SEGURIDAD PARA MINIMIZAR RIESGOS EN EL DESARROLLO DEL SOFTWARE

Vanegas Barrero, Alejandro
alejandrovanegasb@gmail.com
Universidad Piloto de Colombia

Abstract— This The software has an important place today, due to processing all kinds of information, connectivity and mobility are features that have increased the importance of software in the same way have increased the risks to which it is exposed information It handled by any software, so it is necessary to take into account security in each of the stages Life Cycle software Development (SDLC); Analysis and requirements definition, system design, coding, testing, deployment and maintenance, so you can minimize the risks of software.

Resumen— El software tiene un lugar importante en la actualidad, debido al procesamiento de todo tipo de información, la conectividad y movilidad son características que han incrementado la importancia del software, de la misma manera se han incrementado los riesgos a los que está expuesta la información que se maneja por cualquier software, por esto se hace necesario tener en cuenta la seguridad en cada una de las etapas del Ciclo de Vida del Desarrollo del Software (SDLC); Análisis y definición de requerimientos, Diseño del sistema, Codificación, Pruebas, Despliegue y mantenimiento, de esta manera se pueden minimizar los riesgos del software.

Índice de Términos— Ciclo de vida del desarrollo del software, Riesgos, Seguridad, Software.

I. INTRODUCCIÓN

El software genera valor para las organizaciones al mejorar y sostener su productividad de una manera eficiente, la complejidad e importancia de las aplicaciones en la organización aumenta cada día más por lo que es necesario garantizar la seguridad para que la organización y sus servicios continúen operando de la manera esperada aun cuando se presenten eventos que tengan objetivos negativos sobre la organización.

Debido a la importancia del software en las organizaciones, es importante que se tenga en cuenta la seguridad durante todas las etapas del ciclo de vida del desarrollo del software (SDLC).

La importancia de incluir seguridad durante todo el ciclo de vida de desarrollo de software debería ser la misma que la de garantizar la calidad en el software, actualmente el inconveniente es que las organizaciones se preocupan por la seguridad de las aplicaciones solo cuando se ha implementado.

II. CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

El ciclo de vida del software es la sucesión de etapas por las que pasa el software desde que un nuevo proyecto es concebido hasta que se deja de usar, estas etapas representan el ciclo de actividades involucradas en el desarrollo, uso y mantenimiento de software, además de llevar asociadas una serie de documentos que serán la salida de cada una de estas fases y servirán de entrada en la fase siguiente; existen generalmente cinco etapas en los diferentes modelos de desarrollo de software: [1]

A. *Análisis y definición de requerimientos:* en esta etapa, se establecen los requerimientos del producto que se desea desarrollar. Éstos consisten usualmente en los servicios que debe proveer, limitaciones y metas del software, una vez que se ha establecido esto, los requerimientos deben ser definidos en una manera apropiada para ser útiles en la siguiente etapa, esta etapa incluye también un estudio de la factibilidad y viabilidad del proyecto con el fin de determinar la conveniencia de la puesta en marcha del proceso de desarrollo, puede ser tomada como la concepción de un producto de software y ser vista como el comienzo del ciclo de vida.

B. *Diseño del sistema:* el diseño del software es un proceso que se centra en cuatro atributos

diferentes de los programas: estructura de datos, arquitectura del software, detalle del proceso y caracterización de las interfaces, el proceso de diseño representa los requerimientos en una forma que permita la codificación del producto (además de una evaluación de la calidad previa a la etapa de codificación), al igual que los requerimientos, el diseño es documentado y se convierte en parte del producto de software.

C. *Codificación:* esta es la etapa en la cual son creados los programas. Si el diseño posee un nivel de detalle alto, la etapa de codificación puede implementarse mecánicamente, a menudo suele incluirse un testeo unitario en esta etapa, es decir, las unidades de código producidas son evaluadas individualmente antes de pasar a la etapa de integración y testeo global.

D. *Pruebas:* una vez concluida la codificación, comienza el testeo del programa, el proceso de testeo se centra en dos puntos principales: las lógicas internas del software; y las funcionalidades externas, es decir, se solucionan errores de "comportamiento" del software y se asegura que las entradas definidas producen resultados reales que coinciden con los requerimientos especificados.

E. *Despliegue y* M*antenimiento:* el despliegue comienza cuando el código ha sido suficientemente probado, aprobado para su liberación y ha sido distribuido en el entorno de producción, el mantenimiento consiste en la corrección de errores que no fueron previamente detectados, mejoras funcionales y de performance, y otros tipos de soporte, la etapa de mantenimiento es parte del ciclo de vida del producto de software y no pertenece estrictamente al desarrollo. Sin embargo, mejoras y correcciones pueden ser consideradas como parte del desarrollo.

Existen diferentes modelos de ciclo de vida de desarrollo del software, todos ellos buscan cumplir principalmente con los requerimientos funcionales y no funcionales solicitados.

Actualmente los requerimientos de seguridad son de gran importancia debido al aumento de ataques maliciosos, no solo a la infraestructura tecnológica de la organización sino también a su software.

III. SEGURIDAD EN EL CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

Debido a la explotación de vulnerabilidades en el software, que ocasionan perdida de información confidencial, daño del buen nombre de la organización, no disponibilidad de servicios, etc.

Las organizaciones le han dado más importancia a la seguridad desde la creación del software para que este continúe funcionando correctamente después de cualquier ataque malicioso.

El costo de solucionar las vulnerabilidades es mayor cuanto más tarde se detectan en el SDLC, lo que quiere decir que es mucho menos costoso construir software seguro que corregir problemas de seguridad cuando ha sido completado, sin mencionar los costos que pueden estar asociados a un quiebre en la seguridad [2], como se observa en la Fig. 1.

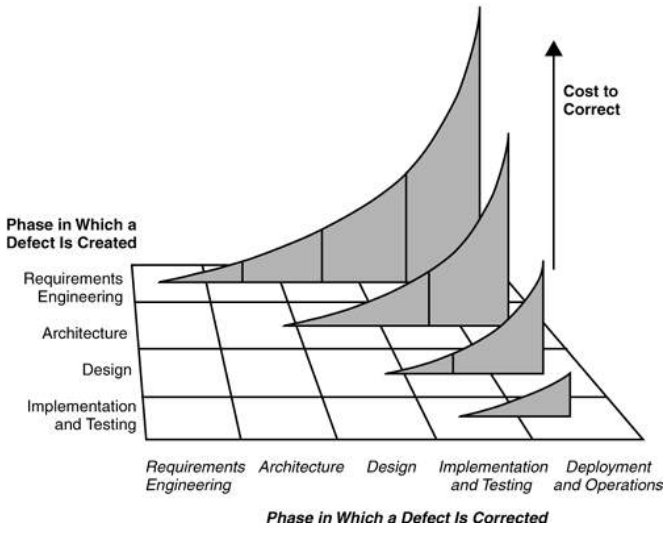

Figura 1. Costo de corregir los defectos de la fase del ciclo de vida. [3]

Es importante incorporar seguridad a lo largo de todas las etapas del ciclo de vida del desarrollo del software: [4]

- Análisis y definición de requerimientos

- Diseño del sistema
- Codificación
- Pruebas
- Despliegue y Mantenimiento

Incorporando seguridad en cada una de las capas del ciclo de vida de desarrollo del software, tratamos de garantizar las propiedades principales de seguridad del software: [5]

A. *Confidencialidad:* el software no debe permitir que entidades no autorizadas tengan acceso al ambiente de ejecución, usuarios, código fuente, contenidos y activos relacionados.

B. *Integridad:* el software debe ser capaz de proteger sus diferentes componentes contra las modificaciones no autorizadas, destrucción, inserción de código malicioso, se debe prevenir todo lo que afecte la integridad durante el desarrollo y ejecución del software.

C. *Disponibilidad: e*l software debe ser funcional y accesible por todos los usuarios y servicios y procesos autorizados en cualquier momento.

Se deben considerar las siguientes propiedades como parte fundamental para garantizar la protección de la confidencialidad, integridad y disponibilidad:

D. *Identificación, autenticación y autorización:*

Identificación: la identificación es el primer paso en la secuencia de identificar, autenticarse y autorizar, que es realizado todos los días innumerables veces por los seres humanos y computadoras, es cuando el usuario se presenta ante la aplicación.

Autenticación: la autenticación se presenta después de la identificación y antes de la autorización, verifica la autenticidad de la identidad declarada en la etapa de identificación, es decir, es en la fase de autenticación que se prueba que es la persona o el sistema que dice ser. Los tres métodos de autenticación son:

Lo que sabes: entre lo que sabes se encuentran métodos de autenticación como contraseñas, frases de acceso secretas, códigos y números de identificación personal (PIN).
Lo que tienes: tal vez el más ampliamente utilizado y familiar lo que tienes son los métodos de autenticación como llaves que utilizamos para bloquear y desbloquear las puertas, los coches, y los cajones, este método de autenticación en sistemas de información implica que si usted posee algún elemento, tal como una tarjeta inteligente o un token usb, usted es la persona que está que dice ser.
Lo que eres: lo que es, se refiere a los métodos de autenticación biométrica, un biométrico es una característica fisiológica o del comportamiento de un ser humano que puede distinguir una persona de otra y que se puede utilizar para la identificación o la verificación de la identidad.

Autorización: después de declarar la identidad en la fase de identificación y demostrarlo en la etapa de autenticación, a los usuarios se les asigna un conjunto de autorizaciones (derechos, privilegios o permisos) que definen lo que pueden hacer en el sistema.

E. *Responsabilidad*: la rendición de cuentas es otro principio de la seguridad del software, todas las acciones y eventos importantes de los usuarios se deben poder rastrear y almacenar, para determinar responsabilidad por las acciones u omisiones.

F. No repudio: evitar el repudio garantiza que ninguna de las partes relacionadas al software (usuarios) pueda negar operaciones realizadas.

IV. SEGURIDAD EN EL ANÁLISIS Y DEFINICIÓN DE REQUERIMIENTOS

Al igual que se definen los requerimientos funcionales, se deben definir los requerimientos de seguridad que garanticen que garanticen la confidencialidad, integridad y disponibilidad ante ataques maliciosos.

En el análisis de requerimientos de seguridad, se deben tener en cuenta lo siguiente: [4]

- Arquitectura de la aplicación: Cliente/servidor o Desktop.
- Plataforma donde correrá la aplicación: PC / Teléfono celular.
- Tipos de datos que se almacenan o transfieren: Confidenciales / públicos.
- Requerimiento que cumplan con normativas y marcos regulatorios.
- Tipos de registro que el sistema debe generar: Acceso a recursos, uso de privilegios, etc.
- Perfiles de usuario necesarios para la aplicación: Administrador, revisor, editor, usuario básico, etc.
- Tipos de acceso a los datos por parte de cada perfil: Lectura, escritura, modificación, agregado, borrado, etc.
- Acciones sobre el sistema que puede hacer cada perfil: Cambiar la configuración del sistema, Arrancar o detener servicios
- Modos de autenticación: Passwords, Tokens, Biométricos.

V. SEGURIDAD EN EL DISEÑO DEL SISTEMA

La etapa del diseño es donde se define como se construirá el software, como se cumplirá con los requerimientos funcionales y de seguridad establecidos en la etapa de análisis y definición del requerimiento.

Algunos principios de seguridad que se deben tener en cuenta en esta etapa de diseño del sistema son:[4]

- Reducción de Superficie de ataque.
- Criterio del menor privilegio.
- Criterio de defensa en profundidad.
- Diseño seguro de manejo de errores.
- Diseño seguro de autenticación.
- Separación de privilegios.
- Interacción “amigable” con Firewalls e IDS's.
- Administración segura información Sensible.
- Diseño de Auditoría y Logging.
- Análisis de riesgo.

Algunas técnicas que se podrían tener en cuenta en esta etapa del ciclo de vida del desarrollo del software para garantizar seguridad son:

A. Análisis de riesgo – Threat Modeling: técnica formal, estructurada y repetible que permite determinar y ponderar los riesgos y amenazas a los que estará expuesta nuestra aplicación, consta de los siguientes pasos:

- Conformar un grupo de análisis de riesgos.
- Descomponer la aplicación e identificar componentes clave.
- Determinar las amenazas a cada componente de la aplicación.
- Asignar un valor a cada amenaza.
- Decidir cómo responder a las amenazas.
- Identificar las técnicas y tecnologías necesarias para mitigar los riesgos identificados.

B. Método STRIDE: ayuda a identificar amenazas en los componentes de un sistema, su nombre es un acrónimo de:

- Spoofing Identity: suplantar la identidad de otro usuario o servicio.
- Tampering with Data: modificar maliciosamente datos almacenados.
- Repudiation: Imposibilidad de identificar el autor de una acción.
- Information Disclosure: divulgar información a usuarios no autorizados.

- Denial of Service: provocar que un servicio deje de funcionar.
- Elevation of privilege: conseguir privilegios mayores a los asignados.

C. Método DREAD: ayuda a ponderar las amenazas identificadas, su nombre es un acrónimo de los siguientes pasos:

- Damage Potencial: ¿cuán importante es el daño de esta amenaza?
- Reproducibility: ¿cuán reproducible es la vulnerabilidad?
- Exploitability: ¿cuán fácil es de explotar?
- Affected Users: ¿cuáles y cuántos usuarios se verían afectados?
- Discoverability: ¿cuán fácil de descubrir es la vulnerabilidad?

VI. SEGURIDAD EN LA CODIFICACIÓN

En esta etapa de codificación o construcción del software, se deberían seguir los lineamientos de la etapa de diseño, el tiempo en esta etapa está relacionado a que también definido este el diseño.

Durante esta etapa se deben tener en cuenta los siguientes aspectos para la seguridad en el software:[4]

- Validar siempre los datos de entrada antes de procesarlos.
- Nunca confiar en que los datos recibidos sean correctos.
- Realizar validación de datos en todas las capas.
- Usar siempre criterio de "White List" en las validaciones.
- Controlar tamaño y tipo de datos.
- Eliminar o "escapear" caracteres especiales.
- Reemplazar sentencias SQL dinámicas por Stored Procedures.
- Evitar generar código con valores ingresados por el usuario.
- No mezclar datos con código.
- Capturar errores de capas inferiores y no mostrarlos al usuario.

VII. PRUEBAS DE SEGURIDAD

Las pruebas de seguridad del software buscan confirmar que se cumplieron los requerimientos de seguridad establecidos en la etapa de análisis y definición de requerimientos, por los controles definidos en la etapa de diseño del sistema.

Algunas técnicas de pruebas de seguridad son: [4]

A. Testing de seguridad funcional: aplicado a las funcionalidades de seguridad de una aplicación como por ejemplo:

- Autenticación.
- Complejidad de contraseñas.
- Bloqueo automático de cuentas.
- Funcionalidad de captchas.
- Restricciones de acceso según diseño.
- Mecanismos de registro y logging.
- Mensajes de error especificados.

*B. Testing de seguridad basado en Riesgo: t*écnica que se desprende del Threat Modelling.

C. Test de stress: consiste en llevar la carga o funcionalidad de la aplicación al límite.

- Generar una carga alta de peticiones/transacciones a la aplicación.
- Mantener esta carga durante tiempos prolongados.
- Simular tráfico en ráfagas.

D. Test de mutación de datos: se prueba la aplicación ingresando en sus interfaces datos "mutados":

- Diferente signo.
- Diferente tipo.
- Diferente longitud.
- Fuera de rango.
- Caracteres especiales.
- Código (ej: javascripts).
- Valores nulos.
- Valores aleatorios.

VIII. SEGURIDAD EN LA IMPLEMENTACIÓN

En esta etapa de implementación del software en el ambiente de producción se debe verificar que la línea base de configuración sea la misma sobre la cual se desarrolló el software.

Si no se implementa la aplicación de forma segura, se pueden perder el tiempo consumido en cada etapa del ciclo de vida del software, se deben tener en cuenta los siguientes aspectos: [4]

A. *Hardening de software de base.*
B. *Proceso de implementación - Separación de ambientes.*
C. *Administración de implementación y mantenimiento:* Releases, Parches y Firma de código.

IX. ESTÁNDARES Y NORMAS

Algunos estándares, modelos y buenas prácticas para el desarrollo de software seguro que podríamos tener en cuenta:

A. *ISO / IEC 27034-1:* proporciona orientación sobre el diseño, selección, especificación y aplicación de los controles de seguridad de la información mediante un conjunto de procesos que están integrados a través del Desarrollo de Sistemas de Ciclo de una organización (SDLC). [6]

B. *ISO / IEC 12207:* establece un marco común para los procesos del ciclo de vida del software, con la terminología bien definida, que puede ser referenciado por la industria del software. Contiene los procesos, actividades y tareas que se van a aplicar durante la adquisición de un producto de software o servicio y durante el suministro, desarrollo, operación, mantenimiento y eliminación de productos de software.[7]

C. *Microsoft Trustworthy Computing SDL:* describe el ciclo de vida de desarrollo de seguridad, un proceso que Microsoft utiliza para desarrollar software que pueda resistir ataques malintencionados. [8]

D. *OWASP CLASP (Proceso de seguridad en aplicación completo y ligero):* proporciona un enfoque bien organizado y estructurado para mover las inquietudes de seguridad a las fases iniciales del ciclo de vida de desarrollo de software, cuando esto sea posible. [9]

E. *OWASP Development Guide:* establece estándares de codificación segura, proporciona una guía práctica e incluye J2EE, ASP.NET, y ejemplos de código PHP, abarca una amplia gama de problemas de seguridad a nivel de aplicación, a partir de la inyección de SQL a través de las preocupaciones modernas como el phishing, manejo de tarjetas de crédito, la fijación de sesión, peticiones falsas de cross-site, el cumplimiento y cuestiones de privacidad.[10]

F. *OWASP Enterprise Security API (ESAPI):* es una colección gratis y abierta de todos los métodos de seguridad que un desarrollador necesita para construir una aplicación Web segura. [11]

G. *OWASP Application Security Verification Standard (ASVS):* proporciona una base para los controles de seguridad técnica de aplicaciones web pruebas y también ofrece a los desarrolladores una lista de requerimientos para el desarrollo seguro. [12]

H. *Security Requirements Engineering Process (SREP):* es un proceso basado en activos y dirigido por el riesgo para el establecimiento de requerimientos de seguridad en el desarrollo de SI seguros. [13]

X. CONCLUSIONES

La seguridad del software se debe contemplar desde el inicio del ciclo de vida del desarrollo del software, para definir los requerimientos de seguridad que mitigaran los riesgos que se pueden

presentar al ser implementado en un ambiente de producción.
Se deben definir y documentar en la organización los estándares y buenas prácticas que se van a seguir durante el ciclo de vida del desarrollo del software.

El área de seguridad de la información debería tener participación durante el ciclo de vida del desarrollo del software.

Los integrantes del equipo de desarrollo de software deben recibir capacitación sobre las buenas prácticas y estándares de seguridad que sigue la organización.

Para desarrollar software con un nivel aceptable de seguridad, se debería por lo menos tener en cuenta aspectos de seguridad en las etapas: *análisis y definición de requerimientos, pruebas* (ya que usualmente solo se tienen en cuenta las pruebas funcionales) *y despliegue e implementación*.

XI. REFERENCIAS

[1] Copyright © aposta.uv.es. SISTEMAS SOFTWARE. [Online] Disponible: http://aposta.uv.es/givaro/modulo/Ciclo.htm

[2] The OWASP Foundation. OWASP Secure Coding Practices Quick Reference Guide. [Online] Disponible: https://www.owasp.org/images/a/aa/OWASP_SCP_Quick_Reference_Guide_SPA.pdf

[3] J. H. Allen et al. "Software Security Engineering: A guide for Project Managers". Addison-Wesley. 2008.

[4] P. Milano. (2007, Sep 12). Seguridad en el ciclo de vida del desarrollo de software [Online]. Disponible: http://www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

[5] Copyright © CRYPTOME (2013). Fundamental Security Concepts [Online]. Disponible: https://cryptome.org/2013/09/infosecurity-cert.pdf

[6] the International Organization for Standardization. ISO/IEC 27034-1:2011. [Online] Disponible: https://www.iso.org/obp/ui/#iso:std:iso-iec:27034:-1:ed-1:v1:en

[7] the International Organization for Standardization. ISO/IEC 12207:2008. [Online] Disponible: http://www.iso.org/iso/catalogue_detail?csnumber=43447

[8] Microsoft Corporation. El ciclo de vida de desarrollo de seguridad de Trustworthy Computing. [Online] Disponible: https://msdn.microsoft.com/es-es/library/ms995349.aspx

[9] The OWASP Foundation. OWASP CLASP Project. [Online] Disponible: https://www.owasp.org/index.php/Category:OWASP_CLASP_Project/es

[10] The OWASP Foundation. OWASP Development Guide. [Online] Disponible: https://www.owasp.org/index.php/Projects/OWASP_Development_Guide

[11] The OWASP Foundation. OWASP Enterprise Security API. [Online] Disponible: https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API/es

[12] The OWASP Foundation. OWASP Application Security Verification Standard. [Online] Disponible: https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

[13] D. Mellado, E. Fernández-Medina, and M. Piattini, A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems, in Computer Standards and Interfaces. 2006. [Online] Disponible: http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol05/vol5issue4July2007/5TLA4_03Mellado.pdf