



Seguridad en el ciclo de vida del desarrollo de software

**POR PABLO MILANO,
CONSULTOR CYBSEC**

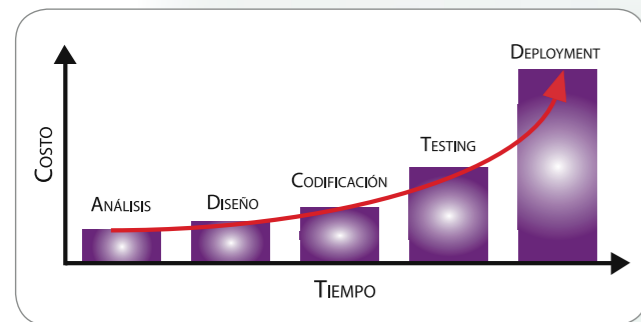


La mayor parte de las organizaciones desarrolla o contrata el desarrollo de aplicaciones propias para su gestión de negocio. Como todo software, estas aplicaciones pueden contener fallas de seguridad y a diferencia del software comercial, no se dispone de actualizaciones o parches liberados en forma periódica por el fabricante. El tratamiento de las vulnerabilidades en aplicaciones propias corre por parte de la organización que las desarrolla.

¿Qué está pasando en las organizaciones?

Lamentablemente es una práctica habitual en muchas organizaciones la “puesta en producción” de sistemas sin la participación del sector de Seguridad de la Información. Muchas otras veces, el sector de Seguridad se entera demasiado tarde, y no tiene suficiente margen de acción para el análisis de seguridad de la aplicación desarrollada.

Por lo general, en el mejor de los casos, se coordina un testeo de seguridad una vez que la aplicación ya está desarrollada. Aquí muchas veces se encuentran errores que requieren el rediseño de parte de la aplicación, lo cual implica un costo adicional en tiempo y esfuerzo.



Está comprobado que cuánto más temprano se encuentre una falla de seguridad en el ciclo de vida del desarrollo de software, más rápida y económica será su mitigación. ¿Cuál es el rumbo a seguir? Las buenas prácticas indican la conveniencia de incluir seguridad de la información desde el principio y a lo largo de todas las etapas del ciclo de vida de desarrollo, conocido como SDLC por ser las siglas en inglés de Software Development Life Cycle.

Estas etapas pueden variar según la modalidad de cada organización, pero a grandes rasgos son las siguientes: análisis de requerimientos, diseño funcional y detallado, codificación, testing/QA, implementación/puesta en producción.

Como hemos mencionado antes, es habitual incluir seguridad únicamente al momento de realizar el Testing. De manera análoga a un arquero de fútbol, al que le llegan continuos pelotazos por fallas en la defensa y debilidades en el ataque, el equipo de Testing/QA puede hacer su mejor esfuerzo para tratar de detectar la mayor cantidad de fallas de seguridad, pero deberían aplicarse medidas

previas para que estas fallas sean las menos posibles y no tener que dejar ‘toda la responsabilidad en el arquero’.

¿Cómo implementar seguridad a lo largo del SDLC? Es importante que el personal de seguridad de la información participe en las distintas etapas de desarrollo, supervisando la realización de las mismas.

Seguridad en el análisis de requerimientos

En esta etapa, se deben identificar aquellos requerimientos funcionales que tendrán impacto en los aspectos de seguridad de la aplicación. Algunos de ellos son: requerimientos de compliance con normativas locales o internacionales (ej: PCI, SOX, “A” 4609, etc.), tipo de información que se transmitirá o procesará (ej: Información pública o confidencial, datos personales, datos financieros, contraseñas, datos de pago electrónico, etc.) y requerimientos de registros de auditoría (ej: Qué debe registrar la aplicación en sus Logs).

Seguridad en el diseño

Antes de comenzar a escribir líneas de código, hay numerosos aspectos de seguridad que deben ser tenidos en cuenta durante el diseño de la aplicación. Algunos de ellos son: diseño de autorización (ej: Definir los roles, permisos y privilegios de la aplicación), diseño de autenticación (aquí se debe diseñar el modo en el que los usuarios se van a autenticar, contemplando aspectos tales como los mecanismos o factores de autenticación con contraseñas, tokens, certificados, etc. posibilidades de integrar la autenticación con servicios externos como LDAP, Radius o Active Directory y los mecanismos que tendrá la aplicación para evitar ataques de diccionario o de fuerza bruta (ej: bloqueo de cuentas, implementación de “captchas”, etc.), diseño de los mensajes de error y advertencia, para evitar que los mismos brinden demasiada información y que ésta sea utilizada por atacantes y diseño de los mecanismos de protección de datos (aquí se debe contemplar el modo en el que se protegerá la información sensible en tránsito o almacenada; según el caso, se puede definir la implementación de encriptación, hashes o truncamiento de la información).

Una vez que se cuenta con el diseño detallado de la aplicación, una práctica interesante es la de realizar sobre el mismo un análisis de riesgo orientado a software. Existen técnicas documentadas al respecto, tales como Threat Modeling. Estas técnicas permiten definir un marco para identificar debilidades de seguridad en el software, antes de la etapa de codificación. Como valor agregado, del análisis de riesgo orientado a software se pueden obtener casos de prueba para ser utilizados en la etapa de Testing/QA.

Seguridad en la codificación

Una vez concluido el diseño, le toca a los desarrolladores el turno de codificar los distintos componentes de la aplicación. Es en este punto en donde suelen incorporarse, por error u omisión, distintos

tipos de vulnerabilidades. Estas vulnerabilidades podríamos dividir las en dos grandes grupos a saber: vulnerabilidades clásicas y vulnerabilidades funcionales. Las primeras son bien conocidas y categorizadas. Ejemplo de estas vulnerabilidades son las presentes en el “OWASP Top 10” (Vulnerabilidades de inyección, Cross Site Scripting, errores en manejo de sesiones, etc.) como así también otras vulnerabilidades no ligadas directamente con las aplicaciones WEB, como desbordamiento de buffer, denegación de servicio, etc. Los ‘Frameworks’ de desarrollo de aplicaciones son una buena ayuda en este punto, ya que ofician de intermediario entre el programador y el código, y permiten prevenir la mayoría de las vulnerabilidades conocidas. Ejemplos de estos frameworks son *Struts*, *Ruby on Rails* y *Zope*.

Vulnerabilidades funcionales son aquellas ligadas específicamente a la funcionalidad de negocio que posee la aplicación, por lo que no están previamente categorizadas. Algunos ejemplos ilustrativos de este tipo de vulnerabilidad son los siguientes: una aplicación de banca electrónica que permite realizar transferencias con valores negativos, un sistema de subastas que permite ver los valores de otros oferentes, un sistema de venta de entradas para espectáculos que no impone límites adecuados a la cantidad de reservas que un usuario puede hacer.

En la etapa de codificación, una de las reglas de oro es ‘verificar todos los valores de entrada y de salida’. Esto es, asumir siempre que el valor pudo haber sido manipulado o ingresado maliciosamente antes de ser procesado.

Testing / QA de seguridad

Tradicionalmente, la labor del equipo de Testing/QA fue la de encontrar y reportar errores funcionales de la aplicación. Para esto, se desarrollan ‘casos de test’ basados en la funcionalidad esperada. A esto denominamos ‘testing funcional’ y básicamente consiste en validar que la aplicación ‘haga lo que se esperaba que hiciera’. Sucede que habitualmente hay un desfase entre el diseño original



de la aplicación (lo que se espera que haga) y la implementación real (lo que realmente hace). Aquí surgen 3 áreas bien definidas: lo que fue definido y la aplicación hace, lo que fue definido y la aplicación no hace (errores funcionales) y lo que no fue definido pero la aplicación hace.

Es en este último grupo, en donde habitualmente están las vulnerabilidades, y el testing funcional clásico no es capaz de encontrarlas. Por este motivo se necesitan nuevas técnicas para explorar lo desconocido. El testing de seguridad se basa principalmente en probar la aplicación con escenarios no planificados, incluyendo valores mutados, fuera de rango, de tipo incorrecto o malformados, acciones fuera de orden, etc.

Existen herramientas automáticas que pueden ayudar al analista de QA en la búsqueda de errores. Las mismas son útiles por su velocidad y capacidad de automatización, pero pueden causar falsos positivos, y por lo general no son buenas detectando vulnerabilidades funcionales. Es por esto que los mejores resultados resultan de la combinación de técnicas automáticas y manuales.

Implementación / Puesta en producción

Una mala configuración al momento de implementar la aplicación podría echar por tierra toda la seguridad de las capas anteriores. Tanto la aplicación como el software de base deben configurarse de manera segura al momento de poner el software en producción. En este punto se deben contemplar tareas tales como: cambio de usuarios y contraseñas iniciales o por defecto, borrado de datos de prueba y cambio de permisos de acceso. Es también importante mantener una correcta separación de los ambientes de desarrollo, testing y producción y procedimientos de traspaso seguro de uno a otro de estos ambientes.

Conclusión

La seguridad en las aplicaciones de software debe abordarse desde el primer día del proceso de desarrollo y a lo largo de todas las etapas del mismo. En cada una de estas etapas, se pueden realizar diversas actividades que en su conjunto ayudarán a aumentar la seguridad de la aplicación de software que se está desarrollando. Es importante que en cada organización, el sector de seguridad de la información sea invitado a participar a lo largo de todo el proceso de desarrollo como supervisor de las tareas y verificaciones de seguridad. La integración de seguridad a lo largo del SDLC ayuda a reducir las fallas de seguridad como así también los costos de la aplicación, tanto tangibles (tiempo / dinero) como intangibles (imagen de la organización)