país declarar el estado de emergencia. Ataque de **Elevation of Privilege** donde un usuario sin privilegios se vale de herramientas para asignarse privilegios de administrador.

Según una entrevista radial de Radio Victoria del 9 de abril del 2021 al especialista Javier Dieguez[11] Director del Centro Vasco de Ciberseguridad en España, dice que el volumen de ataques criminales ha aumentado cuatro veces el PIB del estado español. Las empresas y organizaciones se encuentran en permanente exposición las cuales a veces no reportan o no alcanzan a publicarse por riesgo de pérdida de confianza por parte de los usuarios; gracias a que las organizaciones públicas y privadas que están legalmente obligadas a reportar se han podido calcular el aumento de ataques, según el especializa asegura que pueden ser organizaciones especializadas para realizar espionaje con dos móviles espiar competidores o estados que rivalizan tanto militar o tecnológicamente. Dice que se ha observado ataques más frecuentes en tiempo de pandemia los cuales se observan como ataque de fraude, estafa y suplantación dirigidos a infraestructuras sanitarias, con el aumento del teletrabajo, la interacción a través de las redes sociales por la falta de interacción física hace que los ataque sean más frecuentes. Recomienda que la seguridad tenga una relación muy cercana con el desarrollo tecnológico y nuevas tecnologías las cuales debe también tomarse de forma positiva ya que permite un desarrollo humano-tecnológico. Recomienda que todas estas tecnologías deban proteger el software por posibles ataques.

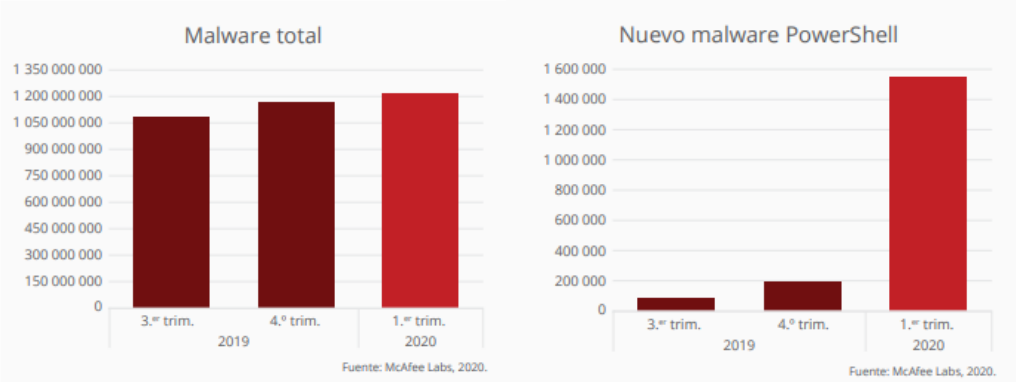
Concluye que no es sencillo rastrear a los responsables y que son organizaciones que tienen recursos económicos fuertes convirtiendo los ciberataques en una profesión ya que como los ataques no tienen fronteras en ocasiones no pueden ser penalizados. Es primordial reconocer los riesgos de la digitalización y divulgación web de las organizaciones y recomienda tomar las precauciones adecuadas en función de sus peculiaridades del negocio, adoptando la gestión de riesgos administrativos.

Según el informe de Macfee[12] sobre las amenazas- COVID-19, julio 2020, la estadística de amenazas de Malware se encuentra representada en la figura 7, describiendo el primer trimestre del 2020 con los siguientes aumentos:

- 375 amenazas por minuto en el primer trimestre del 2020
- Aumento la frecuencia de malwere PowerShell en un 689%
- Malwere en aplicaciones móviles aumento en un 71%

11 DIEGUEZ, Javier. "La digitalización es imparable, pero falta cultura de ciberseguridad". España. Radio Victoria. 2021
12 Mcafee. Informe de Mcafee Labs sobre amenazas-COVID-19. 2020

Figura 8. Estadística de amenaza de Malware



Informe de McAfee Labs sobre amenazas -COVID 19 julio 2020 [En línea] disponible en:https://www.mcafee.com/enterprise/es-es/assets/reports/rp-quarterly-threats-july-2020.pdf

Por motivos del contagio en la pandemia los puntos de trabajo han cambiado, los datos se encuentran en la nube lo que implica una mayor amenaza que ha aumentado en un 630%; Mcafee ha clasificado las amenazas en dos categorías:

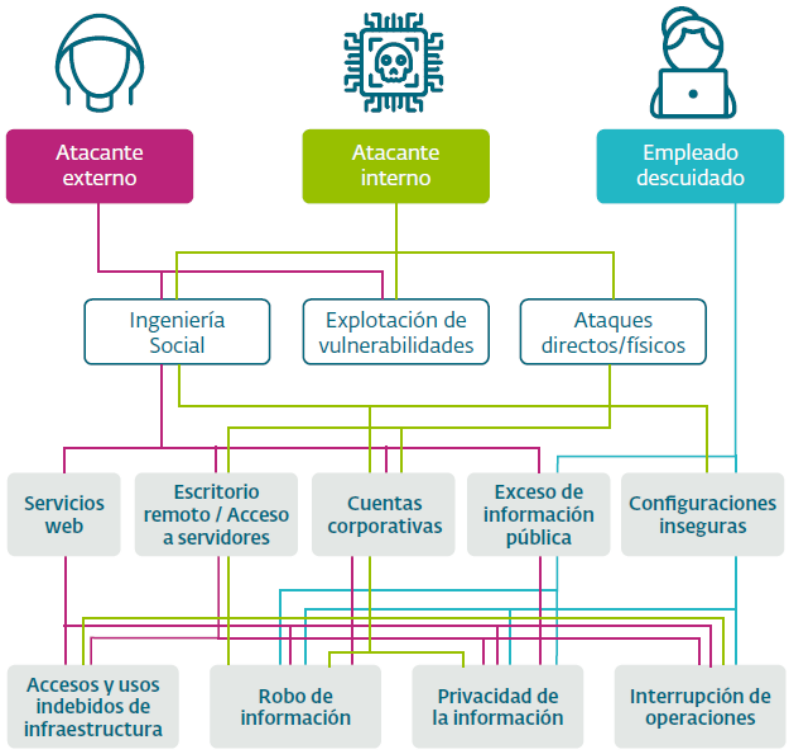
- Acceso de privilegios sin autorización
- Comportamientos sospechosos de inicio de sesión desde distintos puntos geográficos, este hecho se evidencio en los servicios colaborativos de Microsoft 365.

La recomendación que realiza Mcafee es realizar una combinación entre medidas tecnológicas de protección y una buena educación para estar preparados a las diferentes amenazas y cambios tecnológicos.

Analizando el reporte de la de ESET Latinoamérica del 2020 realizado a diferentes empresas donde se demuestra un comportamiento muy similar, el 70% no contaba con un plan de respuesta ante una pandemia. Esta información fue recolectada a diferentes empresas en Latinoamérica en 14 países de la región. Los encuestados coinciden en que lo más preocupante es el acceso no autorizado a la información confidencial. Por lo cual la variante más común de esta amenaza son los malware. Debido a las necesidades de seguir con las actividades laborales de las organizaciones en el momento de iniciar la pandemia fue fundamental realizar una trasformación digital para suplir las necesidades tanto de sus trabajadores como de sus clientes y todas las tareas que se desarrollan en un ambiente laboral. Este escenario hizo que se desarrollaran medidas tanto de buenas prácticas como de herramientas y planes de contingencia. Hay que ser realistas la gran mayoría de las organizaciones que contaba con un plan para este tipo de situaciones lo cual hace que se planteen nuevas medidas tanto en capacitación, concientización e implementación de mejores tecnologías para proteger los datos.

Según el laboratorio de investigación de ESET, Figura 8. Donde se demuestra al atacante tanto interno como externo, tipo de empleado, los servicios víctimas y efectos de estos ataques.

Figura 9. Diagrama donde se demuestra los medios donde se ejecutan los ataques Investigación hecha por Laboratorio de Investigación ESET



Fuente ESET Security Report Latinoamérica 20202 [En línea] disponible en: https://empresas.eset-la.com/archivos/novedades/87/ESET-security-report-LATAM2020.pdf

Gracias a las circunstancias de la pandemia se ha acelerado el proceso de desarrollo digital, abriendo nuevos escenarios de trabajo, permitiendo realizar comprobaciones tanto de herramientas como de metodologías. Valorando cada actor y componente que interacciona con las aplicaciones y los servicios tecnológicos.

En el diagrama de ESET se puede analizar que la ingeniería social y la explotación de vulnerabilidades son uno de los factores más insistentes y principales que pueden afectar la integridad de la información; aprovechándose de las vulnerabilidades tanto de software como de hardware, la poca capacitación del personal que integra la organización se refleja en la afectación de los servicios que se prestan.

En la figura 9 se visualiza el informe de los incidentes presentados en los países que conforman Latinoamérica, el 60% de las empresas han sufrido por lo menos alguna eventualidad de seguridad; de cada una de tres empresas sufrió ataque de algún tipo de código malicioso y ransomware.

Figura 10. Empresas en Latinoamérica con incidentes de seguridad

Fuente ESET Security Report Latinoamérica 20202 [En línea] disponible en: https://empresas.eset-la.com/archivos/novedades/87/ESET-security-report-LATAM2020.pdf

La estadística arrojada por ESET demuestra que en las empresas en Latinoamérica el 98% tiene algún tipo de control tecnológico, el 39 % de las empresas no han adoptado políticas de seguridad, el 18 % asegura haber tenido en un instante del año 2020 de un acceso no autorizado a su información. Un dato que es muy significativo es que el 50% de los ataques a vulnerabilidades (en aplicaciones, redes, sistemas operativos, sitios web etc.) se centraron en países como México, Perú y Colombia.

Los códigos maliciosos más comunes son los que se utilizan para robar datos, otros suplantan sitios para robar las credenciales de autenticación, otros códigos maliciosos se encargan en infectar sistemas operativos para duplicarse y permitir su acceso a diferentes archivos, partes del sistema y medios de almacenamiento.

5.1 ¿CÓMO SE HA HECHO EL CONTROL Y PREVENCIÓN DE ESTOS RIESGOS?

Las empresas en Latinoamérica han adoptado políticas de seguridad, se siguen usando herramientas de primer nivel el cual es el antivirus como medida básica de protección. Los firewalls y las copias de respaldo de información también son utilizadas, aunque sorprende que un 48%[13] cumple en su totalidad sobre estas medidas de protección.

Los ataques han venido evolucionando igual que las medidas de protección como el EDR (Detección y respuesta de amenazas los cuales son herramientas que pueden monitorear la red, contralar las aplicaciones y dar respuestas a incidentes), autenticación de múltiples factores (el usuario tiene acceso si aprueba dos o más pruebas de su identidad). Estas nuevas medidas no ha sida implementadas en su totalidad en la región, se presenta una implementación de estas medidas entre el 17% y 18%.

Por otro lado, la gestión de implementar políticas de seguridad ha tenido una gran acogida.

Según el reporte de ESET para Latinoamérica del año 2020 la implementación de políticas de seguridad ha acumulado un porcentaje del 61 %[14] (según las empresas encuestadas). En el reporte del 2021 también con ESET para Latinoamérica aumento el 68 %[15].

Los niveles de implementación de políticas de seguridad en países como, Argentina, Brasil, Chiles y Colombia llegan a un promedio de entre 60% y 70% y los países con más baja implementación son Paraguay, Panamá y México entre un 38% y 50%.

[13] ESET. Security Report Latinoamérica 2020.
[14] ESET. Security Report Latinoamérica 2020
[15] ESET. Security Report Latinoamérica 2021

Los estudios y encuestas demuestran que es preocupante que las empresas no tengan plan de respuesta y continuidad del servicio en caso de un ataque que sea perjudicial, solo 33% si tiene un plan de contingencia. Esto demostraría que la falta de planeación e inversión no está considerada como una medida de control de riesgos.

La seguridad no se puede limitar a solo controles de tecnología requiere una combinación de controles administrativos, físicos y técnicos.

6 EVIDENCIAR LAS DIFICULTADES Y MEJORAS EN LAS FASES DE CONFIGURACIÓN Y CODIFICACIÓN EN LAS APLICACIONES WEB

Las aplicaciones web son programas que son accedidos desde la web, generalmente son almacenados en servidores; son usados por diferentes organizaciones o empresas para interactuar con más frecuencia y cercanía con sus clientes.

Técnicamente el programador debe tener el dominio del proyecto a desarrollar por lo cual está limitado a cumplir los objetivos por causa de tiempo restringido, falta de conocimiento en seguridad y buenas prácticas se llegan a cometer errores a la hora de desarrollar un proyecto.

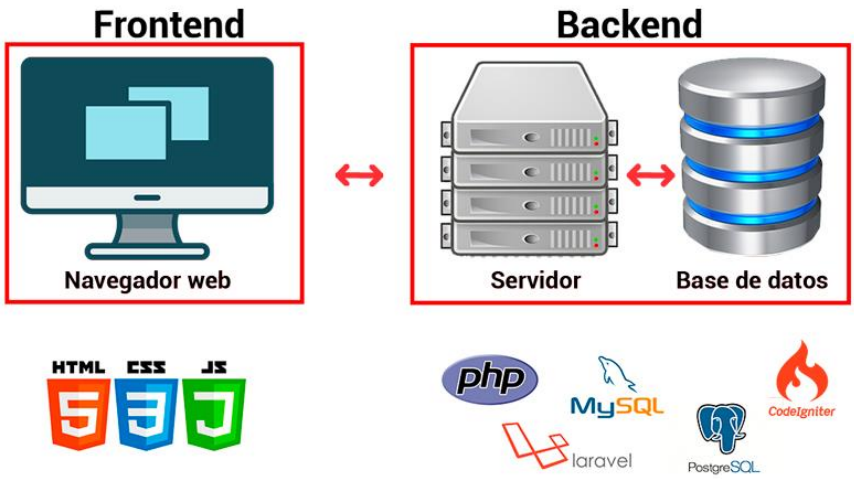
Se debe recordar y tener siempre presente que los ataques que pueden producir mayor al servidor donde se administran, recopilan y se procesan los datos.

6.1 FASE DE CONFIGURACIÓN:

Para que un sitio o aplicación web funcione correctamente es necesario combinar ambas tecnologías frontend y backend; estos entornos trabajan en diferentes áreas del desarrollo, utilizando diferentes herramientas, utilizando diferentes lenguajes y requisitos diferentes.

En la figura 10 se muestra que ambos entornos realizan funciones diferentes, Fronten con un entorno más visual para el usuario y Backend que permite toda la interacción del código, configuración y manejo de la base de datos.

Figura 11. El Backend y el Frontend



Fuente que-es-el-backend-y-que-es-el-frontend [En línea] disponible: https://www.suratica.es/que-es-el-backend/que-es-el-backend-y-que-es-el-frontend/

6.1.1 BACKEND (LADO DEL SERVIDOR)

Las actividades realizadas en el Fronend y el nivel de acción son configuradas en el Backend, las malas configuraciones pueden llevar a un mal funcionamiento y exposición de datos que no deben ser divulgados. La falta de conocimiento, o falta de atención a la hora de configurar puede ocasionar ingresos no deseados. A continuación, se mencionan las configuraciones que presentan más error.

- Se recomienda guardar la configuración en variable de entorno como las conexiones a la base de datos, esta información cambia dependiendo del ambiente (base de datos de producción,). No es bueno guardar las variables de entorno en el código ya que es un riesgo de seguridad, el acceso debe ser restringido. La variable de entorno debe estar protegía ya que su acceso debe ser restringido.

- Asegurar las rutas del API, se debe dar acceso al usuario correcto validando la autenticación (token o firma cifrada que permite identificar el usuario). Validar la autorización así este autenticado (verificar permisos). En este tipo de configuración es posible ver configuraciones erradas de controles de acceso, las cuales están divididas por categorías según la importancia de los usuarios; esta configuración es almacenada en la Lista de Control de Accesos (ACL), la lista permite controlar los niveles de acceso y priorizar dependiendo de la necesidad, es importante realizar una correcta configuración y examinar con frecuencia que nivel se tiene de acceso apropiado y que usuario lo utiliza, si está vigente su labor o es necesario su bloqueo.

- Validar al lado del servidor: no se debe asumir que los datos de entrada son enviados por el cliente, pueden ser enviados por usuarios mal intencionados o inyección de datos; las validaciones pueden ser por campos requeridos, permisos, entrada de datos etc. La falta de verificación de origen de los datos al momento que se realiza alguna consulta, da como resultado el acceso a la información. Es una obligación que las aplicaciones no acepten peticiones no autorizadas sin realizar verificación.

- Mensajes de error con rutas de accesos exponiendo información, por lo general este error es visible en Frontend pero resulta que proviene de la incorrecta configuración de algún componente del software dando la ubicación real de los datos tanto de configuración, dirección del servidor y otros datos al público. Esto se previene verificando el leguaje de codificación.

- Es vital tener como prioridad la seguridad de cada componente de software, es justo verificar si ya es obsoleto o se encuentra en la recta final de su vida útil ya que es más susceptible a ataques; se debe verificar sus actualizaciones, comprobar las versiones y su vigencia para reducir los riesgos.

- Se recomienda tener cifrado las comunicaciones entre frontend y el backend para prevenir escuchas de comunicaciones. Este tipo de comunicaciones son los llamados del API (interfaz de programación de aplicaciones). Para garantizar la seguridad se debe usar comunicaciones en https, para evitar la captura de datos se debe utilizar SSL (Es un protocolo de cifrado) que permite un canal entre la red interna y el internet, en este caso protege la comunicación entre el navegador web y el servidor web, cambiando la dirección del sitio web de HTTP a HTTPS.

6.1.2 BASES DE DATOS:

- Restringir el acceso a la base de datos, es necesario configurar el acceso más seguro desde el origen conocido (configurar IP desde la aplicación a la base de datos).

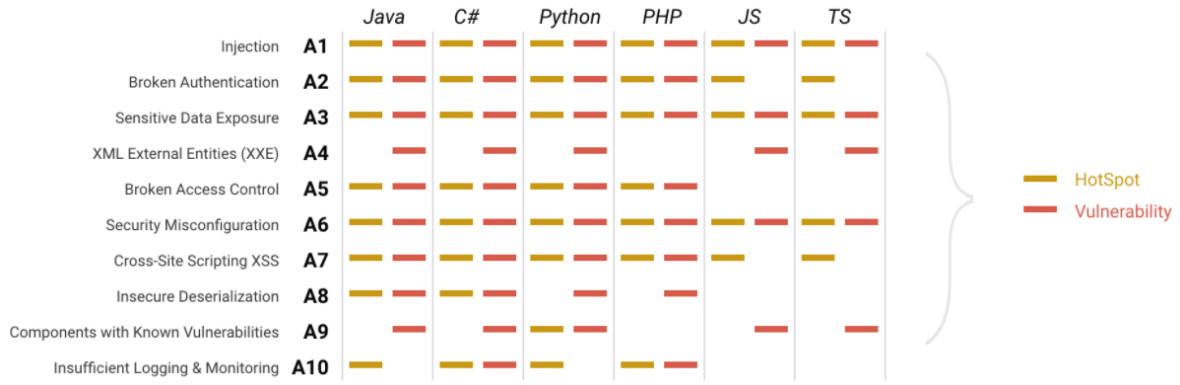
- No guardar información en textos claros (información comprometida). No guardar contraseñas, datos financieros donde el texto sea claro, se recomienda utilizar algoritmos Hashing especializados (codifica o trasformado en cadena de caracteres).

6.2 CODIFICACIÓN

Se presenta los siguientes problemas descritos en el top 10 de OWASP en los cuales se visualiza las diez problemáticas más frecuentes en los lenguajes de programación más utilizados en este momento.

En la figura 11, se muestra en la parte vertical los diez principales riesgos de seguridad de las aplicaciones web para la vigencia 2017. Al lado izquierdo se presentan los principales riesgos según OWASP en su top 10 dirigido para desarrolladores en los cuales se evidencia:

Figura 12. Problemas en las 10 categorías de riesgo de seguridad más críticas en sus aplicaciones web en 2017



Fuente OWASP Top 10 - We've got you covered! [En línea] disponible:https://www.sonarqube.org/features/security/owasp/?gads_campaign=South-America-Generic&gads_ad_group=OWASP&gads_keyword=owasp%20top10&gclid=Cj0KCQjw_fiLBhDOARIsAF4khR1ngeie4WrYobpvKVcBUzZN__p7m7tkSvWK_Jx-XTWhPbLTDj_pL7kaAkUiEALw_wcB

6.2.1 A1. La Inyección

Se presentan cuando se envían datos que son dudosos, estos son trasportados por medio de comandos o consultas. Estos datos pueden adulterar los datos sin que se evidencie. Las aplicaciones son vulnerables cuando no validan correctamente, estas son hechas desde un cliente de aplicación SQL, NoSQL, LDAP, XPath, comandos del sistema operativo entro otro estos son los más importantes; el cual puede leer los datos de la base de datos en el caso de SQL y NoSQL. Puede revelar la información, eliminación o daño de los datos.

Como se presentan estas vulnerabilidades:

Por medio de consultas no parametrizadas, consultas dinámicas que no se encuentran codificados los parámetros.

Como prevenirlas colocando en las consultas parámetros, validando datos de entrada del servidor, utilizar LIMIT y controles de SQL dentro de cada consulta.
Buscando API seguras. Correr herramientas que realicen análisis de las consultas a la base (verificación de relación de objetos).

En la figura 10 se visualiza las aplicaciones que tienen posible riesgo al momento de un ataque con inyección. Al analizar la gráfica (figura 10), todas las aplicaciones mencionadas tienen punto de acceso y son vulnerables, el código que puede ser frágil, aunque esto no quiere decir que este errado. Es necesario realizar un análisis con alguna herramienta que pueda identificar posibles riesgos; permitiendo la detección de código mal ejecutado o corrupto.

6.2.2 A2. Pérdida de Autenticación.

Este incidente se da cuando los controles de autenticación no son suficientemente fuertes; (las contraseñas son muy débiles las cuales pueden ser identificadas con facilidad, reutilizadas o conocidas y fáciles de descifrar). Los controles de restablecimiento de contraseñas no cumplen con los estándares de seguridad.
Las contraseñas almacenadas no cuentan con un correcto método de cifrado.
Los controles de seguridad a la hora de la autenticación son pobres (autenticación multi-factor), tiempo de sesión, validación de cierre de sesión.

Como se previenen:

- Utilización de autenticación multi-factor, no utilizar credenciales conocidas o por defecto en los perfiles de administración, estructurar contraseñas fuertes con signos, letras mayúsculas, números y longitud. Emplear cierta caducidad o tiempo de vigencia de contraseñas, programar la aplicación para detectar cierta cantidad de intentos fallidos ya que pueden ser ataques de fuerza bruta.
- Configurar cada cierre de sesión según el tiempo de inactividad.

- Las aplicaciones que presentan posibles vulnerabilidades por este ataque son Java, C#, Python y PHP. JS y TS son Hotsot

6.2.3 A3. Exposición de Datos Sensibles.

Este ataque se presenta con el robo de credenciales, textos planos o tráfico entre el cliente y el servidor para descifrar la información y encontrar las contraseñas o información importante. Esto sucede porque no se cifran de forma correcta los datos con protocolos débiles.

Como prevenir: Ubicar la información que es de un valor significativo y clasificarla según su valor implementándole controles de seguridad según su importancia.
Los datos importantes deben ser cifrados con protocolos seguros.
Los protocolos de comunicación deben cumplir con estándares de seguridad según la importancia de los datos.

6.2.4 A4. Entidad Externa de XML.

Cuando los procesadores XML tienen una configuración no adecuada lo cual hace que la documentación XML sea víctima de códigos vulnerables; permitiendo la extracción de datos, las solicitudes remotas y denegación de servicios.

Para prevenir esta vulnerabilidad se debe utilizar formato de intercambio de datos que evite la serialización (proceso que convierte un objeto en un grupo de bytes) como el Json. Actualizar las bibliotecas XML, comprobar la carga de archivos XML o XSL. Utilizar herramientas que puedan detectar XXE (XML External Entity). Validación de entrada al servidor, realizar actualizaciones de parches de seguridad, verificando la funcionalidad de los archivos XML o XSL validándolos a la entrada.

6.2.5 A5. Pérdida de control de acceso.

Las configuraciones de los usuarios que inician las sesiones no son correctas permitiendo que tengan acceso a datos no autorizados, o los perfiles no son los adecuados. Este hecho conlleva a la utilización del control del acceso en las cuales no es posible detectar si son correctas. No es posible comprobar el control de acceso, los privilegios pueden ser superiores en los usuarios estándar; esto permite que los datos puedan ser maniobrados.

Como prevenirlos: Estas configuraciones debe ser realizadas en el backend (del lado del servidor), se debe crear módulos de control de acceso los cuales debe tener ciertos requisitos, es necesario realizar las pruebas al control de acceso antes de su implementación.

6.2.6 A6. Configuración de Seguridad Incorrecta.

Cuando se realiza la configuración manual da más posibilidad de error, no se configura correctamente las actualizaciones y parches, falta de habilitar servicios o servicios innecesarios. Los sitios web que se abandonan por falta de soporte o uso,

archivos que se encuentran desprotegidos. Las configuraciones incorrectas pueden darse tanto en las aplicaciones, en los servicios, en la red, en el servidor y en la base de datos.

Como prevenirlo, se deben utilizar entornos de seguridad que permitan la verificación de las configuraciones de servicios.
Deshabilitar funciones innecesarias, actualizar o parchar las configuraciones, verificar permisos y separación de ambientes (Desarrollo y producción).

6.2.7 A7. Comandos en sitios cruzados.

Cuando se envían datos sin validar los cuales pueden ocasionar perdida de datos, secuestro de sesiones.

Se utilizan datos no confiables, es posible que el atacante robe datos de sesiones y apropiándose de la cuenta. Cuando no se validan datos ni son evacuados pueden ser usados por atacantes para robo de sesiones. Cuando los datos de sesiones son almacenados en API no seguras.

Como prevenir: Separar los datos, utilizar un frameworks que codifique los contenidos, codificar datos http en los campos de salida html, implementar políticas de seguridad.

6.2.8 A8. Deserialización Insegura.

Cuando se reciben datos que contienen códigos maliciosos en los cuales se permite perdida de privilegios, acceso a los datos, inyección de datos o pérdida de control del equipo (accesos remotos), afectando servicios, protocolos, base de datos y sistemas operativos.

Como prevenirlo: No permitir el acceso de objetos serializados (codificación de un objeto con el fin de trasmitirlo) de fuentes desconocidas.

Efectuar la verificación de integridad, verificación del tipo de objeto. Aislamiento de la deserialización y ejecución de la misma con privilegios restringidos. Monitoreo de este proceso.

6.2.9 A9. Componentes con Vulnerabilidades Conocidas.

Las vulnerabilidades pueden dar puerta abierta a accesos no autorizados, perdida de datos, modificación de privilegios, pérdida de control del equipo, secuestro de información. No tener conocimiento de las versiones de los componentes que se utilizan puede llegar a ocasionar vulnerabilidad en el software, tanto de sistemas operativos, aplicaciones web, componentes, servidores, APIS, ambientes de desarrollo etc.

Como prevenirlo: Utilizar herramientas que busquen vulnerabilidades, verificar las bibliotecas o parches, verificar si existen mantenimiento o versiones recientes, instalar actualizaciones tanto del sistema operativo como de las aplicaciones.

6.2.10 A10. Monitoreo Insuficiente.

Falta de respuesta de incidentes, no tomar las medidas necesarias para proceder a una prevención o en caso de ataque. No se realiza un monitoreo periódico, los registros son guardados localmente, los registros de incidencias de las aplicaciones no son revisados con frecuencia, no existen auditorias sobre fallos.

Como prevenirlo: Establecer un plan de respuesta a posibles ataques,
Realizar verificación y control de auditoría, adoptar software de protección y de registro de alertas.

6.2.11 FRONTEND (lado del cliente)

El lado del cliente a veces no se le presta la atención que se debería, hay que recordar que el frontend es la puerta principal de la aplicación web que se pretende desarrollar donde se permite el ingreso a los visitantes. No se presta atención suficiente a esta parte ya que los datos son almacenados en el servidor (backend); el frontend tiene las llaves principales por lo cual es necesario su protección. También actualmente existen aplicaciones web que trabajan en plataformas sobre la nube (sin servidor), dando la responsabilidad de una buena configuración al frontend.

- No guardar información importante del lado del cliente (guardar en cookes, local storage). Almacenamiento web son almacenamientos fáciles de acceder.

- La simplificación: muchas variables o funciones son hecha de forma ilegible no tiene ninguna relación con la operación que se realiza. Se recomienda definir variables y funciones con nombres que hagan relación a la operación o función que van a realizar.

- Convenciones de Nombres: Los nombres de las funciones no son coherentes con el resultado de la operación. Es preciso tener un código con un leguaje más legible para que sea fácil de analizar por otros desarrolladores (se recomienda utilizar un solo idioma preferiblemente ingles).

- Funciones con muchos parámetros: No se recomienda listar varios parámetros o datos relacionados, es mejor agruparlos por su relación, creando estructuras de datos.

- Funciones sin tipos específicos: Es una mala práctica ya que se crean muchas excepciones o muchos casos para determinar qué hacer cuando se recibe el dato. Es mejor hacer funciones que trabajen el mismo tipo de dato.

- Anidaciones: Se presentan en las condicionales que validen los datos, en la cual la estructura es de triangulo con el código encapsulado. Es mejor separar y tratar de tener una sola condicional para validar los datos cuando se cumple la condición. Es más recomendable separar el código.

- Utilización de librerías de terceros se usan para agilizar los tiempos en el desarrollo, esto implica que tenga vulnerabilidad que no sepamos, para realizar la verificación de estas librerías OWASP tiene una aplicación que permite realizar un análisis de librerías como las de Java o .NET donde se generan reportes de vulnerabilidades.

Es importante que los equipos de trabajo tengan la capacitación suficiente al momento de abordar la codificación y configuración. Hay proyectos que no se arriesgan contratando ingenieros sin la experiencia ni el conocimiento esto con el fin de no tener retrasos en el cronograma o posibles fallas.

7 EXPONER LAS MAYORES VULNERABILIDADES PRESENTADAS DENTRO DEL CICLO DE VIDA DE DESARROLLO DE SOFTWARE EN APLICACIONES WEB SEGÚN LA GUÍA DE PRUEBAS OWASP V3.0

Es necesario implementar la seguridad en los procesos de desarrollo de software ya que solucionar las vulnerabilidades después de la implementación puede acarrear costos; entre más se tarde en detectar las vulnerabilidades más riesgo tiene la aplicación.

El ciclo de vida de desarrollo de software contiene todas las fases del desarrollo desde el inicio de la idea hasta su funcionamiento. Cada fase es un engranaje fundamental de la siguiente fase por lo cual cada fase tiene la misma importancia en todo el proceso.

Figura 13. Ciclo de vida de un software

Fuente: ÁLVAREZ. Gerardo 24 septiembre 2018. Ciclo de vida de un software [En línea] disponible: https://www.kyocode.com/2018/09/ciclo-de-vida-de-un-software/

El ciclo de vida de una aplicación web está constituido por una serie de procesos que permiten llegar a un producto deseado que debe cumplir con ciertos rasgos para ser aprobado.

Conforme al entorno de pruebas de OWASP estas son aplicadas en el ciclo de vida de desarrollo de software (SDLC), esta comprobación puede verificar las fases de definición, diseño, desarrollo, implementación y mantenimiento.

Se pretende asegurar e identificar las políticas, la documentación y normas adecuadas que se debe emplear en la documentación ya que estas son las directrices para el equipo de trabajo.

Según la guía de OWASP[16] para construir aplicaciones y servicios web se recomienda ejecutar el modelo de evaluación y monitoreo utilizado para comprobar el funcionamiento del negocio a nivel tecnológico. Gracias las diferentes herramientas de OWASP para identificar los riesgos junto con COBIT realizan un buen equipo en la revisión y control en las diferentes fases del ciclo de vida del software.

7.1 ANTES DEL DESARROLLO

Recolección los requerimientos, es donde se reconoce por primera vez el fin de la aplicación web. Que se debe cumplir, que requisitos puede tener en seguridad (una vista general: usuarios autenticados, almacenamiento de información vital etc).
Es de gran importancia el modelado de requisitos del negocio ya que permite diseñar la arquitectura de seguridad, reduciendo el área de ataque.

Es muy significativo meditar si como desarrollador fuera el atacante (cambio de roles) sería capaz de explotar el riesgo.

Según la gráfica 12 del flujo de entorno de pruebas de OWASP las principales características que más llaman la atención y donde se puede presentar más errores son en la verificación de políticas implementadas al proyecto (después de conocer los requisitos), la construcción de la documentación ya que en estos escritos se van a plasmar las normas para que el equipo de trabajo tenga una guía. En esta documentación se deben encontrar sucesos más frecuentes según la aplicación que se utilice en el proyecto.

7.2 DEFINICIÓN Y DISEÑO

Es importante tener claro las especificaciones de seguridad que debe tener el proyecto de desarrollo más si se va a realizar a la medida. Se debe considerar todas las posibilidades según su principio de funcionamiento y su misión.

Se debe asegurar una trazabilidad donde se evidencie todo el proceso en el desarrollo de la aplicación web. En ocasiones no se lleva como lo describe OWASP en su guía por limitación de tiempo, desorganización o falta de profesionalismos y ética del equipo de trabajo.

16 Fuente: Guía de Pruebas OWASP 2008 3.0 [En línea] disponible: https://owasp.org/www-pdf-archive/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

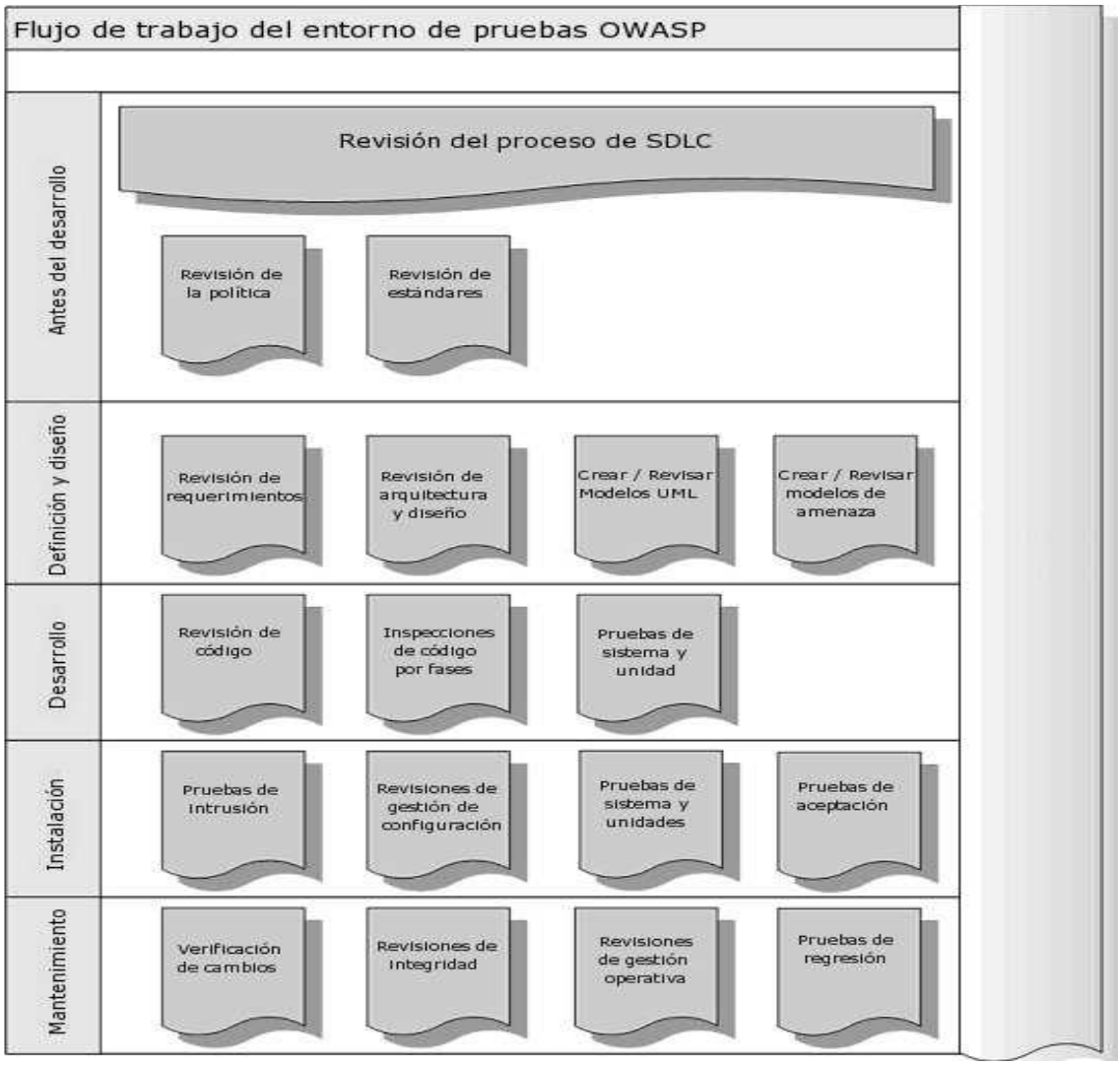
En el diseño se selecciona la arquitectura y el modelo según los requisitos a cumplir. En esta fase se pueden detectar más fácil los errores de seguridad ya que se tiene el modelo a mano donde se debe reflejar como va a funcionar la aplicación web.

En el ciclo de vida de desarrollo de software, la arquitectura sería el diseño donde se debe profundizar las políticas de seguridad. Según OWASP la arquitectura de seguridad es el pilar o los cimientos de un desarrollo de software, la arquitectura es el conjunto de atributos que se han considerado según su importancia; como controles, procesos y situaciones posibles.

Se debe considerar cada característica funcional, si este proceso tiene algún defecto o si tiene algún riesgo, considerar si es apremiante habilitarlo o si existe algún defecto. Cuando se detectan vulnerabilidades esta debe ser solucionadas con el arquitecto del sistema.

En la figura 13 se presenta el flujo de pruebas en el SDLC según OWASP, este flujo define una lista de procesos que se deberían realiza en cada etapa del siglo de vida de desarrollo de software y que deberían ser verificados.

Figura 14. Flujo de pruebas típico en un SDLC



Fuente: Guía de Pruebas OWASP 2008 3.0 [En línea] disponible: https://owasp.org/www-pdf-archive/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

7.3 DESARROLLO

El desarrollo es un paso de codificación la cual debe buscar los estándares de codificación según la arquitectura que se escoja (esto depende de tipo de proyecto, pequeño, media o robusto). Es importante la arquitectura ya que por ejemplo la capa web no debe llamar directamente la base de datos.

Se debe tener por lo menos una mínima documentación entre código y estilo de comentarios preferidos.

Los errores más frecuentes en:

Manejo de excepciones a la hora de usar flujo de bloques de control, no utilizar sentencias específicas, esto ayudaría aprevenir los errores (se deben usar sentencias específicas).

Método de nombramiento para variable, tablas, funciones y variables son nombradas de forma errónea, estas deben ser nombradas de forma coherente con el proyecto.

El código debe ser testeado, reversible y versionado (repositorios) en caso de error sea posible retroceder a un punto en particular.

Es muy importante el uso de herramientas de revisión de código.

OWASP recomienda en este período una auditoria por fases tanto con los desarrolladores como con los arquitectos permitiendo explorar el código y por qué su estructura o forma de construcción. Esto permite identificar defectos de seguridad. En este proceso se debe considerar la lista de comprobación de top 10 de OWASP (A1. Inyección, A2. Autenticación, A3. Exposición de datos valiosos, A4. Entidades Externas XML, A5. control de acceso, A6. Configuraciones incorrectas de seguridad, A7. Secuencia de comandos en sitios cruzados, A8 Deserialización Insegura. [17]

En la revisión de código fuente su objetivo es buscar bug o errores de seguridad como errores de concurrencia o lógica, problemas de control de acceso, puertas traseras y debilidad de encriptación.

Es bueno comprender la estructura del código, porque se ha programado de la manera particular.

En esta fase de codificación se considera si el equipo de desarrollo está satisfecho con su labor, ataque de virus o gusanos y sus consecuencias secundarias y otros tipos de ataques a nivel criminal.

7.4 IMPLEMENTACIÓN

Es un deber realizar pruebas de configuración, según la guía de pruebas de OWASP la cual cumple con procedimientos y herramientas para verificar la seguridad en las aplicaciones no solo debe ser comprobada la aplicación, se debe realizar una comprobación de personas, procesos y tecnología ya que son factores importantes en el proyecto en los cuales pueden afectar el éxito o el fracaso del mismo. Porque es importante comprobar el conocimiento del personal, se convendría hacer antes de iniciar el proyecto y la codificación ya que, de su competencia, el grado de educación y conciencia profesional pueden dar un estado más robusto y de mejor calidad a los procesos.

17 OWASP Top 10 - 2017 Los diez riesgos más críticos en Aplicaciones Web

7.5 INSTALACIÓN

OWASP no solo permite evaluar la comprobación como tal de un proyecto en desarrollo de forma técnica, es realizar la comprobación de todo un conjunto en la cual se pueden descubrir posible problemas o incidentes que se pueden prevenir realizando una comprobación integral.

En la guía menciona las principales pruebas de:

- configuración (prueba de puertos, certificados de seguridad, robustez de claves, prueba de comportamiento de la aplicación, prueba de las interfaces y administración)
- Autenticación (Credenciales cifradas, cierre de sesión, tiempo de sesión, fuerza bruta, número de usuarios autenticados etc)
- Autorización (Privilegios, autorizaciones)
- Validación de datos (todo tipo de inyección sql, ldap,xml etc)
- Servicios web (WS,WSDL,HTTP,GET etc)

Según las técnicas de comprobación se consideran la inspección y revisión de manuales en la cual se verifican las políticas, procesos y los diseños de arquitectura escogidos para el proyecto. Esto se realiza analizando y haciendo entrevistas tanto al cliente como al equipo de trabajo.

Comprobación de la red o pruebas de intrusión (Pruebas de caja negra). Este tipo de prueba son realizadas con herramientas para que el proceso sea automatizado, son pruebas rápidas, no necesita mucha teoría, aunque puede solo verificar algunas amenazas relacionadas con el código.

Se debe considerar los tipos de datos y su importancia. La confidencialidad y la integridad.

7.6 MANTENIMIENTO

El mantenimiento son todas las labores que permiten cumplir con los objetivos para un buen funcionamiento. Estas deben ser realizadas con una frecuencia periódica en la aplicación, la infraestructura que la contiene y en sus componentes. Esta tarea debe ser ejecutada por el administrador del sistema o personal especializado y autorizado.

7.6.1 Modelado:

Después de realizar esta comprobación vendría el paso de Modelado y Amenazas el cual permite realizar la comprobación de recursos y la valoración de los riesgos el cual debe ir documentado. En esta valoración es importante identificar los activos dependiendo de su importancia, examinar las posibles vulnerabilidades tanto de maniobra y misión. Inspeccionar amenazas potenciales y crear maniobras de amortiguamiento de las mismas.

7.6.2 Implementación:

Al realizar la implementación nos queda por realizar la última comprobación que son las pruebas de intrusión. También se realizan las pruebas de configuración de la infraestructura (Es ineludible tener un plan de contingencia según el resultado de estas pruebas).

7.6.3 Mantenimiento y Operación:

Es irremediable realizar la comprobación de mantenimiento mensual o trimestral de la infraestructura para verificar que no se han sumado nuevos riesgos.
Al realizar los cambios se debe verificar para implementarlo en el entorno de producción.

Se evidencio todos los riesgos que se presentan en el SDLC, si las organizaciones ejecutan con dedicación las normas internas, implementándolas en todos sus proyectos de desarrollo de software, se realizará de forma automática y será parte de los procesos habituales de la compañía.

8 PLANTEAMIENTO DE METODOLOGÍA QUE CONTIENE PRACTICAS SEGURAS BASADAS EN OWASP PARA EL DESARROLLO Y DISEÑO DE SOFTWARE

Los sitios Web o SW contienen una gama de funcionabilidades de muy amplia variedad, permiten realiza consultas conectándose a una base de datos o dirigiéndose a otras aplicaciones o sitios; gestionando proceso, demostrando que pueden operar casi como una aplicación de un ordenador; por tal razón se generan tantas vulnerabilidades.

Para la correcta asignación de una metodología es requisito inicial identificar:

- Los tiempos estipulados para el proyecto y la cantidad de personas que desarrollaran el proyecto
- Reconocer el sistema operativo donde se va a trabajar
- Asignar el sistema de control de versiones vigilado, el cual depende de las necesidades del proyecto.
- El entorno de desarrollo debe ser muy similar al de producción para no presentar inconvenientes en la implementación, como prevenir e instalar posibles aplicaciones que impidan su funcionamiento.
- Disponer de plataformas modernas y robusta dependiendo de las necesidades del proyecto
- Planear la correcta configuración del servidor de almacenamiento en el cual se debe especificar qué servicios se encontrarán abierto y cuales bloqueados para su correcto funcionamiento y protección.
- Seleccionar la Metodología más adecuada para el proyecto.

El objetivo principal al iniciar el desarrollo es realizar el análisis de las condiciones que se deben plasmar en la aplicación; evitar que se encuentre en vulnerabilidad y que al mismo tiempo se pueda suplir los requerimientos del cliente sin afectar la integridad del software y de la información que se maneje.

Se debe tener en cuenta las amenazas existentes, es posible efectuar una comparación con la publicación de OWASP a las 10 principales amenazas que se publican anualmente (Injection, cross-site scripting etc). Es importante considerarlo antes de iniciar el desarrollo con el código o lenguaje de programación.

Es necesario seleccionar un framework robustos o confiables, que contengan librerías de uso activo y que permitan un mantenimiento o actualización y se reconozca su utilización.

Se recomienda la encapsulación de librerías para permitir solo la exposición de la función requerida en el sistema donde se está desarrollando.

8.1 CONSULTAS A LA BASE DE DATOS

Es recomendable tener en cuenta las vulnerabilidades de la inyección SQL, los cuales son insertados de forma dinámica por una consulta SQL, esto permite al intruso modificar, eliminar o extraer datos de la base de datos llegando en ocasiones hasta el control del sistema si no se toman las medidas adecuadas.

En la construcción hay que cumplir unas normas básicas de los principios de seguridad:

- Confidencialidad
- Integridad
- Disponibilidad

Al cumplir estos principios se cumple con un desarrollo exitoso.

Hay que recordar que el código puede ser víctima de dos amenazas durante el desarrollo: el sabotaje interno a la hora de programar y el sabotaje al tiempo del despliegue (entrega del proyecto, control del funcionamiento).

Para evitar alguno de estos dos eventos el cliente o líder del proyecto debe hacer un acuerdo legal de confidencialidad (evitar que se divulgue información valiosa del proyecto); Acuerdo de no competencia (Para evitar que el ex desarrollador del proyecto utilice información para benefició propio o de otros.)

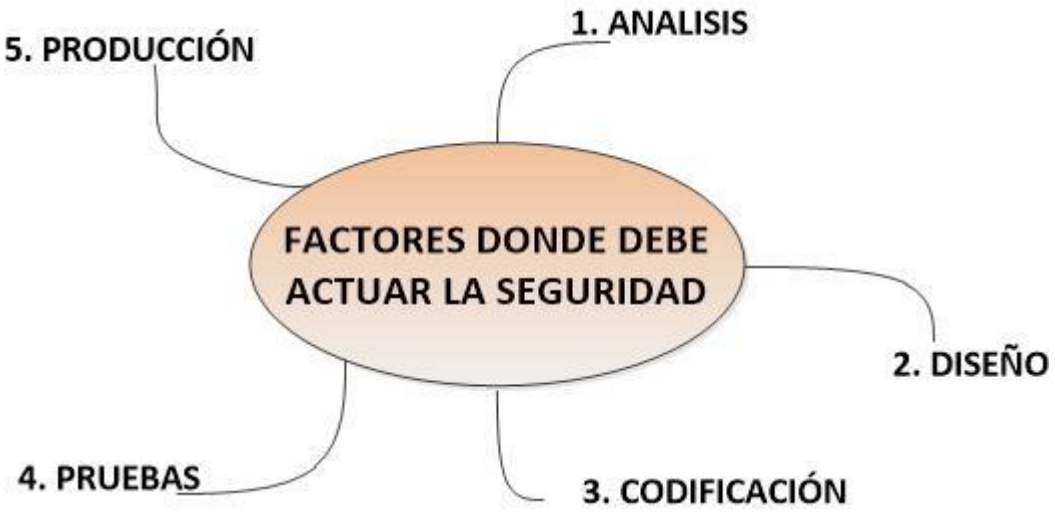
Se debe dar una formación inicial al equipo de trabajo para dar a conocer los procedimientos de seguridad como son:

- Seguridad en las contraseñas.
- Uso de elementos de almacenamiento.
- Software permitido para la labor.

La formación del equipo de trabajo también debe ser durante la duración de proyecto, dando motivación para evitar algún descontento que implique deslealtad. Capacitar a todo el equipo por igual para evitar posibles demoras en caso de que alguien del equipo de trabajo abandone el proyecto.

Hay que tener siempre claro los siguientes factores de desarrollo para cumplir con los requerimientos de seguridad, cada una de las representaciones de la figura 14 debe ser estimada según su importancia; ninguna debe ser descuidada y cada una debe considerar las vulnerabilidades para mejorar la seguridad.

Figura 15. Factores de desarrollo donde debe actuar la seguridad



Fuente: Propia.

- La planificación-análisis: Levantamiento de requerimientos, tanto del funcionamiento, el sistema como de los usuarios. Identificación del problema como de los objetivos.
    - Comprender los objetivos por parte del grupo de trabajo

- Validar la viabilidad. La aplicación cumple con las exigencias esperadas. Establecer las obligaciones del proyecto.

- Identificar los riesgos del Negocio o aplicación (Riesgos Técnicos)
    - Crear un top de riesgos (Alto, medio, bajo) para priorizar el riesgo.
    - Definir soluciones de los riegos
    - Realizar nuevamente la comprobación de riesgos.
    - Ejecutar Modelado de Amenazas
        - Técnica formal para determinar los niveles de riego.

- Capacitar o recordar las buenas prácticas al grupo de trabajo
    - Aminorar la amenaza
    - Evitar cambios que generen riesgo
    - Desactivar los servicios que no se requieran (Deshabilitar puerto que no se requieran)
    - Dar el mínimo privilegio a la aplicación o proyecto
    - Implementar controles y aplicaciones para disminuir el riego (Actualizaciones, antivirus etc.)

- Análisis del Desarrollo. Se asignan recursos, tiempo, se comprueba las limitaciones y se identifica los impactos del proyecto.

- Es posible utilizar aplicaciones que automaticen la administración de requisitos (Owasp Security RAT); al especificar los parámetros se pueden generar listas de requisitos que tienen una seguridad en común. Al revisar o escoger cada requisito se puede elegir en qué modo se gestiona el requisito. Cada requisito es un ticket donde es posible revisar su progreso.

- Diseño: Físico y lógico (se elaboran diagramas de flujo o pseudocódigos) para realizar un mejor análisis

  Es posible realizar un modelado o diagrama de flujo para modelar amenazas, si el equipo de trabajo resuelve trabajar bajo Python; existe una herramienta llamada Owasp Pytm que puede generar diagramas de flujo y secuencias de amenazas para el sistema que se esté abordando.

- Desarrollo y Documentación. En esta etapa se elige el lenguaje de programación que mejor convenga al proyecto con la codificación del mismo el cual sea robusto y cuente con actualizaciones y librerías disponibles. Se debe buscar la fácil integración de framework o software de codificación con otros programas como son las bases de datos.

- Pruebas. Permite confirmar el funcionamiento para evidenciar errores o evitarlos. Esto permite hacer los ajustes pertinentes para la mejora de la aplicación. Estas pruebas también son aplicadas tanto al resultado como a la seguridad que debe cumplir la aplicación.

    - Es muy importante dividir en el período del Desarrollo y las Pruebas en dos ramas de testing: Área testing: Herramientas de análisis de código:

        - Primera rama: testing Funcional-Este debe cumplir su función u objetivo del cliente (consultas, operaciones, vistas etc.)
        - Segunda Rama: Análisis y testing de seguridad Análisis de código (Herramientas) para detectar error (si detecta error se devuelve a la parte de codificación)- Implementación de guía de fallos de seguridad según metodología de OWAP, correr aplicaciones o herramientas de OWAP para comprobar código.

        - Aplicar las categorías de pruebas OWAP:
            - Recopilación de Información
            - Pruebas de gestión de la configuración
            - Pruebas de la lógica de negocio
            - Pruebas de Autenticación
            - Pruebas de Autorización

- Pruebas de gestión de sesiones
- Pruebas de validación de datos
- Pruebas de denegación de Servicio
- Pruebas de Servicios Web
- Pruebas de AJAX

- Implementación. Se coteja la integridad, la interoperabilidad del producto y su portabilidad. Se efectúa la capacitación o formación al usuario dependiendo del tipo de perfil que desempeñe en la aplicación.

- Mantenimiento y funcionamiento (Producción). Se resuelve errores pequeños, se verifica el buen funcionamiento realizando monitoreo, se ejecutan mejoras y se dan nuevas capacitaciones y entrega documentación para saber cómo operar y mantener la aplicación en funcionamiento.

Existe otra herramienta que permite realizar análisis de la composición de código (Dependency-Check) desde las famosas bibliotecas de terceros para facilitar la construcción de las aplicaciones en las cuales pueden ser inseguras; se encuentran las más famosas vulnerabilidades. Esta aplicación se ejecuta en línea de comandos. Contiene una serie de analizadores que escanean casa parte del proyecto.

OWASP brinda herramientas que facilitan la aplicación de pruebas como la guía de seguridad Web WSTG aplica pruebas de ciberseguridad de aplicaciones web y servicios web.

Un proyecto importante es el modelo de madurez de software el cual analiza el ciclo de vida del software basado en riesgos llamado modelo OWASP SAMM. Es un medidor para verificar los niveles de seguridad. Permite evaluar que niveles de seguridad se están aplicando, definir los controles de seguridad y su implementación.

Figura 16. Modelo OWASP SAMM V2

Fuente: OWASP SAMM Software Assurance Maturity Model [En línea] disponible: https://owasp.org/www-project-samm/

Este modelo permite realizar un análisis y tomar medidas de mejora en el ciclo de vida de desarrollo seguro, está basado en el riesgo, permite realizar la medición ya que se encuentra completamente definido, se basa en prácticas de seguridad, cada practica es un nivel de madurez, tiene cinco funciones y 15 prácticas de seguridad, cada practica desarrolla una actividad. Este modelo tiene la misión de evaluar cada situación del software, identificando su objetivo y direccionando a una hoja de ruta donde se puedan emplear criterios de mejora.

Se debe tener claro que a veces no es obligatorio ejecutar todos los pasos ya que dependen del tipo de aplicación.

Su estructura es la siguiente:

- Evalúa la seguridad actual del software
- Identifica el objetivo principal
- Define una hoja de ruta
- Establece fórmulas para usar las actividades especiales.

El principal modelo de madurez de OWASP SAMM (Software Assurance Maturity Model). Este modelo es evolutivo y efectivo para realizar prácticas de software. Son cinco funciones que contienen 15 prácticas de seguridad para medir los niveles de seguridad y madurez de la aplicación.

Cuadro 1 Descripción general del modelo SAMM

GOBERNANCIA	DISEÑO	IMPLEMENTACIÒN	VERIFICACION	OPERACIONES
Estrategias y Métricas	Evaluación de amenazas	Construcción segura	Evaluación de arquitectura	Administración de incidentes
Política y Cumplimiento	Requerimientos de seguridad	Implantación segura	Pruebas basadas en requisitos	Gestión de Medio Ambiente
Educación y Orientación	Arquitectura de seguridad	Gestión de defectos	Pruebas de seguridad	Gestión operativa

Fuente: OWASP SAMM Software Assurance Maturity Model [En línea] disponible: https://owaspsamm.org/model/

La Gobernanza: especifica la estructura de una organización y sus actividades de desarrollo de software.

Estrategias y Métricas: El objetivo de la métrica es construir un plan eficiente para al lograr los objetivos del software.

Política y Cumplimiento: su objetivo es que se entiendan los requisitos legales mientras que se cumplen las políticas de seguridad.

Educación y orientación: Permite involucrar a los miembros de la organización para que adquieran los conocimientos que les permitan identificar y mitigar los riesgos de seguridad.

Diseño: Son todos los procesos y actividades que la organización tiene para cumplir sus objetivos en el desarrollo de software.

Evaluación de amenazas: Su tarea es identificar y comprender los riesgos en todos los niveles de desarrollo del proyecto.

Requerimientos de seguridad: Exalta los diferentes tipos de requisitos de seguridad según su importancia en el contexto del desarrollo seguro.

Arquitectura de seguridad: su gestión es brindar seguridad a los procesos de diseño arquitectónico del software sin descuidar sus componentes.

Implementación: su propósito son los procesos y actividades que componen el software de forma eficiente el cual permite su funcionamiento de forma confiable con mínimo defecto.

Construcción segura: Resalta la importan de la creación de software seguro utilizando estándares seguros (componentes, librerías de terceros que cumplan los lineamientos de seguridad). Permite el manejo, ejecución de normas y rastreos del estado de seguridad.

Implantación segura: en esta etapa se pretende garantizar que el software sea seguro y que cumpla con la integridad de la información, todo acompañado de la seguridad, se realizan prácticas de verificación de seguridad en todo el software.

Gestión de defectos: recopila todos los registros de seguridad para analizarlos e identificar cada defecto de seguridad para su clasificación.

Verificación: Analiza todos los procesos de pruebas, controles, revisiones y evaluaciones de seguridad durante el desarrollo de software.

Evaluación de arquitectura: permite que se cumplan los requisitos de seguridad y cumplimiento de las políticas de seguridad.

Pruebas basadas en requisitos: Son controles de pruebas que se realizan con un periodo de frecuencia para saber si se están aplicando los requisitos de seguridad en el proyecto.

Pruebas de seguridad: So pruebas automatizadas las cuales son rápidas y eficientes.

Operaciones: son todas las actividades que mantienen el buen funcionamiento de una aplicación como la integridad, disponibilidad y la confidencialidad.

Administración de incidentes: Cuando ya se encuentra en marcha el proyecto es posible que se presenten incidentes de seguridad, el objetivo principal es identificar el incidente, darle atención de manera oportuna y limitar los daños para reparar y volver a un normal funcionamiento.

Gestión de Medio Ambiente: Cuando las aplicaciones se encuentran operativas pueden sufrir de obsolescencia ya que las nuevas versiones de las aplicaciones y los nuevos parches de seguridad pueden afectar el buen funcionamiento de la aplicación; esto no significa que no se deban actualizar, al contrario, es muy necesario instalar las actualizaciones con frecuencia para permitir la compatibilidad y la vida útil de la aplicación. Es preciso monitorear con frecuencia los informes de vulnerabilidad para identificar los tipos de ataque o vulnerabilidades más destacadas.

Gestión operativa: son todas las actividades, funciones y prácticas que se realizan para cumplir con la seguridad.

Implementación:

8.2 HERRAMIENTAS PARA COMPROBAR LA SEGURIDAD EN APLICACIONES WEB

Existen diversas herramientas para realizar auditoria de las aplicaciones web, a continuación, se enumeran las más utilizadas.

Nessus: Realiza un escaneo y búsqueda de vulnerabilidades tanto en la infraestructura como en las aplicaciones web. Busca malware y escanea rangos de IPS. La diferencia que tiene esta aplicación con NMAP es que realiza pruebas de vulnerabilidad en los servidores web, revisa la configuración SSL (Capa de trasporte seguro). Esta aplicación tiene una versión de descarga gratis que permite el escaneo de 16 IP y dos versiones de paga para servicios más robustos como gestión en la nube, escaneo de IP ilimitadas entre otras propiedades. **Esta aplicación sería muy apropiada en el SDLC tanto en la instalación como en el mantenimiento.**

SQLmap: Herramienta de código abierto la cual permite identificar las vulnerabilidades de inyección SQL. Cuenta con varios sistemas de gestores de bases de datos. Tiene un robusto motor de detección de fallas de SQL permitiendo la conexión directa con la base de datos, busca datos específicos en las tablas de la base de datos, permite enumerar los usuarios, las contraseñas, los privilegios, las bases de datos (con tablas y columnas). **Es muy adecuado tanto en la fase de instalación como en la fase de mantenimiento en el ciclo de vida del software.**

Wfuzz: Herramienta para escaneado de aplicaciones web en la cual se busca contenidos sospechosos que son utilizados en los ataques de fuerza bruta, estos ataques utilizan parámetros GET y POST que vienen en la inyección de datos, muestra el número de peticiones realizadas al servidor. **Esta herramienta puede ser utilizada en el ciclo de vida tanto en la instalación como en el mantenimiento.**

Nmap: Software que permite el rastreo de puerto abiertos, servicios y host en la red de datos. Permite detectar aplicaciones en puerto no convencionales con la opción –sV, conocer el nombre del host, su dirección IP, podemos ver qué servicios están saliendo con que puerto y entre otras, gracias a sus listas de opciones. Esta aplicación tiene un gran potencial para descubrir redes, esta aplicación es de código abierto permitiendo escanear redes grandes el cual funciona en todos los sistemas operativos. **Esta aplicación es perfecta a la hora de realizar auditoria de redes, ideal para la revisión de la aplicación web en el ciclo de vida del software tanto en la instalación como en el mantenimiento.**

Metasploit: Permite la detección y ejecución de exploits contra maquinas remotas. Valida vulnerabilidades y permite realiza pruebas de explotación de fallas, verificando los servicios de red y revisando cuales se encuentran activos almacenando los datos del equipo, importa archivos o importa sitios permitiendo agregar proyectos de prueba para ejecutar análisis para probar vulnerabilidades. **Igualmente se puede correr en el ciclo de vida del software tanto en la instalación como en el mantenimiento.**

Ratproxy: Esta herramienta permite la búsqueda de vulnerabilidades web compatible con varios sistemas operativos, puede identificar los datos que se trasportan por el SSL. Examina las respuestas JSON sospechosas, puede saber qué tipo de datos existen en el cache (si son datos de contenido sensible). Monitorea los redirectores (envió de datos desde el servidor o del cliente de una URL). **Esta aplicación se puede correr en el SDLC tanto en la instalación como en el mantenimiento.**

La mejor manera de comprender las metodologías que se mencionan es este capítulo es comenzar a usarlas paso a paso desde el inicio para entender cómo funcionan y alcanzar sus objetivos.

Estas metodologías se deben implementar por medios de capacitaciones y prácticas, las cuales deben ser realizadas por el equipo involucrado en cada ciclo de desarrollo de software.

9 CONCLUSIONES

No existe una aplicación web que sea segura en un 100%, pero si es posible encontrar mecanismos que permitan evitar posibles vulnerabilidades, desde la fase más básica como la toma de requisitos o requerimientos, hasta el tiempo de la codificación, OWASP en sus diversos proyectos brinda capacitación y orientación al momento de desarrollar una aplicación web.

Al codificar se presenta una práctica muy recurrente que es utilizar bibliotecas de software que presentan posibles amenazas; OWASP en su top 10 la clásica en el item 9A: uso de componentes con vulnerabilidades conocidas, esta vulnerabilidad obliga a los desarrolladores de software a aplicar herramientas de análisis de búsqueda de amenazas ya que por más conocimiento sobre la herramienta de desarrollo esta puede ser un vehículo para posibles vulnerabilidades; por lo cual esto demuestra la importancia de aplicar las herramientas de seguridad en cada fase del ciclo de vida de desarrollo de software.

La seguridad no solo se presenta en las pruebas de funcionamiento donde se cree que se tiene mayor vulnerabilidad; es importante realizar los criterios de seguridad en todo SDLC, esto permite un desarrollo más eficiente y estable al momento de cualquier ataque de seguridad.

OWASP presenta herramientas que permiten una identificación de las vulnerabilidades, proporcionando guías, listas de verificación, documentación, entornos de aprendizaje y entornos de pruebas para identificar las amenazas que pueden presentarse en los desarrollos de aplicaciones web, gracias a su comunidad es posible tener una información precisa e implementar buenas prácticas en los proyectos de desarrollo de aplicaciones web.

Algunas herramientas OWASP requieren de un conocimiento más profundo, como son algunos leguajes de programación específicos (Java, Rubi, Python etc) configuraciones de entorno, sistemas operativos Linux y su línea de comandos, dependerá de la robustez de proyecto y del equipo de trabajo adoptar la herramienta más adecuada según las necesidades a cubrir.

Es posible iniciar un plan de aprendizaje desde cero con OWASP con leguaje GO, aunque OWASP presenta las prácticas de codificación seguras de este lenguaje también es posible acceder a go.dev para iniciar el recorrido de aprendizaje desde lo esencial hasta los métodos de interfaz.

Es posible aprender jugando con las herramientas OWASP, por ejemplo, el curioso juego de cartas Cornucopia donde cada carta contiene un patrón de ataque del software y posibles vulnerabilidades. Otro muy pedagógico es el popular juego de escalera donde se encuentran controles y riesgos. Y para aprender con simulaciones está el entorno para realizar prácticas de seguridad, donde se simula varias fallas de seguridad y problemas de seguridad para agregar más emoción a cada prueba.

10 RECOMENDACIONES

El desarrollo de aplicaciones web está en un constante cambio, es necesaria conocer la información, adquirir conocimientos de nuevas técnicas, identificar los nuevos ataques, aprender las vulnerabilidades, conocer las herramientas de comprobación, identificar los errores de seguridad, establecer los framework con más demanda y seguros para el desarrollo, entre otros, ya que los avances que ha tenido la seguridad en aplicaciones web se han vuelto más exigente con el pasar del tiempo.

Para poder iniciar en el mundo de aprendizaje de OWASP es posible aprovechar los sitios web, las guías de documentación como los existentes en la página secureflag referenciada por OWASP; tiene un repositorio de información de vulnerabilidades de software con ejercicios de código en diferentes framework y laboratorios prácticos permitiendo una explicación detallada de los errores más comunes y vulnerabilidades en el código.

Es preciso no desmotivarse al iniciar la búsqueda del aprendizaje en el entorno OWASP. ya que muchos sitios y aplicaciones son de pago, existen otros repositorios ubicados en Github que podrían ser de provecho, es ineludible aprender de este entorno para poder aprovechar las guías, códigos y proyectos publicados de manera libre.

Es fundamental seguir enriqueciendo la base del inglés técnico que se debe tener para abordar toda la documentación e información compartida en OWASP; ya que la gran mayoría de sitios y publicaciones se encuentran en este idioma.

11 BIBLIOGRAFÍA

AESS_L13_G3_14. Descripción de la Metodología: Métrica 3 {En línea}. Disponible en: https://sites.google.com/site/aessl13g314/practica-2/1-2-descripcion-de-la-metodologia

ARJONA Rafa. ¿QUÉ ES BACKEND? {En línea}. Disponible en: https://rafarjonilla.com/que-es/backend/

BBC News Mundo. EE.UU. declara estado de emergencia tras un ciberataque a la mayor red de oleoductos del país. {En línea}. {10 Mayo 2021}. Disponible en: https://www.bbc.com/mundo/noticias-internacional-57033536

BRITO ABUNDIS Carlos Joaquín. Metodología para Desarrollar Software Seguro. Universidad Autónoma de Zacatecas. {En línea}. {3 Diciembre 2013}. Disponible en: https://www.redalyc.org/articulo.oa?id=512251564005

CARO Andrés. Modelos de Desarrollo Seguro de Software. Cátedra VIEWNEXT. {En línea}. {7 junio 2017}. Disponible en: https://es.slideshare.net/InformaticaUCM/modelos-de-desarrollo-seguro-de-software

CIBERSEGURIDAD CHILE. Lineamientos para Desarrollo de Software Guía Técnica –Gob Digital Chile. {En línea}. {2021}. Disponible en: https://cms-dgd-prod.s3-us-west-2.amazonaws.com/uploads/pdf/Guia_Desarrollo_Software_-_v2.0_-_Mayo_2021.pdf?

Como detectar cuando las consultas no parametrizadas dañan el rendimiento de SQL Server y que hacer {En línea}. Disponible en: https://geeks.ms/rcorral/2009/12/28/como-detectar-cuando-las-consultas-no-parametrizadas-daan-el-rendimiento-de-sql-server-y-que-hacer/

ÁLVAREZ Gerardo. Ciclo de vida de un software. {En línea} {24 Septiembre 2018}. Disponible en: https://www.kyocode.com/2018/09/ciclo-de-vida-de-un-software/

Content Security Policy {En línea}. Disponible en: (CSP)https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

CUENCA DIAZ Cesar. Desarrollo Seguro: Principios y Buenas Practicas. OWASP. {En línea}. Disponible en: https://owasp.org/www-pdf-archive/Desarrollo_Seguro_Principios_y_Buenas_Pr%C3%A1cticas..pdf

DAVILA Manuel. Diseño de Software Seguro. {En línea}. Disponible en: https://acis.org.co/portal/Revista/119/Uno.pdf

DIAZ Brian Alexander. {En línea} {29 julio 2020}. Disponible en: https://www.doblefactor.com/secdevops/los-5-mejores-consejos-de-seguridad-de-owasp-para-disenar-api-rest-seguras/

DIÉGUEZ, Javier. “La digitalización es imparable pero falta cultura de ciberseguridad". {En línea}. {9 Abril 2021}. Disponible en: https://www.eitb.eus/es/radio/radio-vitoria/programas/radio-vitoria-gaur-actualidad/detalle/7966984/dieguez-la-digitalizacion-es-imparable-falta-cultura-ciberseguridad/

Dracon {En línea}. Disponible en: https://github.com/thought-machine/dracon/

DW.FEW (EFE, AFP, El Tiempo). Colombia: Anonymous se atribuye "hackeo" de la página del Ejército. {En línea}. {5 Mayo 2021}. Disponible en: https://www.dw.com/es/colombia-anonymous-se-atribuye-hackeo-de-la-p%C3%A1gina-del-ej%C3%A9rcito/a-57436738

El Modelo Descripción general de modelo SAMM {En línea}. Disponible en: https://owaspsamm.org/model/

ELLINGSWORTH John, OWASP SAMM version 2.0 {En línea}. {5 Febrero 2020}. Disponible en: https://owasp.org/www-event-2020-NewZealandDay/assets/presentations/Ellingsworth--OWASP_SAMM--20200221.pdf

ESET. Security Report Latinoamérica 2020. {En línea}. Disponible en: https://empresas.eset-la.com/archivos/novedades/87/ESET-security-report-LATAM2020.pdf

ESET. Security Report Latinoamérica 2021. {En línea}. Disponible en: https://www.welivesecurity.com/wp-content/uploads/2021/06/ESET-security-report-LATAM2021.pdf

ESPITIA Diego, BORGHELLO Cristian. Metodologías de desarrollo seguro (Secure-SDLC). {En línea}. {10 Septiembre 2016}. Disponible en: https://www.youtube.com/watch?v=MSRCqSFr2Q4

FERNANDEZ Yúbal. {En línea} {23 agosto 2019}. API: qué es y para qué sirve disponible en: https://www.xataka.com/basics/api-que-sirve

Fuerza Bruta contra Directorios utilizando Wfuzz {En línea} {3 Agosto 2020}. Disponible en: http://www.reydes.com/d/?q=Fuerza_Bruta_contra_Directorios_utilizando_Wfuzz

GOB DIGITAL. Lineamientos para Desarrollo de Software. {En línea}. {diciembre del 2018}. Disponible en: https://www.ciberseguridad.gob.cl/media/2018/12/Guia_de_desarrollo_de_software_para_el_Estado.pdf

GONZALEZ FUNG Carlos. Aplicar Exitosamente un ciclo de Desarrollo Seguro de Aplicaciones. {En línea}.{2018}. Disponible en: https://cidecuador.org/wp-content/uploads/congresos/2018/congreso-internacional-de-desarrollo-de-software/diapo/aplicar-exitosamente-un-ciclo-de-desarrollo-seguro-para-las-aplicaciones_carlos-gonzales.pdf

GUÍA DE INICIO RÁPIDO PARA LA VERSIÓN 2.0 -Acerca de OWASP SAMM {En línea} disponible en: https://owaspsamm.org/guidance/quick-start-guide/

Guía Para Desarrollo Sitio Web Gobierno De Chile Ministerio De Secretaria General De Gobierno {En línea}. {2004}. Disponible en: https://www.guiadigital.gob.cl/guiaweb_old/guia/archivos/GuiaWeb2004.pdf

HERNANDEZ. Uriel. MVC (Model, View, Controller) Explicado. {En línea}. Disponible en: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

Informe de McAfee Labs sobre amenazas - COVID-19. {En línea} {Julio 2020}. Disponible en:https://www.mcafee.com/enterprise/es-es/assets/reports/rp-quarterly-threats-july-2020.pdf

IT DIGITAL SECURITY. El 82% de las vulnerabilidades de las aplicaciones web están en el código fuente. {En línea} {19 febrero del 2020}. Disponible en: https://www.itdigitalsecurity.es/vulnerabilidades/2020/02/el-82-de-las-vulnerabilidades-de-las-aplicaciones-web-estan-en-el-codigo-fuente

ITFORTECHIES. Desarrollo seguro de software. ISO 27001. {En línea} {30 agosto del 2019}. Disponible en: https://itfortechies.wordpress.com/2019/08/30/desarrollo-seguro-de-software-iso-27001/

JUSTCODEIT BY KEEPCODING. Desarrollo Seguro de Aplicaciones. {En línea}. Disponible en: https://justcodeit.io/que-es-el-desarrollo-seguro-de-aplicaciones/

Las librerías y los frameworks JavaScript más populares {En línea} disponible en: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/frameworks-javascript-y-librerias-populares/

LOPEZ Amparo. El Modelo de Entidad de Relación {En línea}. Disponible en: http://hp.fciencias.unam.mx/~alg/bd/er.pdf

LLAMAS Jonatan. Historia del software {En línea} {9 Diciembre 2020} Disponible en: https://economipedia.com/definiciones/historia-del-software.html

MAIDA Esteban Gabriel. PACIENZIA Julián. Metodologías de desarrollo de software. Biblioteca Digital de la Universidad Católica Argentina {En línea} Diciembre 2015}. Disponible en: https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

MARULANDA L Cesar. CEBALLOS H. Julián. Una Revisión De Metodologías Seguras En Cada Fase Del Ciclo De Vida Del Desarrollo De Software. Ingenierías USBMed. {En línea}. {11 enero 2012}. Disponible en:https://revistas.usb.edu.co/index.php/IngUSBmed/article/view/260/174

MELON Luis.10 herramientas para escanear vulnerabilidades web {En línea} disponible en: https://ciberseguridad.blog/10-herramientas-para-escanear-vulnerabilidades-web/

METODOLOGÍA DE DESARROLLO DE SOFTWARE VERSIÓN 001- UNIVERSIDAD CATÓLICA LOS ÁNGELES CHIMBOTE PERÚ. Elaborado por la División de Sistemas {En línea} {2017}. Disponible en: https://www.uladech.edu.pe/images/stories/universidad/documentos/2018/metodologia-desarrollo-software-v001.pdf

MILANO Pablo. Seguridad en el Ciclo de vida del Desarrollo de Software. {En línea}. {12 Septiembre 2007}. Disponible en: http://www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

MORALES Da. Metodología tradicional o ágil ¿Cuál es la mejor opción para mi proyecto de desarrollo de software? {En línea} {20 febrero 2019}. Disponible en: https://www.scio.com.mx/blog/metodologia-tradicional-o-agil-software/

Nessus {En línea} disponible en: https://es-la.tenable.com/products/nessus?tns_redirect=true

NIETO Ana. OWASP Top-10 2017 está muriendo, larga vida a OWASP Top-10 2021 {En línea} {25 septiembre 2021}. Disponible en: https://unaaldia.hispasec.com/2021/09/owasp-top-10-2017-esta-muriendo-larga-vida-a-owasp-top-10-2021.html

NMAP {En línea} disponible en: https://nmap.org/

Open SAMM. Modelo de Madurez de Garantía de Software Versión 1.0. {En línea}. Disponible en: https://www.opensamm.org/downloads/SAMM-1.0.pdf

OWASP, apostando por un software seguro {En línea} {25 enero 2021}. Disponible en: https://ciberseguridadtotal.com/owasp-apostando-por-un-software-seguro/

OWASP. SALAZAR Edgar. Pruebas de Seguridad en aplicaciones web según OWASP. {En línea}. Disponible en: https://owasp.org/www-pdf-archive/OWASP_SUSCERTE.pdf

OWASP TOP 10 -2021 A03:2021 – Injection {En línea}. Disponible en: https://owasp.org/Top10/A03_2021-Injection/

OWASP Top 10 - We've got you covered! {En línea} disponible en: https://www.sonarqube.org/features/security/owasp/?gads_campaign=South-America-Generic&gads_ad_group=OWASP&gads_keyword=owasp%20top10&gclid=Cj0KCQjw_fiLBhDOARIsAF4khR1ngeie4WrYobpvKVcBUzZN__p7m7tkSvWK_Jx-XTWhPbLTDj_pL7kaAkUiEALw_wcB

OWASP Web Security Testing Guide. OWASP {En línea}. Disponible en: https://owasp.org/www-project-web-security-testing-guide/

OWASP WebGoat {En línea}. Disponible en: https://owasp.org/www-project-webgoat/

OWASP Go Secure Coding Practices Guide {En línea}. Disponible en: https://owasp.org/www-project-go-secure-coding-practices-guide/

OWASP Snakes And Ladders {En línea}. Disponible en: https://owasp.org/www-project-snakes-and-ladders/

OWASP Code Pulse {En línea}. Disponible en: https://owasp.org/www-project-code-pulse/

OWASP Security RAT {En línea}. Disponible en: https://github.com/SecurityRAT/SecurityRAT

OWASP Application Security Verification Standard {En línea}. Disponible en: https://owasp.org/www-project-application-security-verification-standard/

OWASP Mobile Top 10 {En línea}. Disponible en: https://owasp.org/www-project-mobile-top-10/

PARODI Alejandro. Qué es XXE (XML External Entity) y cómo se soluciona July {En línea} {20 julio 2021}. Disponible en: https://blog.hackmetrix.com/xxe-xml-external-entity/

PEREIRA.Dani.Desarrollo seguro de software. ISO 27001 {En línea}. Disponible en: https://itfortechies.wordpress.com/2019/08/30/desarrollo-seguro-de-software-iso-27001/

PROYECTOS AGILES.org. Qué es CRUM. {En línea}. Disponible en: https://proyectosagiles.org/que-es-scrum/

Proyectos OWASP {En línea} disponible en: https://owasp.org/projects/

Que-es-el-backend-y-que-es-el-frontend {En línea} disponible en: https://www.suratica.es/que-es-el-backend/que-es-el-backend-y-que-es-el-frontend/

¿Qué es el SSL? {En línea}. Disponible en: https://www.globalsign.com/es/ssl-information-center/what-is-ssl

¿Qué aportó a la ciencia Alan Turing? {En línea}. {12 septiembre 2019}. Disponible en:https://www.lavanguardia.com/historiayvida/historia-contemporanea/20180611/47312986353/que-aporto-a-la-ciencia-alan-turing.html

Quick Start Guide {En línea}. Disponible en: https://docs.rapid7.com/metasploit/

Ratproxy: herramienta pasiva de evaluación de seguridad de aplicaciones web {En línea}. Disponible en: https://code.google.com/archive/p/ratproxy/wikis/RatproxyDoc.wiki

Redirección {En línea} disponible en: https://es.ryte.com/wiki/Redirecci%C3%B3n

REYES Daniel. Incorporando la seguridad al proceso de desarrollo de software. Security Atwork. {En línea}. {23 octubre de 2009}. Disponible en: https://www.securityartwork.es/2009/10/23/incorporando-la-seguridad-al-proceso-de-desarrollo-de-software/

SAMM OWASP {En línea} disponible en: https://owasp.org/www-project-samm/

SUAREZ Heiner. TAPIERO Hawin. Políticas de Seguridad Versión 1. {En línea}. {29 marzo 2017}. Disponible en: http://repository.udistrital.edu.co/bitstream/11349/8322/4/Anexo%20C%20-%20Politicas%20de%20seguridad.pdf

SUÁREZ Herly. Lineamientos de Desarrollo Seguro de Software: Ministerio Salud y Protección Social. {En línea}. {14 septiembre 2018}. Disponible en: https://www.minsalud.gov.co/Ministerio/Institucional/Procesos%20y%20procedimientos/CVSS01.pdf

SQLMAP {En línea} disponible en: https://sqlmap.org/

The world's most used penetration testing framework {En línea}. Disponible en: https://www.metasploit.com/

TRAINING & GUIDANCE FOR DOING APPSEC RIGHT! {En línea}. Disponible en: https://www.securityknowledgeframework.org/

¿Cómo se utiliza Github pages? {En línea}. Disponible en: https://developer.mozilla.org/es/docs/Learn/Common_questions/Using_Github_pages

Trends Report- Cenzic Application vulnerability trend report 2014 {En línea}. {2014}. Disponible en: https://www.infopoint-security.de/medien/cenzic-vulnerability-report-2014.pdf

UNGOTI Innovating Ideas, Ciclo de Vida del Desarrollo de Software {En línea}. Disponible en: https://ungoti.com/es/soluciones/desarrollo-de-software/sdlc/

Welcome to a tour of Go {En línea}. Disponible en: https://go.dev/tour/list

WFUZZ {En línea}. Disponible en: https://www.kali.org/tools/wfuzz/

Wiki OWASP {En línea}. Disponible en: https://wiki.owasp.org/index.php/Main_Page

WHITEHEAD y Russell. Principia Mathematica Volume I {En línea}. Disponible en: https://lesharmoniesdelesprit.files.wordpress.com/2015/11/whiteheadrussell-principiamathematicavolumei.pdf

RESUMEN ANALITICO EDUCATIVO
RAE

1. Información General	
Tema	Aplicar la seguridad en el desarrollo de software de aplicaciones web utilizando la metodología OWASP y sus herramientas que permitan su verificación, prevención de ataques y vulnerabilidades.
Titulo	La Seguridad Informática En El Desarrollo De Aplicaciones Web Mediante El Uso De La Metodología OWASP
Autor	Tania Sierra Huertas
Fuente bibliográfica	Se referencia 80 fuentes bibliográficas, algunas que mencionan la temática principal son: OWASP. SALAZAR Edgar. Pruebas de Seguridad en aplicaciones web según OWASP. {En línea}. Disponible en: https://owasp.org/www-pdf-archive/OWASP_SUSCERTE.pdf OWASP TOP 10 -2021 A03:2021 – Injection {En línea}. Disponible en: https://owasp.org/Top10/A03_2021-Injection/ OWASP Top 10 - We've got you covered! {En línea} disponible en: https://www.sonarqube.org/features/security/owasp/?gads_campaign=South-America-Generic&gads_ad_group=OWASP&gads_keyword=owasp%20top10&gclid=Cj0KCQjw_fiLBhDOARIsAF4khR1ngeie4WrYobpvKVcBUzZN__p7m7tkSvWK_Jx-XTWhPbLTDj_pL7kaAkUiEALw_wcB OWASP, apostando por un software seguro {En línea} {25 enero 2021}. Disponible en: https://ciberseguridadtotal.com/owasp-apostando-por-un-software-seguro/
Año	2022
Resumen	Gracias a los avances del Internet y las necesidades de consulta y almacenamiento, se han podido utilizar diferentes aplicaciones web que permiten brindar información de forma rápida y eficiente; para mantener la integridad de los datos y la seguridad de las aplicaciones es indispensable que desde su definición, diseño, desarrollo, implementación y mantenimiento (Ciclo de vida de desarrollo de software) se determine las posibles vulnerabilidades o riesgos utilizando la metodología OWASP que permite implementar principios y buenas prácticas de seguridad.
Palabras Claves	Metodología, Seguridad, Software, Vulnerabilidad.
Contenido	Introducción Definición del problema Antecedentes del problema Formulación del problema Justificación Objetivos

	Marco referencial Marco teórico Ciclo de Vida de Software ¿Qué es OWASP? Marco conceptual Marco histórico Línea de Tiempo Marco Científico o Tecnológico Requirements (Requisitos) Design (Diseño) Implementation (Implementación) Verification (Verificación) Policy Gap Evaluation Training/Education (Formación / Educación) Culture Building & Process Maturing (Construcción de cultura y proceso de maduración) Operation (Operación) Marco Legal Amenazas Recientes Que Afectaron La Seguridad En Las Aplicaciones Web ¿Cómo se ha hecho el control y prevención de estos riesgos? Evidenciar Las Dificultades Y Mejoras En Las Fases De Configuración Y Codificación En Las Aplicaciones Web Fase De Configuración: Backend (Lado del Servidor) Bases de datos: Codificación La injetion Pérdida de Autenticación Exposición de Datos Sensibles Entidad Externa de XML Pérdida de control de acceso Configuración de Seguridad Incorrecta Comandos en sitios cruzados Deserialización Insegura Componentes con Vulnerabilidades Conocidas Monitoreo Insuficiente Frontend (lado del cliente) Exponer Las Mayores Vulnerabilidades Presentadas Dentro Del Ciclo De Vida De Desarrollo De Software En Aplicaciones Web Según La Guía De Pruebas OWASP V Antes Del Desarrollo Definición Y Diseño Desarrollo Implementación Instalación Mantenimiento

	Modelado Implementación Mantenimiento y Operación: Planteamiento De Metodología Que Contiene Practicas Seguras Basadas En OWASP Para El Desarrollo Y Diseño De Software Consultas a la base de datos herramientas para comprobar la seguridad en aplicaciones web Conclusiones Recomendaciones Bibliografía

2. Descripción del problemas de investigación

La ausencia de buenas prácticas, preparación y metodologías abre un riesgo de seguridad o vulnerabilidad a amenazas. Se requiere la identificación de las posibles amenazas en el ciclo de vida del software ya que pueden existir peligros no identificados.

La falta de recursos y los tiempos de entrega limitados conlleva a que el desarrollo no tenga un ciclo de tiempo adecuado para incluir la seguridad al proyecto. El afán por cumplir los requerimientos tanto en funcionabilidad como en operabilidad hace que solo se realice un testeo de funcionamiento y no de seguridad. Sumado la carencia de competencia tecnológica por parte del equipo de desarrollo, no tener claridad en los objetivos, no utilizar la metodología adecuada pueden llevar a un resultado de una aplicación vulnerable la cual puede presentar riesgo en la integridad de los datos e información que la compone, con frecuencia solo se considera el papel de “atacado o victima” al instante de realizar pruebas (se debe considerar también el papel de atacante) para tener ambas percepciones de posibles situaciones de vulnerabilidad.

3. Objetivos

General:

Establecer cuáles son las vulnerabilidades de seguridad en el desarrollo de aplicaciones web a partir de la metodología OWASP.

Específicos:

Inspeccionar los acontecimientos más importantes sobre ataques y amenazas de seguridad en aplicaciones web.

Evaluar las prácticas deficientes presentes en el desarrollo de aplicaciones web en sus fases de configuración y codificación.

Estimar dentro del ciclo de vida de desarrollo de software donde se presentan las mayores vulnerabilidades en la seguridad del desarrollo de aplicaciones web.

Proponer una metodología para el desarrollo seguro de aplicaciones web, acompañado de buenas prácticas basadas en OWASP para los desarrolladores de aplicaciones web.

4. Metodología

La metodología que utilice para el desarrollo de este trabajo fue la siguiente:

- Análisis de metodologías Agiles y Robustas
- El ciclo PHVA (Planificar-Hacer-Verificar-Actuar)
- Análisis del ciclo de vida del software
- Ciclo de vida de del software según OWASP
- Metodología OWASP

5. Referentes teóricos

Se consultan diferentes metodologías, se estudia y analiza los proyectos de seguridad más reconocidos de OWASP como son el top 10, Modelo madurez de software SAMM, ciclo de vida de desarrollo de software según OWASP y guía de pruebas de seguridad web de OWASP.

6. Referencias conceptuales

Se realizan análisis de las incidencias de inseguridad en aplicaciones web, ataques y vulnerabilidades.

7. Resultados

Se propone que se aplicada la metodología OWASP en todas las fase del SDCL

8. Conclusiones

El desarrollo de aplicaciones web está en un constante cambio, es necesaria conocer la información, adquirir conocimientos de nuevas técnicas, identificar los nuevos ataques, aprender las vulnerabilidades, conocer las herramientas de comprobación, identificar los errores de seguridad, establecer los framework con más demanda y seguros para el desarrollo, entre otros, ya que los avances que ha tenido la seguridad en aplicaciones web se han vuelto más exigente con el pasar del tiempo.

Para poder iniciar en el mundo de aprendizaje de OWASP es posible aprovechar los sitios web, las guías de documentación como los existentes en la página secureflag referenciada por OWASP; tiene un repositorio de información de vulnerabilidades de software con ejercicios de código en diferentes framework y laboratorios prácticos permitiendo una explicación detallada de los errores más comunes y vulnerabilidades en el código.

Es preciso no desmotivarse al iniciar la búsqueda del aprendizaje en el entorno OWASP. ya que muchos sitios y aplicaciones son de pago, existen otros repositorios ubicados en Github que podrían ser de provecho, es ineludible aprender de este entorno para poder aprovechar las guías, códigos y proyectos publicados de manera libre.

Es fundamental seguir enriqueciendo la base del inglés técnico que se debe tener para abordar toda la documentación e información compartida en OWASP; ya que la gran mayoría de sitios y publicaciones se encuentran en este idioma.