Seguridad en el desarrollo del software

Marlon Estuardo Rodríguez
marlonrod.vas@hotmail.com
Estudiante de Ingeniería en Ciencias y Sistemas - USAC
Palabras claves: Ciclo, Software, Desarrollo, Seguridad, Análisis, Diseño, Sistemas.

Ciertamente, la seguridad en los productos de software establece una participación con un nivel ascendente a lo largo del proceso de desarrollo, desde su inicio hasta su fin, quiere decir en todo su ciclo de vida. Al hablar sobre la seguridad del software, debemos idealizar el sistema, acaparando el diseño, la codificación y el fortalecimiento del mismo. Con esto establecemos que la seguridad es una propiedad dinámica que varía en el tiempo, esta seguridad en los sistemas resultan críticos si consideramos la relación en la sociedad moderna.

Cabe destacar que gran porcentaje de las vulnerabilidades tienen su origen en fallas del diseño, estas fallas se van aumentando en desarrollo del software y tienen un impacto tan profundo en el sistema que demanda la reingeniería del mismo. Para esto se debe desplegar los recursos necesarios y fijar un buen diseño al inicio del ciclo de vida del software a fin de reducir costos que puedan producir las fallas de seguridad en el producto de software.

La seguridad en cada uno de los procesos del software es muy importante, ya que este marca el destino del software al momento de ser desplegado. Las fallas en seguridad en el software afectan drásticamente su funcionamiento, marcando grandes problemas que muchas veces pueden ser remediados, dejan una mala imagen, para la organización que desarrollo dicho software. Con el fin de implementar buenas prácticas en cada una de las fases del desarrollo del software, muchos Ingenieros en Sistemas computacionales se han esmerado mucho para mitigar por medio de procesos y estándares estos fallos que tienen consecuencias graves.

Al momento de desarrollar software se debe tener en cuenta muchos puntos importantes, para no tener problemas en el ciclo de vida del mismo. En este artículo nos enfocaremos en el uso de los principios y/o buenas prácticas de seguridad durante el ciclo de vida del Software.

Entendemos por ciclo de vida del software como los pasos que son necesarios para su desarrollo, con esto se lleva un mejor control en el proceso. Estos pasos se definen de la siguiente manera:

- **Análisis de los requisitos:** En esto paso se describen los requisitos del cliente, para así examinarlos y reestructurar para comprenderlos y absorber cada uno de los detalles que con llevan los mismos.
- **Diseño:** Al momento de comprender cada uno de los requisitos del cliente, el analista de software con su equipo de desarrollo deriva la estructuración del sistema, todas las interfaces y los datos que comprenden el software.
- **Codificación:** El paso de codificación conlleva la programación del sistema, teniendo en cuenta todos los requisitos del cliente ya bien diseñados y comprendidos.
- **Validación:** En esta parte del ciclo de vida se prueban cada una de las funcionalidades del programa, esto con el fin de evitar errores que se hayan codificado al momento de su desarrollo, con esto se asegura el buen funcionamiento del mismo.
- **Documentación:** Acá documentamos toda la información relevante del software, para así ayudar al usuario a comprenderlo.

Una vez comprendidos estos pasos nos adentraremos más en las características de seguridad de cada uno de ellos. Para esto establecemos requerimientos y controles de seguridad en el ciclo de vida de desarrollo de software:

Análisis de los requerimientos

Control de autenticación: La autenticación es el proceso de identificación de un individuo sobre la base de sus credenciales. El objetivo de la autenticación es decidir si alguien es quien dice ser. Hay tres formas de reconocer a un usuario, que se conocen como factores:

- Algo que saben, como una contraseña o PIN.
- Algo que tienen, tal como la licencia de conducir o tarjeta de crédito.
- Algo que son, como las huellas digitales o la inserción de los patrones.

El control de acceso es el proceso de decidir si el usuario

tiene permiso para ejecutar algo o no.

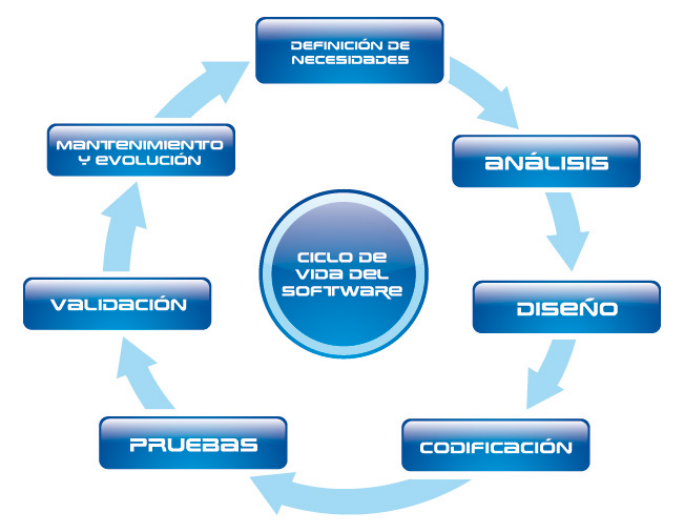
Control de Roles y Privilegios: También llamado autorización, se refiere a la gestión de acceso a los recursos protegidos y al proceso de determinar si un usuario está autorizado a acceder a un recurso particular. Muchas aplicaciones cuentan con recursos que sólo están disponibles para los usuarios autenticados, recursos que sólo están disponibles para los administradores, y los recursos que están disponibles para todos. Así, al establecer privilegios de acceso a los usuarios podemos asegurar la confidencialidad y disponibilidad de la información; pero, además podemos:

- Asegurarnos que sólo las personas autorizadas podrán acceder a ciertos recursos por sus funciones laborales.
- Nos permiten identificar y auditar los accesos realizados, estableciendo controles de seguridad internos.
- Documentar los procedimientos de acceso a las diferentes aplicaciones que tratan datos personales.
- Controlar los accesos desde diferentes vistas: red, sistemas y aplicaciones.

Requerimientos Orientados al riesgo: Acá se procura formalizar conocimientos orientados a la minimización o evitación de riesgos en proyectos de desarrollo de software, mediante la generación de principios y buenas prácticas de aplicación realista.

Planteamos según Robert Charrete que:

- El riesgo afecta a los futuros acontecimientos.



Ciclo de vida de Software. (6)

- El riesgo implica cambios.

- El riesgo implica elección, y la incertidumbre que entraña esta.

Es indiscutible que están presentes permanentemente las características de incertidumbre (acontecimiento que caracteriza al riesgo y que puede o no ocurrir) y de pérdida (si el riesgo se convierte en una realidad ocurrirán consecuencias no deseables o pérdidas).

Así mismo definimos las categorías de riesgos de la siguiente manera:

- Riesgos del proyecto: Que amenazan el plan.
- Riesgos técnicos: Que amenazan la calidad y la planificación temporal.
- Riesgos del negocio: Que amenazan la viabilidad del proyecto o del producto.
- Riesgos conocidos: Los que se descubren en las evaluaciones.
- Riesgos predecibles: Se extrapolan de la experiencia.
- Riesgos impredecibles: Pueden ocurrir, pero es muy difícil identificarlos de antemano.

A continuación se listaran las cuatro etapas en la administración o gestión de riesgos, el cual es un proceso iterativo que se aplica durante todo el proyecto.

- Identificación de riesgos: Listado de riesgos potenciales.
- Análisis de riesgos: Listado de priorización de riesgos.
- Planificación de riesgos: Anulación de riesgos y planes de contingencia.
- Supervisión de riesgos: Valoración de riesgos.

Diseño

Acceso a recursos y Administración del sistema: La razón principal para las cuentas de usuario es verificar la identidad de cada individuo. Otra razón muy importante es la de permitir la utilización personalizada de recursos y privilegios de acceso.

Los recursos incluyen archivos, directorios y dispositivos. A menudo el acceso a un recurso es controlado por grupos, los grupos son construcciones lógicas que se pueden utilizar para enlazar a usuarios para un propósito común.

Ejemplifiquemos este tema, si una organización tiene varios administradores de sistemas, todos ellos se pueden colocar en un grupo administrador del sistema. Luego se le pueden dar permisos al grupo para acceder a recursos claves del sistema.

Además vamos a manejar los nombres de usuario y contraseñas, las cuales nos indica quien está accediendo al sistema, para esto vemos que los nombres de usuario deben tener una característica principal, el indicar que deben ser únicos. Así mismo para garantizar una buena seguridad se debe forzar al usuario a tener contraseñas robustas y largas, con esto habrá menos probabilidad de que tenga éxito un ataque de fuerza bruta.

Auditoria en el software: Con esto nos enfocamos en auditar el software que se está desarrollando, esto significa realizar un control de todos los componentes de la empresa y ver cuál es la situación actual en referencia a las novedades y la legalidad de los programas utilizados, así como evaluar la seguridad en cada uno de los componentes del software que se está desarrollando.

Con esto se busca la actualización continua de los programas y equipos tecnológicos de las empresas y ver si se corresponden con la normativa legal establecida en el entorno en el que se mueve.

Manejo apropiado de errores: La fiabilidad y seguridad son requisitos usualmente mucho más rigurosos para sistemas incrustados que para el resto de sistemas computacionales. Podemos destacar que un sistema es fiable si cumple sus especificaciones; es seguro si no pueden producir situaciones que causen daños, independientemente de que se cumpla la especificación o no.

Evitar fallos implica limitar la introducción de código potencialmente defectuoso en la construcción del sistema. Para mejorar esto se puede tomar las siguientes medidas:

- Especificaciones rigurosas de requisitos.
- El uso de buenas metodologías de diseño.
- El uso de lenguajes con facilidades para modularidad y abstracción de datos.

Separación de funciones (segregación): Uno de los principios del control interno es la segregación de funciones es para prevenir el fraude interno en la organización. Con esto un individuo llevará a cabo todas las actividades de operación, no todo estará bajo su responsabilidad; ninguna persona debe ser capaz de registrar, autorizar y conciliar una transacción. Ello como mecanismo de protección para esas mismas personas y de la misma organización.

Codificación

Aseguramiento del Ambiente de desarrollo: Las personas que realizan las actividades deben comprometerse con el trabajo y con la calidad. Resalta la importancia de la formación de los miembros del equipo como instrumento para sensibilizar y seguir un proceso de desarrollo con criterios para producir productos de calidad.

Codificación Segura:

- **Código neutral respecto a la seguridad:** Respecto a la seguridad el código neutral no ejecuta ninguna acción concreta con el sistema de seguridad. El código neutral se ejecuta con cualquier permiso que sea concebido. Esto con el fin de que puedan detectar excepciones de seguridad asociadas con operaciones protegidas (como la utilización de archivos, trabajo en red, etc.) ejecute una excepción no controlada.

- **Código de la aplicación que no sea un componente reutilizable:** Vemos que la seguridad es muy sencilla en una aplicación no llama a otro código. Esto es diferente cuando una parte de código malicioso puede llamar al código de nuestra aplicación. Muchas veces el desarrollador tiene controlada esta parte de acceso, para evitar que código malicioso use los recursos del sistema, pero esta seguridad puede ser burlada dejando al descubierto valores de campos o propiedades que contengan información confidencial.

Validación

Inspección de software: Muchos desarrolladores creen que la calidad del software se debe de inspeccionar al terminar debe codificarse, esto es un gran error, ya que la calidad del software es una actividad de protección que debemos garantizar a lo largo de todo el proceso de ingeniería de software. Por esto el llevar el control de calidad del software implica una serie de revisiones, y pruebas que se hacen en

el trayecto del ciclo de desarrollo, con esto se asegura que el software cumple con los requisitos asignados.

Pruebas de caja blanca: Las pruebas de caja blanca son destinadas para comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones. Estas pruebas también son llamadas "Pruebas estructurales".

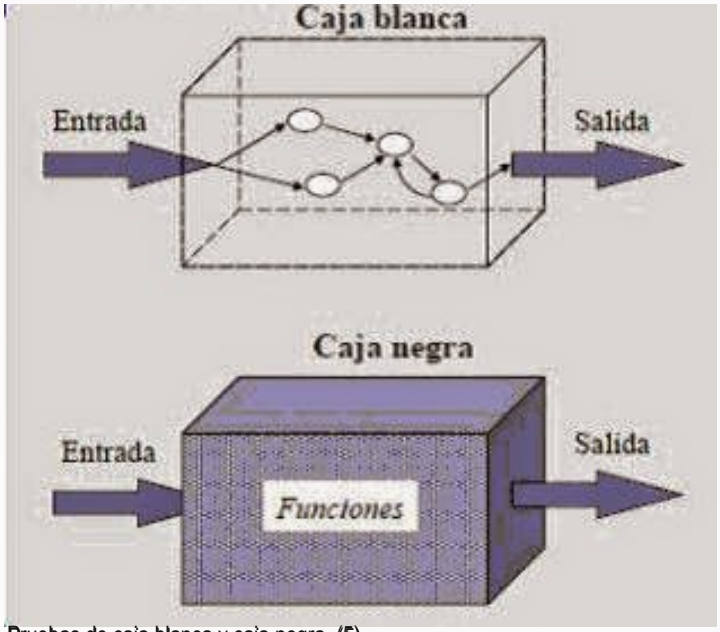
Pruebas de caja negra: Las pruebas de caja negra son diseñadas para que se puedan aplicar sobre el sistema sin necesidad de conocer como está construido por dentro. Al sistema se ingresan un conjunto de datos y se analizan los datos de salida para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Estas pruebas también son llamadas "Pruebas funcionales".

Conclusiones

- Como pudimos observar, la seguridad en todo el ciclo de desarrollo del software no es un producto, ya que esto es una sumatoria de personas, procesos y tecnologías. También podemos concluir que es muy costoso aplicar la seguridad al final del desarrollo del software y no durante el proceso, ya que esto implica reingeniería en el software finalizado.

- Así mismo debemos de estar pendientes de muchos detalles en el proceso de desarrollo de software, para evitar fallos de vulnerabilidad que puedan comprometer el sistema, dando a personas mal intencionadas acceso a los datos de los clientes.

- No podemos asegurar el sistema en un 100%, pero podemos evitar fallos desde el inicio del proyecto haciendo este menos vulnerable a intrusiones que puedan afectar el sistema.



Pruebas de caja blanca y caja negra. (5)

Referencias

1. Carlos Allendes (08/04/2016), Desarrollo Seguro https://www.owasp.org/images/1/17/DesarrolloSeguro_RD.pdf, (20/03/2017)

2. Cesar R. Cuenca Díaz (OWASP) (6/7/2016) Desarrollo Seguro: Principios y Buenas Prácticas, https://www.owasp.org/images/9/93/Desarrollo_Seguro_Principios_y_Buenas_Prácticas.pdf, (20/3/2017)

3. Denise Guisto Bilic (12/03/2015), 10 consejos para el desarrollo seguro de aplicaciones, http://www.welivesecurity.com/la-es/2015/03/12/10-consejos-desarrollo-seguro-de-aplicaciones/, (20/3/2017).

4. José María Luna (3/7/2009). Pruebas de Caja Negra y Caja Blanca, http://ingenierogestion.blogspot.com/2009/06/pruebas-de-caja-negra-y-caja-blanca.html. (20/3/2017).

5. Martinez Burgos Arturo.Pruebas de caja blanca y caja negra. http://martinezburgosarturoivan.blogspot.com/2015/01/2do-parcial-2da-tarea-pruebas-de-caja.html

6. Mary Tenelema Ciclo de vida de Software. http://es.calameo.com/read/003285581c078a5847539

7. Microsoft (01/11/2007). Información general sobre codificación segura, https://msdn.microsoft.com/es-es/library/8a3x2b7f(v=vs.90).aspx, (20/3/2017).

8. SoftwareSeguridad (01/02/2016). Seguridad en el desarrollo de Software, http://www.softwareseguridad.com/seguridadeneldesarrollodesoftware.html, (20/3/2017).