GESTIÓN DE SEGURIDAD EN EL PROCESO DE DESARROLLO DE SOFTWARE (2016)

Narváez Vallejo, Yudy.
yudynarvaez27@hotmail.com
Universidad Piloto de Colombia

***Resumen*—El avance tecnológico viene acompañado de una gran cantidad de variables, dentro de las cuales se encuentra el software; cada vez son más los dispositivos y aparatos electrónicos que adecúan sus mecanismos y funcionalidad por medio de un software, esto implica que gradualmente sea mayor la exigencia en cuanto a la calidad y seguridad de las aplicaciones, pues mientras se propende por adecuar sistemas inteligentes y biométricos, también se requiere que el desarrollo de software que se realice para ellos sea mediante procesos y procedimientos que garanticen la seguridad de la información contenida. La proliferación de la información a través de los diferentes medios de comunicación, ha llevado a las diferentes compañías a incorporar medidas de seguridad cada vez mejores y más robustas en los diferentes procesos involucrados para el normal funcionamiento de estas, tanto en su infraestructura tecnológica como en la adquisición de Sistemas Operativos que brinden actualizaciones permanentes. Sin embargo el proceso de desarrollo de Software no cuenta con una base homogénea para trabajar, y representa una vulnerabilidad que puede ser aprovechada por un atacante. Por lo tanto es importante realizar una gestión de Seguridad al Proceso de Desarrollo de Software, que cumpla con los requerimientos tanto funcionales, como de calidad y seguridad.**

***Abstract* – Technological progress is accompanied by a lot of variables, among which is the software; more and more electronic devices and appliances that adjust their mechanisms and functionality through software, this means that gradually is greater requirement regarding the quality and safety of applications, because while it tends to align intelligent systems and biometric , also it requires the development of software that is made for them either through processes and procedures to ensure the security of the information. The proliferation of information through different media, has led to different companies to incorporate security measures increasingly better and more robust in the different processes involved for the normal functioning of these, both in its technological infrastructure and the acquisition of operating systems that provide constant updates. However the software development process does not have a homogeneous basis for work, and represents a vulnerability that can be exploited by an attacker. Therefore it is important to management Software Security Development Process that meets both the functional requirements, quality and safety requirements.**

***Índex Terms*—Desarrollo de Software Seguro, S-SDLC, Seguridad, Gestión de Seguridad, Modelo de Amenaza, Gestión de Riesgos, Vulnerabilidad, Controles, OWASP, Cobit, ISO27001.**

I. INTRODUCCIÓN

El Software representa un conjunto complejo de aspectos de seguridad que deben ser atendidos por los arquitectos, diseñadores y programadores. Es importante que la seguridad se tenga en cuenta durante todo el proceso de desarrollo, gestionando de manera adecuada y oportuna los riesgos y vulnerabilidades, teniendo en cuenta que la seguridad no depende exclusivamente de la protección de las redes, usuarios o configuraciones, sino también de las propias aplicaciones, ya que estas se pueden escribir de muchas maneras diferentes, y en muchos idiomas. En el desarrollo de este documento se analizan algunos aspectos sobre la utilización de principios y buenas prácticas de Seguridad durante el Ciclo de Vida del Desarrollo de Software (S-SDLC)[1]. Cada una de las fases en el ciclo de desarrollo se proponen estableciendo requerimientos y controles de seguridad, los cuales deben ser medibles. Se ha encaminado hacia el entorno empresarial y administrativo, tomando como referencia los controles que aplican a la seguridad de aplicaciones de la Norma ISO/IEC 27002:2013, ISO/IEC 27034:2011, Cobit y aspectos importantes del proyecto OWASP[2].

II. S-SDLC SECURE SOFTWARE DEVELOPMENT LIFE CYCLE

El ciclo de Vida de Desarrollo de Software Seguro describe las fases del ciclo del software y el orden en que esas fases se ejecutan de acuerdo a los requerimientos del negocio, con el fin de implementar las aplicaciones que se requieren para un procesamiento seguro. En la Fig. (1), se observan las fases para el ciclo de desarrollo de software:

[1] *Secure Software Development Life Cycle*
[2] *Open Web Application Security Project*

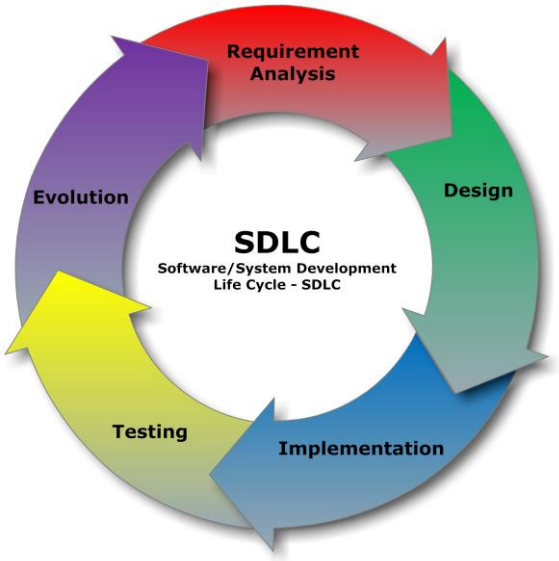

Figura 1. Ciclo de Vida de Desarrollo de Software/Sistemas. Fuente: http://wh0s.org/ [4]

Existen distintos S-SDLC, siendo el más conocido el de Microsoft (Microsoft Trustworthy Computing SDL). Sin embargo en este artículo nos enfocaremos en las actividades generales que se suelen añadir al ciclo de vida; estas actividades básicas, permiten desarrollar software de calidad teniendo en cuenta los principios fundamentales para la seguridad de la información: integridad, disponibilidad y confidencialidad.

El modelado de Riesgo de Amenazas es el método más importante de mitigación en el desarrollo de aplicaciones. Si analizamos detenidamente y seleccionamos los controles a través del modelado de riesgo de amenazas, el resultado final será una implementación de sistemas que demostrablemente reducen los riesgos de negocio, que generalmente conduce a un aumento de la seguridad y la reducción de los fraudes y pérdidas.

A. Seguridad en el análisis de Requerimientos

Es importante la existencia de políticas, normas y documentación, ya que esto proporciona al equipo de desarrollo las directrices a seguir. Las personas hacen lo correcto cuando saben lo que es correcto.

En esta fase se deben añadir a todos los requerimientos del proyecto, los requisitos de seguridad que surjan como gestión de password y autenticación, la gestión de roles, los requisitos de conocimientos del equipo, el sistema de logs, entre otros. Se debe realizar una evaluación de riesgos que permita identificar los posibles riesgos tales como: la cesión de datos a terceros, la política de confidencialidad, los planes de recuperación ante desastres o la seguridad de los datos de los usuarios.

Los requerimientos en esta fase serían:

- ✓ Control de autenticación
- ✓ Control de roles y privilegios
- ✓ Requerimientos orientados al riesgo
- ✓ Aprobación de Privilegios

B. Seguridad en el Proceso de Diseño

Resolver los requerimientos de seguridad. Una vez identificados los requerimientos, durante esta fase, se deberán diseñar las medidas de actuación para los requisitos de seguridad detectados anteriormente. Por ejemplo se deberá resolver casos como el de la política de contraseñas: se tendrá que determinar si se prefieren contraseñas largas y con pocos cambios o si por el contrario se prefieren cortas y de una menor duración.

Revisión del Diseño y la Arquitectura. Antes de implementar el diseño, se deben tener en cuenta los posibles riesgos de seguridad que puedan existir en la arquitectura o en el propio diseño, ya que si se detectan en una fase posterior, el coste de solucionar estos problemas será mucho más elevado. En esta fase, el diseño debe tener una segunda revisión vista desde el prisma de la seguridad en el que se traten temas como el cifrado de las comunicaciones, el cifrado de los datos o el uso del cloud en caso de ser contemplado. Por ejemplo, una aplicación de cálculo de nóminas, puede subir toda la información de los usuarios directamente a la nube, o puede subir únicamente el cálculo de las operaciones, dejando los datos privados y confidenciales en nuestro servidor local. En esta fase se debe revisar todo este tipo de circunstancias y buscar la solución que mejor se adecue a nuestro caso.

Modelado de Amenazas. Se representa de manera estructurada, toda la información que afecta a la seguridad del software, buscando capturar, organizar y analizar esta información para tomar decisiones informadas sobre los riesgos de seguridad en las aplicaciones. Permite a las organizaciones determinar el control correcto y producir contramedidas efectivas dentro del presupuesto.

Pasos para llevar a cabo un análisis de modelo de amenazas:

- ✓ Recopilar información básica.
- ✓ Crear y analizar el modelo de amenazas.
- ✓ Analizar las amenazas.
- ✓ Identificar las técnicas y tecnologías de mitigación
- ✓ Documentar el modelo de seguridad y las consideraciones de implementación.
- ✓ Implementar y probar las mitigaciones.
- ✓ Mantener el modelo de amenazas sincronizado con el diseño.

Requerimientos de seguridad en esta fase:

- ✓ Acceso a componentes y administración del sistema
- ✓ Pistas de auditoria
- ✓ Gestión de Sesiones
- ✓ Datos Históricos
- ✓ Manejo Apropiado de Errores
- ✓ Separación de Funciones (Segregación)

C. Seguridad en el Proceso de Codificación

Requerimientos de Seguridad en el Proceso de Codificación:

- ✓ Aseguramiento del ambiente de desarrollo.
- ✓ Elaboración de Documentación Técnica
- ✓ Codificación Segura
- ✓ Seguridad en las Comunicaciones
- ✓ Seguridad en promoción a ambientes de Producción

Buenas Prácticas de Codificación Segura:

- ✓ Orientación de arquitectura (por ejemplo "la capa Web no debe llamar a la base de datos directamente")
- ✓ Niveles mínimos de documentación requerida
- ✓ Requerimientos de testeo mandatarios
- ✓ Niveles mínimos de comentarios entre código y estilo de comentarios preferidos
- ✓ Manejo de excepciones
- ✓ Uso de flujo de bloques de control (por ejemplo "Todos los usos de flujos condicionales deben usar bloques de sentencias específicos")
- ✓ Método de nombramiento preferido para variables, funciones, clases y tablas.
- ✓ Código mantenible y legible es preferible ante un código complejo.

Los programadores, suelen seguir algunos patrones propios al realizar código. Lo que se busca en este punto es que el equipo de desarrolladores siga una serie de medidas comunes, como pueden ser el manual de código de buenas prácticas del CERT o de OWASP o las guías de safecode.

La revisión de código es una tarea fundamental dentro del S-SDLC ya que ayuda a detectar posibles fallos en el código elaborado. Se encuentran varias herramientas como Fortify, Moose, .NET Compiler Platform, entre otras. Dependiendo del proyecto software y el lenguaje de programación a utilizar se puede optar por la que se adapte mejor al proyecto.

D. Seguridad en el Proceso de Pruebas

Requerimientos de Seguridad en la Fase de Pruebas:

- ✓ Control de Calidad en Controles de Seguridad
- ✓ Inspección de Código por Fases
- ✓ Comprobación de Gestión de Configuraciones
- ✓ Caja Negra (Top Ten de OWASP, Guía de Pruebas)

Evaluación de Vulnerabilidades. Ayuda a identificar rápidamente y a tomar medidas contra los puntos más vulnerables del software, detectando las vulnerabilidades críticas que deben investigarse inmediatamente y los elementos informativos que presentan un riesgo menor. Existen proyectos como el de openVAS para realizar de manera automatizada estas pruebas.

Fuzzing. Se conoce como Fuzzing a la técnica de prueba de software que genera y envía datos secuenciales o aleatorios a una aplicación, con el objeto de detectar defectos o vulnerabilidades existentes. Con esta técnica se consigue ver las excepciones que devuelve la aplicación y los posibles problemas no contemplados. Se pueden detectar algunos stackoverflow o la reacción del programa al recibir campos incorrectos.

Existen proyectos que realizan este tipo de tests, como el de JBroFuzz de OWASP, que aunque actualmente está inactivo, se encuentra disponible para descargar.

E. Seguridad en el Proceso de Despliegue y Mantenimiento

Requisitos de Seguridad en Fase de Mantenimiento:

- ✓ Aseguramiento basado en riesgos.
- ✓ Pruebas de Seguridad (Caja Blanca y Caja Negra) después de los cambios.

Revisión de la configuración del Servidor. Antes de subir la aplicación a producción, se debe revisar que la configuración sea correcta a nivel de seguridad, evitando errores comunes como devolver más información de la debida en caso de error, pues un atacante puede conseguir información adicional que le ayude a preparar su ataque contra nuestra aplicación.

Revisión de la Configuración de la Red. Antes de subir la aplicación a producción, es necesario comprobar que la red que se va a utilizar sea segura. Por ejemplo, si se va a colocar la aplicación en una DMZ, se debe configurar adecuadamente, si se utiliza un servidor en la nube, es necesario configurar los patrones de conexión, para que no pueda acceder cualquier persona que tenga acceso a la red.

En la Fig. (2) podemos mirar el Framework que OWASP propone:

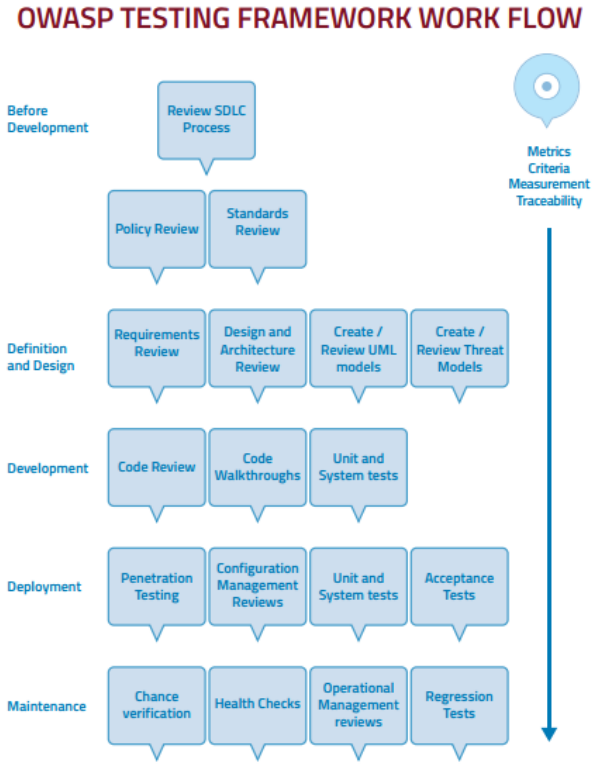

Figura 2. Owasp Testing Framework Work Flow. Fuente: OWASP Testing Guide V.4 [5]

III. GESTIÓN DE SEGURIDAD EN EL DESARROLLO DE SOFTWARE RESPECTO A LA NORMA TÉCNICA COLOMBIANA NTC-ISO-IEC 27001:2013

El Sistema de Gestión de Seguridad de la Información (SGSI), nos ayuda a preservar la integridad, confidencialidad y la disponibilidad de la información. La norma ISO/IEC 27001:2013, suministra los requisitos para establecer, implementar, mantener y mejorar el SGSI mediante la aplicación de un proceso de gestión de riesgo. En el anexo de esta norma encontramos los controles (ISO/IEC 27002:2013), dentro de los cuales nos centraremos en el A.14, adquisición, desarrollo y mantenimiento de los sistemas de información y los controles que aplican para el proceso de desarrollo de software.

A. Adquisición, Desarrollo y Mantenimiento de los Sistemas de Información.

Objetivo de Control. Requisitos de seguridad de los sistemas de información. Asegurar que la Seguridad de la Información sea una parte integral de los Sistemas de Información durante todo el ciclo de vida. Esto incluye también los requisitos para sistemas de información que prestan servicios sobre redes públicas.

Controles:

- ✓ Análisis y especificación de los requisitos de seguridad. Los requisitos relacionados con seguridad de la información se deben incluir en los requisitos para nuevos sistemas de información o para mejoras a los sistemas de información existentes.
- ✓ Seguridad de servicios de las aplicaciones en redes públicas. La información involucrada en los servicios de las aplicaciones pasan sobre redes públicas; se debe proteger de actividades fraudulentas, disputas contractuales y divulgación y modificación no autorizadas.
- ✓ Protección de las transacciones de los servicios de las aplicaciones. La información involucrada en las transacciones de los servicios de las aplicaciones se debe proteger para evitar la transmisión incompleta, el enrutamiento errado, la alteración no autorizada de mensajes, la divulgación no autorizada y la duplicación o reproducción de mensajes no autorizada.

Objetivo de Control. Seguridad en los Procesos de Desarrollo y Soporte. Asegurar que la seguridad de la información esté diseñada e implementada dentro del ciclo de vida de desarrollo de los sistemas de información.

Controles:

- ✓ Política de desarrollo seguro. Se deben establecer y aplicar reglas para el desarrollo de software y de sistemas, a los desarrollos dentro de la organización.
- ✓ Procedimientos de control de cambios en los sistemas. Los cambios a los sistemas dentro del ciclo de vida de desarrollo se deben controlar mediante el uso de procedimientos formales de control de cambios.
- ✓ Revisión técnica de las aplicaciones después de cambios en la plataforma de operación. Cuando se cambian las plataformas de operación, se deben revisar las aplicaciones críticas del negocio, y someter a prueba para asegurar que no haya impacto adverso en las operaciones o seguridad de la organización.
- ✓ Restricciones a los cambios en los paquetes de software. Se deben desalentar las modificaciones a los paquetes de software, los cuales se deben limitar a los cambios necesarios, y todos los cambios se deben controlar estrictamente.
- ✓ Principios de construcción de los sistemas seguros. Se deben establecer, documentar y mantener principios para la construcción de sistemas seguros, y aplicarlos a cualquier actividad de implementación de sistemas de información.
- ✓ Seguridad en entornos de desarrollo. Las organizaciones deben establecer y proteger adecuadamente los ambientes de desarrollo seguros para las actividades de desarrollo e integración de sistemas que comprendan todo el ciclo de vida de desarrollo de sistemas.
- ✓ Pruebas de seguridad de sistemas. Durante el desarrollo se deben llevar a cabo pruebas de funcionalidad de la seguridad [1].

Teniendo en cuenta los controles de la norma ISO 27001:2013, se destacan 10 controles correspondientes al dominio de adquisición, desarrollo y mantenimiento de los sistemas. Estos van a permitir un proceso de desarrollo de software seguro, que minimice el riesgo y garantice las mejores prácticas. Pero es importante establecer cómo se van a implementar estos controles. OWASP nos puede indicar un amplio panorama sobre esta práctica.

IV. ASPECTOS IMPORTANTES EN LA GESTIÓN DE SEGURIDAD

El Proyecto Open Web Application Security (OWASP) es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. Consta de una comunidad abierta dedicada a impulsar el desarrollo, compra y mantenimiento de aplicaciones confiables en las organizaciones. Ofrece herramientas importantes de documentación y software, que ayudan a tener las mejores prácticas para el desarrollo de software seguro.

Muchos de los controles contenidos en la Guía OWASP, se encuentran influenciados por requerimientos incluidos en estándares internacionales o marcos de control tales como Cobit e ISO27001; normalmente los controles seleccionados de la guía satisfarán los requerimientos relevantes de ISO 27001 o Cobit.

A. Principios de Seguridad:

- ✓ Minimizar el área de la superficie de ataque
- ✓ Seguridad por defecto
- ✓ Principio del mínimo privilegio
- ✓ Principio de defensa en profundidad
- ✓ Fallar de manera segura
- ✓ Los sistemas externos son inseguros
- ✓ Separación de funciones
- ✓ No confíes en la seguridad a través de la oscuridad
- ✓ Simplicidad
- ✓ Arreglar cuestiones de seguridad correctamente

B. Modelado de Riesgo de Amenaza:

Utilizando el proceso de modelado de Amenaza de Microsoft. Existen 5 pasos en el proceso de modelado, como se observa en la Fig. (3).

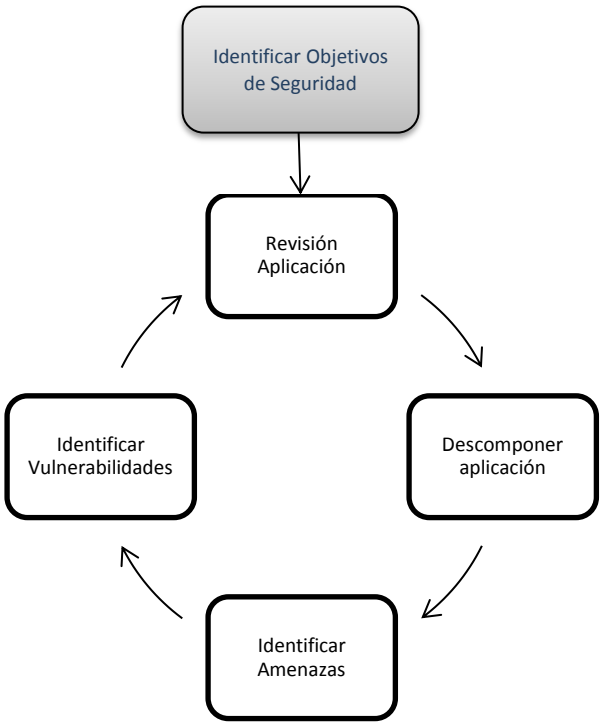

Figura 3. Flujo de Modelo de Amenaza [6]

Cuando se vaya a documentar una amenaza, Microsoft sugiere dos enfoques diferentes. Uno es un gráfico de amenaza y el otro es simplemente una lista estructurada. En la fig. 4 se puede observar un ejemplo de gráfico del árbol de amenaza.

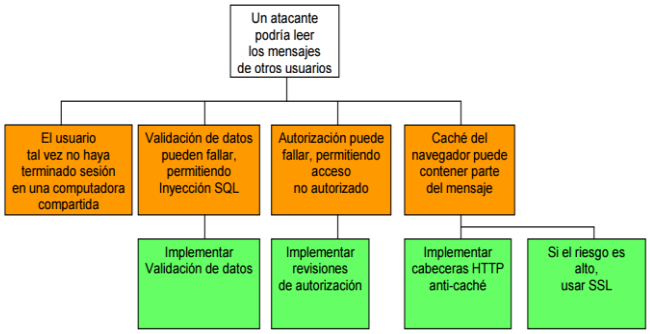

Figura 4. Gráfico de Árbol de Amenaza. Fuente: Guía OWASP [6].

En el proceso de identificación de riesgos se utiliza, *STRIDE*, es un acrónimo que resume 6 categorías de amenazas: *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, *and Elevation of Privilege*. Dado que cada categoría tiene un conjunto específico de medidas de seguridad que las puede evitar, una vez que se analizan las amenazas y se clasifican, es posible saber que será necesario para mitigarlas.

Para asignar valor al riesgo se encuentra, *DREAD*, que se utiliza para formar parte del razonamiento detrás de la clasificación de riesgos. Ayuda a calcular el valor de riesgo [4]. El cálculo del valor del riesgo lo podemos calcular mediante (1).

$$RiskDread = \frac{(D + R + E + U + De)}{5} \quad (1)$$

D – Daño potencial
R – Reproducibilidad
E – Explotación
U – Usuarios Afectados
De – Descubrimiento

La tabla I, a continuación muestra los valores a asignar, dependiendo del riesgo y su impacto.

DREAD	VALOR A ASIGNAR			
	0	5	10	
D: Si una amenaza es cumplida ¿cuánto daño se causa?	nada	Información individual comprometida	Destrucción total del sistema	
R: ¿Que tan fácil es reproducir la amenaza?	Muy difícil o imposible incluso para los administradores de la aplicación.	Uno o dos pasos requeridos, talvez necesite ser un usuario autorizado.	Requiere solo una barra de direcciones sin estar registrado en la aplicación.	
E: ¿Qué necesita tener para explotar esa amenaza?	Habilidades especializadas.	Malware existente.	Solo un navegador.	
U: ¿Cuántos usuarios serán afectados por esta amenaza?	Ninguno	Algunos usuarios	Todos los usuarios.	
De: ¿Qué tan fácil es descubrir esta amenaza?	Difícil a imposible, requiere acceso al sistema o al código.	Se podría dar con el problema de estar adivinando u observando las huellas de la red.	(Valor 9) Detalles de fallas como esta son del dominio público y pueden ser descubiertos utilizando google.	Está en la barra de direcciones o en una forma.

Tabla I. Dread: Valores Asignados. Fuente: Modelado de Riesgos de Amenaza – Guía OWASP [6].

El modelado de amenazas proporciona una "línea de visión" clara a través de un proyecto que justifica los esfuerzos de seguridad.

C. Manejando Pagos en el Comercio Electrónico

Objetivos a Asegurar:

- ✓ Manejar los pagos de una manera segura y equitativa de los usuarios de sistemas de comercio electrónico.
- ✓ Minimizar el fraude de los usuarios de tarjetas que no presencian transacciones.
- ✓ Maximizar la privacidad y confianza para los

usuarios de sistemas de comercio electrónico.
- ✓ Cumplir con todas las leyes locales y normas PCI (acuerdos de Mercado).

Buenas Prácticas:

- ✓ Procesar las transacciones online inmediatamente o pasar el procesamiento a una tercera parte competente.
- ✓ No almacenar ningún número de tarjeta de crédito. Si deben almacenarse, debe seguir las directivas de PCI al pie de la letra.
- ✓ Si usa un servidor compartido para el sitio, no puede cumplir con las directivas PCI. Debe contar con infraestructura propia para cumplir con las directivas PCI [5].

Phishing. Los ataques de *phishing* son uno de los mayores problemas para los sitios bancarios y de comercio electrónico, con el potencial de destruir los medios de subsistencia y calificaciones crediticias de un cliente. Existen algunas precauciones que los escritores de aplicaciones pueden seguir para reducir el riesgo, pero la mayoría de los controles de phishing son procesales y educación del usuario.

Medidas de prevención:

- ✓ Permitir al usuario reportar estafas
- ✓ Comunicación con el cliente utilizando el sitio web en lugar de correo electrónico.
- ✓ Establecer un esquema de políticas de remitentes (SPF) en el DNS para validar sus servidores SMTP.
- ✓ Considere utilizar S/Mime para firmar digitalmente sus comunicaciones.
- ✓ Nunca solicite a sus clientes información confidencial mediante correos o vía telefónica.
- ✓ Arregle todos sus problemas de XSS: no exponer código que contenga problemas de XSS, particularmente código no autenticado.
- ✓ No utilice ventanas emergentes.
- ✓ Utilice la directiva Target para crear una nueva ventana, la cual usualmente lo liberara de trampas en iframe y Javascript.
- ✓ Mueva su aplicación a un enlace de distancia de su página principal: Ubique al autenticador de su aplicación en una página separada.
- ✓ Imponga el uso de referencias locales para imágenes y otros recursos.
- ✓ Mantenga la barra de direcciones, utilice SSL, no utilice direcciones IP.
- ✓ No sea la fuente de robos de identidad
- ✓ Implemente protecciones dentro de su aplicación
- ✓ Monitoree actividad inusual en las cuentas.
- ✓ Ponga rápidamente fuera de línea los servidores victimas de phishing.
- ✓ Tome control de los nombres de dominio fraudulentos.
- ✓ Trabaje con las autoridades competentes.

Servicios Web. Los servicios web pueden ser vistos como aplicaciones web especializadas que difieren principalmente en la capa de presentación. Mientras que las aplicaciones web son típicamente basadas en HTML, los servicios web son basados en XML. Los servicios Web, como otras aplicaciones distribuidas, requieren protección en múltiples niveles:

- ✓ Los mensajes SOAP que son enviados en la red deben ser entregados confiablemente y sin modificaciones.
- ✓ El servidor necesita saber con confianza con quien está hablando y a que tienen derecho los clientes.
- ✓ Los clientes necesitan saber que están hablando al servidor correcto y no a un sitio de "phishing".
- ✓ El registro de eventos debe contener suficiente información para reconstruir confiablemente la cadena de eventos y rastrear de vuelta a los que llamaron funciones sin autentificación previa.

Estándar WS-Security. Las áreas principales, ampliamente definidas por el estándar son:

- ✓ Maneras de agregar encabezados de seguridad (encabezados WSSE) a los sobres de SOAP.
- ✓ Adjuntar testigos de seguridad y credenciales al mensaje
- ✓ Insertando un estampado de tiempo
- ✓ Firmar el mensaje
- ✓ Cifrado del mensaje
- ✓ Extensibilidad

El objetivo principal del estándar WSS es proveer herramientas para la protección de la comunicación a nivel mensaje. En la Fig. (5) se observa la Jerarquía de la especificación de WSS.

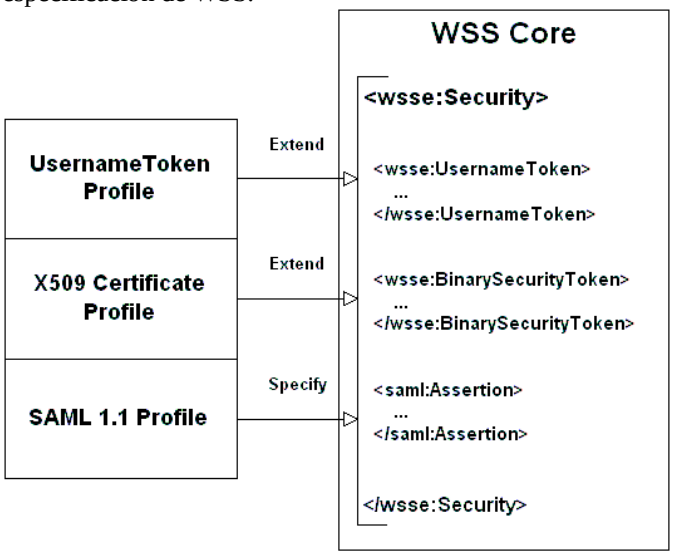

Figura 5. Jerarquía de la Especificación WSS. Fuente: Guía OWASP [5].

V. MECANISMOS, OBJETIVOS DE CONTROL Y CONTROLES

Para establecer los objetivos de control y controles necesarios, se han analizado los objetivos sugeridos por la guía de OWASP, teniendo en cuenta los controles que nos plantea la norma ISO/IEC 27002:2013 y Cobit.

A. Autenticación.

Proveer servicios de autenticación segura a las aplicaciones. Este objetivo lo podemos enmarcar en el objetivo de control de la Norma ISO/IEC 27002:2013 A.9.4. Control de Acceso a los Sistemas y Aplicaciones. Implementar los controles que nos ofrece la misma:

- ✓ Restricción del acceso a la información.
- ✓ Procedimientos seguros de inicio de sesión.
- ✓ Gestión de contraseñas de usuario.
- ✓ Uso de herramientas de administración de sistemas.
- ✓ Control de acceso al código fuente de los programas.

Además, Cobit nos ofrece un marco de trabajo en su Dominio DS5. Garantizar la Seguridad de los Sistemas y objetivo de control DS5.3. Gestión de Identidad. Las prácticas de control que nos sugiere Cobit son:

Establecer y comunicar las políticas y procedimientos para identificar de forma exclusiva, autenticar y autorizar los mecanismos de acceso y los derechos de acceso para todos los usuarios, sobre la base de los roles predeterminados y con aprobación previa. Establecer claramente la responsabilidad de cualquier usuario por cualquier acción en cualquiera de los sistemas y aplicaciones involucradas.

Asegúrese de que las funciones y criterios de autorización de acceso para la asignación de derechos de acceso de los usuarios tienen en cuenta:

- ✓ La sensibilidad de la información y las aplicaciones que se traten (clasificación de datos).
- ✓ Políticas para la protección y difusión de la información (como políticas internas de regulación legal y los requisitos contractuales).
- ✓ Los roles y responsabilidades definidas dentro de la empresa.
- ✓ Los derechos de acceso asociados con la función.
- ✓ Los perfiles de acceso de usuario estándar, pero individuales para los roles de trabajo comunes en la organización.
- ✓ Requisitos para garantizar la adecuada segregación de funciones.

Establecer un método para su autenticación y autorización para establecer la responsabilidad y hacer cumplir los derechos de acceso de acuerdo con la sensibilidad de la información y requisitos funcionales de la aplicación y componentes de infraestructura, de conformidad con las leyes aplicables, regulaciones, políticas internas y acuerdos contractuales.

Definir e implementar un procedimiento para identificar y registrar nuevos usuarios, para la autorización y el mantenimiento de los derechos de acceso. Esto tiene que ser solicitado por el administrador de usuarios, aprobado por el propietario de la red y ejecutado por la persona responsable de la seguridad.

Asegurarse de que el flujo de información es oportuno e informa de cambios en el trabajo. Revocar y adaptar los derechos de acceso de usuario en coordinación con recursos humanos para los usuarios que son nuevos y aquellos que han dejado la organización, o que han cambiado los roles o puestos de trabajo.

B. Autorización.

Sus objetivos se dirigen hacia asegurar que únicamente usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio. Además controlar el acceso a recursos protegidos mediante decisiones basadas en el rol o el nivel de privilegio y prevenir ataques de escalada de privilegios, como por ejemplo utilizar funciones administrativas siendo un usuario anónimo o incluso un usuario autenticado. El proceso de Cobit importante para la implementación de controles es el DS5. Debe tenerse en cuenta el principio del menor privilegio, las listas de control de acceso, los controles de autorización personalizados, las rutinas de autorización centralizadas, matriz de autorización, tokens de autorización del lado del cliente, control de acceso y protección a los recursos protegidos.

Un error común es el de llevar a cabo una comprobación de la autorización copiando y pegando un trozo de código, o incluso peor, reescribiéndola cada vez. Las aplicaciones bien desarrolladas centralizan las rutinas de control de acceso, sobre todo las de autorización, por lo que si se encuentra algún fallo, pueden arreglarse todas las ocurrencias sólo modificando el código una vez, aplicándose a todas las secciones al mismo tiempo.

C. Manejo de Sesiones.

En el manejo de sesiones básicamente se debe asegurar que los usuarios autenticados tengan una robusta y criptográficamente segura asociación con sus sesiones. Estas deben cumplir los controles de autorización y deben prevenir los típicos ataques web, tales como la reutilización, falsificación e intercepción de sesiones. Los temas relevantes de Cobit para este ítem son PO8 y PO8.4. Privacidad, Propiedad Intelectual y Flujo de Datos.

Se recomienda utilizar un robusto y bien conocido manejador de sesiones. Los marcos de aplicaciones más populares contienen una adecuada implementación. Sin embargo, las primeras versiones frecuentemente tienen debilidades significativas. Es mejor utilizar la última versión de tecnología elegida, ya que su manejador de sesiones probablemente será más robusto y usara credenciales criptográficas fuertes.

D. Validación de Datos.

El objetivo es garantizar que la aplicación sea robusta contra todas las formas de ingreso de datos, ya sea obtenida del usuario, de la infraestructura, de entidades externas o de sistemas de base de datos. Se puede evaluar frente al Dominio de Cobit DS11. Gestión de Datos.

Estrategias de Validación de Datos: Aceptar buenos conocidos, Desinfectar, prevenir manipulación de parámetros, cifrar URL y HTML, tener cuidado con las Listas de selección, botones de opción y cajas de verificación. Verificar el estado de la Vista de ASP.NET.

Se debe tener en cuenta la confidencialidad y la integridad de los datos.

E. Interprete de Inyección.

Garantizar que las aplicaciones sean seguras de ataques de manipulación de parámetros contra intérpretes comunes. Se

pueden encontrar ejemplos detallados para cada lenguaje y tipos de inyección de código en la guía de pruebas de OWASP V.4.

F. Canonicalización, locales y Unicode

Asegurar que la aplicación es robusta cuando está sujeta a valores de entrada codificados, internacionalizados o en Unicode. Cobit. D11.6 Requisitos de Seguridad para la Administración de los Datos.

G. Manejo de Errores, auditoría y Generación de Logs.

Muchas industrias son requeridas por medio de requisitos legales y regulatorios con lo siguiente:

- ✓ Auditable. Todas las actividades que afectan el estado de un usuario o balances deben ser formalmente rastreables.
- ✓ Trazable. Es posible determinar donde ocurre cada actividad en todas las capas de una aplicación.
- ✓ Alta integridad. Los logs no pueden ser sobrescritos o modificados por usuarios locales o remotos.

Las aplicaciones bien escritas generan logs de doble propósito con trazas de actividad para auditoria y monitoreo, y hace que sea fácil seguir una transacción sin mucho esfuerzo o acceso al sistema. Debería ser posible identificar y seguir una transacción potencialmente fraudulenta de punta a punta.

H. Sistema de Ficheros.

Asegurar que el acceso local al sistema de ficheros por algún sistema está protegido de creaciones, modificaciones o eliminaciones no autorizadas.

Controles y buenas prácticas:

- ✓ Definir e implementar procedimientos para el archivo, almacenamiento y retención de los datos, de forma efectiva y eficiente para conseguir los objetivos de negocio, la política de seguridad de la organización y los requerimientos regulatorios (Cobit DS11.2). [2].
- ✓ Definir e implementar procedimientos para asegurar que los requerimientos del negocio para la protección de datos sensibles y el software se consiguen cuando se eliminan o transfieren los datos y/o el hardware (DS11.4). [2].
- ✓ En ISO 27002, asociamos estos dos controles adicionales:
- ✓ Desarrollar e implementar una Política de control de accesos.
- ✓ Gestionar un control de acceso a las redes y servicios asociados.

Se pueden aplicar estos controles realizando las siguientes prácticas por ejemplo:

- ✓ Usar jaulas “chroot” en plataformas Unix.
- ✓ Usar los mínimos permisos en el sistema de ficheros de todas las plataformas.
- ✓ Considerar la utilización de sistemas de fichero de solo lectura (como CD-ROM o llaves USB bloqueadas) si fuera posible.[3]

Como protegerse:

- ✓ Asegurar o recomendar que el sistema operativo y entorno de aplicaciones web se mantenga al día.
- ✓ Asegurar que ficheros y recursos son de solo lectura.
- ✓ Asegurar que la aplicación no obtiene los nombres de fichero del usuario cuando los guarda o trabaja en ficheros locales.
- ✓ Asegurar que la aplicación comprueba correctamente la información enviada por el usuario para prevenir comandos que no deben ejecutarse.

I. Desbordamiento de memoria.

Los atacantes se valen de los desbordamientos de memoria para corromper la pila de ejecución de una aplicación.

El objetivo es lograr que las aplicaciones no se expongan a componentes defectuosos y tengan un manejo de memoria adecuado, utilizando el uso de lenguajes y *frameworks* que sean relativamente inmunes a desbordamientos de memoria.

Control:

- ✓ Se debe restringir y controlar estrictamente el uso de programas utilitarios que podrían tener la capacidad de anular el sistema y los controles de las aplicaciones.

Como protegerse:

- ✓ Despliegue su aplicación en sistemas capaces de evitar la ejecución de pilas.
- ✓ Valide los datos de entrada del usuario para prevenir campos demasiado largos y revise los valores para asegurar de que se encuentran dentro de una especificación.
- ✓ Si confía plenamente en su sistema operativo y en aplicaciones escritas en C o C++, asegúrese de que utilicen el principio de privilegios mínimos, utilice compiladores que lo puedan proteger contra desbordamientos de montículo y de pila y mantenga el sistema actualizado con los parches correspondientes.

J. Interfaces Administrativas.

Asegurar que las funciones de nivel de administrador están segregadas apropiadamente de la actividad del usuario y que estos no pueden acceder o utilizar funcionalidades administrativas. Se debe proveer trazabilidad de funcionalidad administrativa.

Como protegerse:

- ✓ Trazar la funcionalidad administrativa fuera y asegurarse que los controles apropiados de acceso y auditoria están en su lugar.
- ✓ Considerar procesos, en algunas ocasiones todo lo que se requiere es entender como los usuarios pueden ser prevenidos de utilizar una característica con la simple falta de acceso
- ✓ Acceso de servicio de asistencia es siempre un término medio de acceso para ayudar a los clientes, pero no son administradores.
- ✓ Diseñar cuidadosamente la funcionalidad de servicio de asistencia / moderador /soporte al cliente alrededor de una capacidad administrativa limitada y aplicación segregada o acceso.

K. Cifrado.

Asegurar que el cifrado es usado de manera segura para

proteger la confidencialidad e integridad de los datos sensibles de usuarios.

Controles:

- ✓ Se debe desarrollar e implementar una política que regule el uso de controles criptográficos para la protección de la información (ISO27002:2013, A.10.1)
- ✓ Es importante desarrollar e implementar una política sobre el uso, protección y tiempo de vida de las claves criptográficas a través de todo su ciclo de vida (ISO27002:2013, A.10.1).
- ✓ Determinar que las políticas y procedimientos para organizar la generación, cambio, revocación, destrucción, distribución, certificación, almacenamiento, captura, uso y archivo de llaves criptográficas estén implantadas, para garantizar la protección de las llaves contra modificaciones y divulgación no autorizadas.(Cobit, DS5.8).

L. Configuración.

Creación de aplicaciones que son seguras fuera de la máquina.

Controles. Establecer y mantener un repositorio de configuraciones completo y preciso. Una efectiva administración de la configuración facilita una mayor disponibilidad.

Controles Específicos:

- ✓ No distribuir ningún producto con cuentas preestablecidas.
- ✓ No incluir cuentas de respaldo o como puerta trasera, o mecanismos especiales de acceso.
- ✓ En la plataforma Win32, utilizar "TrustedConnection=yes", y crear un DSN con las credenciales almacenadas. Las credenciales son almacenadas como un LSA Secret, que no es perfecto, pero es mucho mejor que contraseñas en texto claro.
- ✓ Desarrollar un método para ofuscar la contraseña en algún formulario, como por ejemplo cifrar el nombre utilizando el nombre del sistema o similar con un código que no resulte obvio.
- ✓ Pedir al desarrollador de la base de datos que proporcione una biblioteca que permita conexiones remotas utilizando un hash de una contraseña en vez de una credencial en texto claro.
- ✓ Utilizar SSL, SSH y otros métodos de cifrado para evitar que la información sea interceptada en la transmisión.

Buenas prácticas:

- ✓ Desactivar todas las opciones innecesarias de manera predeterminada.
- ✓ Asegurar que todas las opciones y configuraciones para cada función están inicialmente configuradas para ser la elección más segura posible.
- ✓ Inspeccionar el diseño para comprobar si las elecciones menos seguras pudieran ser diseñadas de otra manera. Por ejemplo, los sistemas de restablecimiento de contraseña son pobres desde el punto de vista de seguridad. Si no proporciona este componente, los usuarios de su aplicación estarán más seguros.
- ✓ No confíe en características instaladas opcionalmente en el código base.
- ✓ No configure nada como preparación para una característica opcional de implantación.

M. Mantenimiento.

Asegurar que los productos son correctamente mantenidos después de su despliegue y que los defectos de seguridad son arreglados correctamente y en un tiempo adecuado. Minimizar el área de ataque atreves del ciclo de vida de producción.

Controles:

- ✓ Establecer procedimientos de configuración para soportar la gestión y rastro de todos los cambios al repositorio de configuración. Integrar estos procedimientos con la gestión de cambios, gestión de incidentes y procedimientos de gestión de problemas (Cobit, DS9.2).
- ✓ Controlar los cambios que afectan a la seguridad de la información en la Organización y procesos de negocio, las instalaciones y sistemas de procesamiento de información (ISO 27002:2013, A.12.1.2).

Específicos:

- ✓ Crear y mantener una política de mantenimiento de incidentes.
- ✓ Monitorizar las pruebas de vulnerabilidades.
- ✓ Monitorizar Bugtraq y listas de correo similares. Usar la experiencia de productos similares para aprender de sus errores y solucionarlos ante de encontrarlas en los productos propios.
- ✓ Publicar una sección de seguridad en el sitio web, con la posibilidad de enviar incidentes de seguridad de forma segura (como intercambio de claves PGP o vía SSL).
- ✓ Tener un método para solventar problemas de seguridad rápidamente, completamente probados en menos de 30 días.

N. Ataques de Denegación de Servicio

Asegurar que la aplicación es lo más robusta posible frente a ataques de denegación de servicio.

Controles:

- ✓ Uso de técnicas de seguridad y procedimientos de administración asociados (por ejemplo, firewalls, dispositivos de seguridad, segmentación de redes, y detección de intrusos) para autorizar acceso y controlar los flujos de información desde y hacia las redes (Cobit, DS5.10).
- ✓ Se deben identificar los mecanismos de seguridad, los niveles de servicio y los requisitos de gestión de todos los servicios de red e incluirlos en los acuerdos de servicio de red (ISO27002:2013, A.13.1.2).

Específicos:

- ✓ Permitir únicamente a los usuarios autenticados y autorizados el consumir peticiones de CPU significativas.
- ✓ Medir cuidadosamente los accesos a esos "cuellos de

botella", y recodificar o cambiar parámetros evitando que las peticiones básicas consuman demasiados recursos de Procesamiento.

VI. PRINCIPALES VULNERABILIDADES

El panorama de las amenazas para la seguridad de aplicaciones cambia constantemente. Los factores clave en esta evolución son los avances realizados por los atacantes, la liberación de las nuevas tecnologías con nuevas debilidades, así como integración en las defensas y el despliegue de sistemas cada vez más complejos. Para mantener el ritmo, el OWASP Top 10 actualiza de manera periódica. La última versión que se encuentra disponible es el OWASP Top Ten2013, aunque las vulnerabilidades técnicas pueden surgir nuevas cada año, esta es una importante guía para establecer las principales:

A. Inyección

Las fallas de inyección, tales como SQL, OS y la inyección LDAP, se producen cuando los datos que no son de confianza se envían a un intérprete como parte de un comando o consulta, estos datos son introducidos como una consulta por el atacante y puede engañar al intérprete para que ejecute comandos no deseados o acceder a los datos sin la debida autorización.

B. Autenticación Rota y Gestión de Sesiones

Funciones de aplicación relacionadas con la autenticación y gestión de sesiones con frecuencia no se aplican correctamente, permitiendo a los atacantes comprometer las contraseñas, claves o testigos de sesión, o para explotar otras fallas de implementación asumiendo la identidad de otros usuarios.

C. Cross-Site Scripting (XSS)

Las fallas de XSS ocurren cuando una aplicación toma los datos que no son de confianza y los envía a un navegador web sin la validación. XSS permite a los atacantes ejecutar secuencias de comandos en el navegador de la víctima, que pueden secuestrar sesiones de usuario, modificar sitios web, o redirigir al usuario a sitios maliciosos.

D. Referencias inseguras a objetos directos

Ocurre cuando un desarrollador expone una referencia a un objeto interno de la aplicación, tales como, un directorio o base de datos de archivos claves. Sin un control de accesos o de otro tipo de protección, los atacantes pueden manipular estas referencias para acceder a datos no autorizados.

E. Configuración errónea de Seguridad

Una buena seguridad requiere tener una configuración segura definida e implementada para la aplicación, *frameworks*, servidores, redes y plataformas. Estas configuraciones de seguridad deben ser definidas, implementadas y mantenidas. Los valores por defecto son a menudo bastante inseguros. Además, el software debe mantenerse actualizado.

F. Exposición de Datos Sensibles

Muchas aplicaciones no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, números de identificación fiscal y las credenciales de autenticación. Los atacantes pueden robar o modificar dichos datos débilmente protegidos para llevar a cabo el fraude de tarjetas de crédito, robo de identidad u otros delitos. Los datos sensibles requieren una protección adicional como el cifrado estático o durante el transporte, así como las precauciones especiales cuando se intercambian con el navegador.

G. Falta de funciones de nivel de Control de Acceso

La mayoría de las aplicaciones verifican los derechos de acceso y nivel de función antes de hacer esa característica visible en la interfaz de usuario. Sin embargo, las aplicaciones necesitan realizar las mismas comprobaciones de control de acceso en el servidor cuando se accede a cada función. Si las solicitudes no se verifican, los atacantes serán capaces de interceptar las solicitudes con el fin de acceder a las funciones sin la debida autorización.

H. Cross-Site Request Falsificación (CSRF)

Un ataque CSRF fuerza a la víctima a iniciar sesión en su navegador para enviar una petición HTTP forzada, incluyendo las cookies de sesión de la víctima y cualquier otra información de autenticación que se incluye de manera automática a una aplicación web vulnerable. Esto permite al atacante forzar el navegador de la víctima para generar solicitudes a la aplicación vulnerable para que el servidor piense que son las peticiones legítimas de la víctima.

I. Utilización de componentes vulnerables conocidos

Componentes, tales como bibliotecas, *frameworks* y otros módulos de software, casi siempre se ejecutan con privilegios completos. Si un componente vulnerable se explota, un ataque de este tipo puede facilitar la pérdida de datos importantes o toma de control del servidor. Las aplicaciones que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de la aplicación y permitir una variedad de posibles ataques e impactos.

J. Redirecciones y reenvíos no validados

Las aplicaciones Web frecuentemente redirigen y reenvían a los usuarios a otras páginas y sitios web utilizando datos no confiables para determinar las páginas de destino. Sin una validación adecuada, los atacantes pueden redirigir las víctimas a sitios de *phishing* o malware, o utilizar *forwards* (Peticiones del cliente) para acceder a páginas no autorizadas.

VII. NORMA ISO 27034

Proporciona una guía para las organizaciones en la integración de la seguridad en los procesos que se utilizan para la gestión de sus aplicaciones. Explícitamente toma un enfoque basado en procesos para la especificación, diseño, desarrollo, prueba, implementación y mantenimiento de las funciones de seguridad y controles en los sistemas de aplicación.

Define la seguridad de aplicaciones no como un estado de

seguridad, sino como "un proceso de una organización puede llevar a cabo para la aplicación de los controles y mediciones para sus aplicaciones con el fin de la gestión de dichos riesgos de su uso".

Diseñado para ser utilizado en conjunción con otras normas en la familia ISO27000.

Los principios fundamentales definidos por esta serie incluyen:

La Seguridad es un requisito: Los requisitos deben ser definidos y analizados para cada una de las etapas del ciclo de vida de la aplicación y se gestionan de forma continua.

La Seguridad de la Aplicación es dependiente del contexto: El tipo y el alcance de los requisitos de seguridad de las aplicaciones se ven influidas por los riesgos asociados a la aplicación: Riesgos del Negocio, normativas y regulación, y tecnológico.

Inversión Apropiada para la Seguridad de las Aplicaciones: Los costos para la aplicación de los controles de seguridad de software y la realización de mediciones de auditoría deben estar alineados con los objetivos del nivel de confianza del negocio.

La Seguridad de las Aplicaciones debe ser demostrada: El proceso de auditoría aprovecha la evidencia verificable proporcionada por los controles de Seguridad de la aplicación para confirmar si se han alcanzado los criterios del nivel de confianza.

VIII. CONCLUSIONES

Al iniciar un proyecto de Desarrollo de Software, es fundamental conocer que la seguridad es inherente a cada una de las etapas del Ciclo de Vida de Desarrollo de Software (SDLC), y en cada una de ellas existen amenazas de Seguridad, que se deben modelar e incorporar como requisitos adicionales para establecer los controles necesarios que aseguren la integridad, confidencialidad y disponibilidad de la información.

El OWASP, es un proyecto de seguridad para aplicaciones web bastante robusto que nos proporciona información detallada sobre herramientas, documentación, pruebas y controles que pueden utilizarse e implementarse para un proceso de desarrollo de software seguro.

La Gestión de la Seguridad y el Riesgo de la Información en el Ciclo de Vida del Desarrollo de Software, proporciona los controles necesarios al detectar de manera oportuna las vulnerabilidades en cada una de las fases que permite un proceso de desarrollo seguro, minimizando costos y asegurando el cumplimiento del proyecto.

Este documento brinda importantes aspectos y generalidades para la Administración de la Seguridad en el proceso de Desarrollo de Software, sin embargo es importante apoyarlo tomando como guía los principios y controles establecidos en la Norma ISO/IEC 27001:2013, ISO/IEC 27034-2:2015, El estándar Cobit y apoyándose en guía de pruebas que brinda el proyecto de Seguridad para aplicaciones Web OWASP.

REFERENCIAS

[1] Sistema de Gestión de Seguridad de la Información. ISO-IEC 27001. Primera Actualización, 2013.

[2] Information technology - Security techniques - Application security. ISO-IEC 27034. 2011.

[3] IT Governance Institute. COBIT V.4.1. 2007.

[4] Hacking Toys. (2014, Diciembre). S-SDLC: Aplicando seguridad al ciclo de vida del Software. [Online]. Available: http://wh0s.org/2014/12/07/s-sdlc-aplicando-seguridad-al-ciclo-de-vida-del-software/

[5] The Open Web Application security Project. Project Leaders. Meucci, Matteo. Muller, Andrew . The OWASP Foundation. Creative Commons 2.5 Licence. (2014) Testing Guide Foreword 4.0 Release.

[6] The Open Web Application security Project. Creative Commons License 3.0. (2014, Julio). Modelo de Amenazas. [Online]. Available: https://www.owasp.org/index.php/Modelado_de_Amenazas

[7] The Open Web Application security Project. Creative Commons License 3.0. (2008, Diciembre). CLASP Concepts. [Online]. Available: https://www.owasp.org/index.php/CLASP_Concepts

[8] The Open Web Application security Project. Creative Commons License 3.0. (2016, Febrero). OWASP CLASP Project. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_CLASP_Project

[9] The Open Web Application security Project. Creative Commons License 3.0. (2006, Junio). CLASP Best Practice. [Online]. Available: https://www.owasp.org/index.php/Category:CLASP_Best_Practice

[10] The Open Web Application security Project. Creative Commons License 3.0. (2008, Diciembre). CLASP Concepts. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[11] Microsoft Developer Network. (2002, Septiembre). How to Use the Encrypting File System. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms995356.aspx

[12] FLU-Project (2014, Mayo). Ciclos de Vida del Software Seguro. [Online]. Available: http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s_29.html

[13] FLU-Project (2014, Mayo). Ciclos de Vida del Software Seguro. [Online]. Available: http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s_19.html

[14] FLU-Project (2014, Mayo). Ciclos de Vida del Software Seguro. [Online]. Available: http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s.html

[15] International Organization for Standardization. (2015, Noviembre). ISO/IEC 27034-2:2015 (en) [Online]. Available: http://www.iso27001security.com/html/27034.html

[16] Smart Grid Interoperability Panel - Cyber Security Working Group (SGIP-CSWG). (2010, Septiembre). Introduction to NISTIR 7628 Guidelines for Smart Grid Cyber Security. [Online]. Available: http://www.nist.gov/smartgrid/upload/nistir-7628_total.pdf

[17] Information Security Standards ISO/IEC 27034:2011+ Information technology — Security techniques — Application security (parts 1 & 2 published, remainder in DRAFT). [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:27034:-2:ed-1:v1:en

[18] Beiter, Mike (2013, Noviembre). Steps in a Secure Software Development Lifecycle Model. https://www.michael.beiter.org/2013/11/29/steps-in-a-secure-software-development-lifecycle-model-1/

Autor

Yudy Amparo Narváez V. Especialización en Seguridad Informática, Universidad Piloto de Colombia, Especialista en Diseño de Videojuegos SENA, Ingeniera de Sistemas de la Universidad de Nariño. Research Group Galeras .NET Universidad de Nariño. CEO & Founder Software Design Solutions. Líder Proyecto de Innovación ELVES, Sistema de Entrenamiento de Escritura para Primera Infancia Basado en Sensores de Movimiento. Miembro red de Parthners de Microsoft.