Puntos Clave para el Desarrollo de Software Seguro

Soto Camacho Harvey
Universidad Piloto de Colombia
Bogotá, Colombia
harsoto87@gmail.com

Resumen - La ausencia de prácticas de seguridad al interior y exterior de las organizaciones, afecta tanto los procesos como la imagen del negocio, además de las innumerables pérdidas monetarias que esto puede llegar a abarcar, a tal punto que puede llevar a la quiebra de una organización.
Por medio de este documento se pretende dar luz sobre algunos puntos relevantes para que las casas de software apliquen ciertos parámetros y buenas prácticas en sus procesos, mostrando a grandes rasgos lo que sería un ciclo de vida para el desarrollo de software seguro.

Summary - The absence of safety practices inside and outside of organizations, both processes affect the image of the business in addition to the countless monetary losses that may eventually include this to the point that may lead to bankruptcy of an organization.

Through this paper aims to shed light on some important points for software houses and certain parameters apply good practices in their processes, showing roughly what would be a life cycle for the development of secure software.

I. INTRODUCCIÓN

En nuestra actualidad, cada vez más las empresas apoyan sus procesos con el uso de la tecnología, esto implica de una manera directa o indirecta que las empresas cuenten aplicaciones o componentes de software dentro de sus inventarios, y adicional a esto que la empresa tenga nuevos retos para que su negocio sea continuo y seguro. Sin embargo, a veces las empresas que deciden adquirir software, se preocupan más por la funcionalidad y no se preguntan si su uso le ofrece al negocio la confiabilidad que requiere, es decir, que el software sea de calidad.

En un artículo publicado el mes de septiembre del año pasado en la revista sistemas, anotan lo siguiente: “En el país existen 1.850 empresas de tecnología, de las cuales 850 se dedican al desarrollo de software.”[1], y si contamos a las personas independientes que se dedican al desarrollo de software (estando o no trabajando en una de estas empresas), notamos que Colombia está creciendo en el campo del desarrollo de software; pero, ¿estos desarrollos son construidos con estándares de calidad?

El mismo artículo citado, también especifica que en la federación colombiana de la industria de software y TI (Fedesoft), “se encuentran agrupadas

380 de esas compañías que cumplen con estándares de desarrollo" [1].

De acuerdo con estas estadísticas, se podría establecer que existe una probabilidad considerable de que un producto software finalizado pueda tener un grado considerable de inseguridad o de brechas abiertas, al no cumplir con los estándares de desarrollo.
Esto se ve más claro con la siguiente afirmación que hace Andy Chou cofundador de Coverity: "El problema de la seguridad cibernética es que tiene sus raíces en el deficiente y mal diseño de un código de software" [2].

II. CICLO DE DESARROLLO DE SOFTWARE

Existen varias metodologías de desarrollo de software que pueden ser tomadas como punto de referencia para iniciar la creación de un producto. El ciclo más básico para la creación de software se compone de 5 etapas principales:

- Etapa de Análisis.
- Etapa de Diseño.
- Etapa de Implementación.
- Etapa de Pruebas.
- Etapa de Mantenimiento.

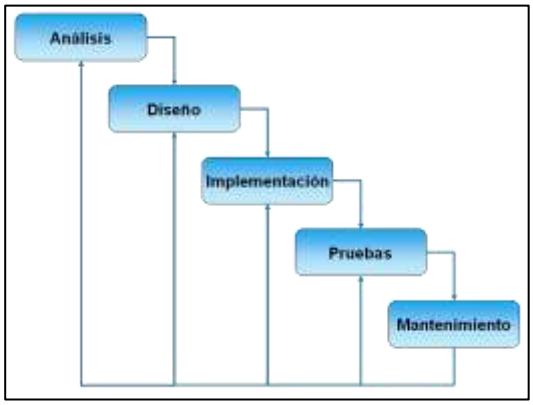

Fig. 1 Ciclo de vida básico de desarrollo de software. [3]

Cada etapa de este ciclo cumple con unas tareas específicas en un tiempo determinado, y no se inicia con una fase si la fase anterior no ha sido completada. Durante este ciclo de desarrollo, la etapa de pruebas es la encargada de certificar la calidad del software, ya que, los testers o el personal de pruebas, son los encargados de encontrar aquellas falencias que hacen que un producto no funcione como debería y "Un sólo bug puede provocar un enorme desastre." [4].

La mala práctica de dejar encargado todo el trabajo de seguridad en sólo una etapa del desarrollo de software debe empezar a cambiarse, la calidad debe ser tema que se incluya en cada momento de la construcción, y con ello el tema de la seguridad, tal y como lo menciona el cofundador de Coverity Andy Chou: "La idea de entregar el código no probado al equipo de Control de Calidad (QA) está empezando a parecer anticuado y poco profesional."[5].

Por otro lado hay que "reconocer que el software perfectamente seguro es imposible de lograr" [5], debemos empezar por anotar que el software es desarrollado por humanos y por esto mismo tiende a tener errores, ya sean graves que afecten el correcto funcionamiento del software, o leves que no afectan la continuidad del negocio, pero al fin y al acabo errores.

III. VISIÓN DE DESARROLLO DE SOFTWARE SEGURO

Las organizaciones necesitan de un software seguro para minimizar el riesgo de que existan brechas que afecten la seguridad de la información, por esto, el término de seguridad o aplicaciones seguras debe tomarse en cuenta desde las fábricas de software, las cuales, deben aplicar las mejores prácticas en cada etapa del desarrollo de software.

Colombia está creciendo en cuanto a desarrollo de software, y que mejor que este país sea reconocido por hacer un buen trabajo. Por ejemplo la casa de software IG WebServices S.A.S. (Intergrupo), tiene presencia en 7 países y es de las pocas empresas colombianas en estar certificada en un nivel relevante de madurez (Nivel 5) en el modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (CMMI) [6], y aunque es una muy buena referencia tener este tipo de conocimiento dentro de una organización, no es suficiente para la creación de software seguro. Hay que estar en un mejoramiento continuo en los procesos y alcanzar cierto estándar, utilizando las mejores prácticas y actividades complementarias para el desarrollo de software seguro.

Actualmente la principal prioridad que tienen los desarrolladores es que el código que estén llevando a cabo funcione, y casi siempre omiten la realización de algunas de las mejores prácticas para la realización de un software seguro, tales como:

- Validar los datos de entrada antes de procesarlos.
- Validar los datos en todas las capas que tenga el proyecto software.
- Controlar el tamaño y el tipo de datos.
- “Sanear” los valores de entrada y salida.
- Eliminar o “escapear” caracteres especiales.
- Evitar generar código con valores ingresados por el usuario.
- No mezclar datos con código.
- Capturar errores de capas inferiores y no mostrarlos al usuario. [7]

Es importante tener en cuenta estas buenas prácticas en el momento del desarrollo, pero así sean aplicadas dentro del ciclo de desarrollo de software básico (Ver Fig. 1), no son suficientes para crear un producto seguro, por este motivo ahora la seguridad debería aplicarse en cada momento del ciclo de desarrollo, implementando ciertas mejores prácticas adicionales (ver Fig.2), las cuales pueden ser iteradas varias veces durante la creación del software.

Algunas de estas tareas claves que ayudan a obtener buenos resultados al final del ciclo de vida son las siguientes [8]:

Casos de abuso: Los casos de abuso consideran las acciones para dar un mal uso deliberado de la aplicación, y determinar como la aplicación responde a estos.

Requerimientos de seguridad: Son los procesos y controles que se deberían aplicar para prevenir (o tratar de reducir) la posibilidad de que los casos de abusos se presenten.

Análisis de riesgos: Se evalúa el impacto que tendría un ataque exitoso descrito en los casos de

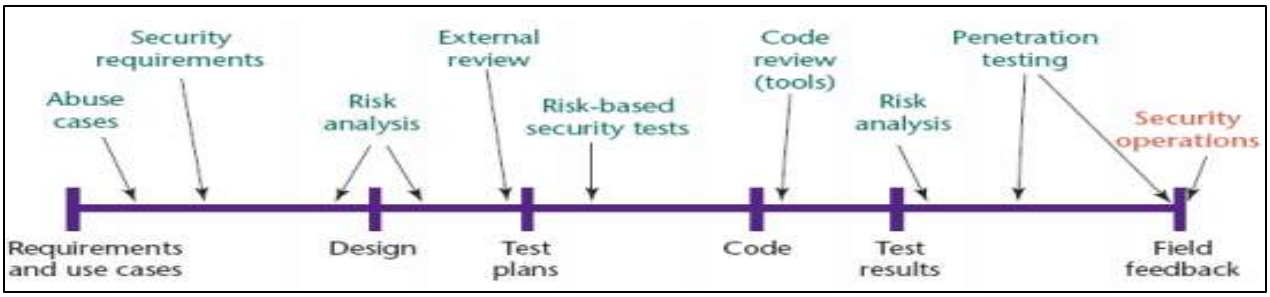

Fig. 2 Mejores prácticas en el ciclo de desarrollo de software bajo un ciclo de desarrollo básico [8].

abuso. Es una de las actividades más importantes a realizar, ya que, se aplica tanto para analizar riesgos de negocio como para analizar riesgos inmersos en la arquitectura.

Planificación de pruebas: La planificación de pruebas es una de las actividades más importantes para darle forma a un software seguro, ya que, aquí se consideran las pruebas funcionales, las pruebas de seguridad que se realizan al código realizado, en torno a los riesgos existentes en la arquitectura.

Revisión de código: La revisión de código se lleva a cabo mediante herramientas y se centra en la búsqueda de errores que se introdujeron durante la codificación, por ejemplo, desbordamientos de búfer.

Pruebas de penetración: Las pruebas de penetración son la mejor alternativa para mejorar la seguridad del software, tanto en funcionalidad como en su arquitectura, debido a que se encargan de ver las vulnerabilidades que están presentes actualmente en el entrono y a su vez se muestra como puede ser explotada.

Despliegue y operaciones: Esto se refiere a la configuración que se le realice a la aplicación con seguridad. Un control de seguridad que ha sido implementado con el objetivo de mitigar un riesgo descubierto, puede que no funcione si la aplicación no se configura correctamente, haciéndolo inútil y riesgoso.

A. Roles y responsabilidades.

Delegar funciones y responsabilidades es sinónimo de organización, esto ayuda a ser más eficientes a la hora de realizar actividades y obtener buenos resultados. Durante el ciclo de desarrollo de software seguro, llevar a cabo un ciclo de vida ordenado, es base para cumplir con los objetivos que se plantean.

Para llegar a ese punto, hay que conocer de ante mano las capacidades de cada uno de los integrantes de un proyecto determinado, para poder asignar los recursos de la mejor manera y brindar los privilegios adecuados para acceso a instalaciones, programas, bases de datos, servidores y todo artefacto involucrado en el desarrollo del proyecto.

Para esto se agrupan en ciertos roles a los participantes en la creación del proyecto, lo cual ayuda a tener una segregación de responsabilidades fáciles de administrar y mantener un desarrollo ordenado.

Por nombrar algunos de los roles que pueden estar involucrados en el ciclo de desarrollo de software son los siguientes:

- Authorizing Official (AO).
- Chief Information Officer (CIO).

- Configuration Management Manager (CM).
- Contracting Officer.
- Information Technology Investment Board.
- Legal Advisor/Contract Attorney.
- Privacy Officer.
- Program Manager / Official (Information Owner).
- QA/Test Director.
- System Architect.
- System Owner.

Nota: Los roles se especifican en inglés de acuerdo con el texto original de donde fueron consultados [9] para no desvariar el significado al ser traducido.

IV. CONCLUSIONES

Ya existen las herramientas y las acciones a tomar en cuenta dentro del ciclo de vida del desarrollo de software para que se pueda alcanzar un grado óptimo de seguridad en las aplicaciones software creadas, hay que tomar la buena decisión de aplicarlas en las organizaciones o en el desarrollo que realicen las personas independientes.

Como práctica adicional, es importante generar un ambiente de colaboración para que la seguridad sea un compromiso de todas las personas interesadas, empezando por el apoyo de la alta gerencia para realizar estas actividades, ya que, sin importar el objetivo del proyecto software que se esté realizando, se debe garantizar que su uso en ambiente productivo cumpla con los pilares de la seguridad informática (Confidencialidad, Integridad, Disponibilidad).

REFERENCIAS

[1] Paola Restrepo. (2013) UNO. DESARROLLO DEL SOFTWARE EN COLOMBIA. [Online]. Available: http://acis.org.co/revistasistemas/index.php/component/k2/item/138-desarrollo-del-software-en-colombia

[2] Andy Chou. (2013) Cyber security: the magic box myth. [Online]. Available: http://www.scmagazineuk.com/cyber-security-the-magic-box-myth/article/304127/

[3] Misty Soloriio. (2013) METODOLOGÍA EN CASCADA. [Online]. Available: http://metodologiaencascada.blogspot.com/

[4] Jaideep Khanduja. (2013) A Single Bug Can Cause A Huge Disaster: NASA Hired Testing Services From Coverity. [Online]. Available: http://itknowledgeexchange.techtarget.com/quality-assurance/a-single-bug-can-cause-a-huge-disaster-nasa-hired-testing-services-from-coverity/

[5] Andy Chou. (2013) Developers need to start thinking about security now. [Online]. Available: http://venturebeat.com/2013/11/08/developer-first-security-2/

[6] Patty Pedroza Barrios. (2013) Elección de una Metodología de Desarrollo a partir de las Ventajas de una Metodología Ágil y un Modelo Robusto como CMMI-DEV 1.3. [Online]. Available: https://www.google.com.co/url?sa=t&rct=j&q=&esrc=s&source=web&cd=19&ved=0CF4QFjAIOAo&url=http%3A%2F%2Fwww.unilibrebaq.edu.co%2Funilibrebaq%2Frevistas2%2Findex.php%2Fingeniare%2Farticle%2Fdownload%2F358%2F325&ei=ZXrJU-bKJe3LsASFz4KACg&usg=AFQjCNEtcFESJ6eQ286pE_IWaMQtxoHlwg&sig2=c8Rr2cPxjXlnCmmvevXAHA&bvm=bv.71198958,d.cWc&cad=rja

[7] Pablo Milano. (2007) Seguridad en el ciclo de vida del desarrollo de software. [Online]. Available: http://www.cybsec.com.ar/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

[8] Kenny Paterson. Kenny Paterson. [Online]. Available: https://www.owasp.org/images/3/35/Software_Development_And_Information_Security-Tom_Neaves.pdf

[9] NIST. (2008) Security Considerations in the System Development Life Cycle. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-64-Rev2/SP800-64-Revision2.pdf