Capítulo 1. Introducción al desarrollo seguro

Según estudios realizados, cerca del 90 % de los incidentes de seguridad del software están causados por atacantes que han explotado fallos conocidos en él. Además, en un análisis de 45 aplicaciones de negocio electrónico, se mostró que el 70 % de los fallos del software estaban relacionados con el diseño. De media, un ingeniero de software experimentado y capacitado introduce un fallo por cada nueve líneas de código. Aproximadamente, un millón de líneas de código podrían tener de mil a cinco mil fallos, los cuales podrían ser el origen de alrededor de cien vulnerabilidades en producción. Además, se ha demostrado que el software se degrada con el tiempo, y lo que vale en un momento en concreto puede no valer al cabo de seis meses.

En este sentido, la ingeniería de software de desarrollo seguro es una disciplina de la ingeniería del software en la que se busca proporcionar garantías objetivas respecto a la seguridad del software desarrollado. Para ello, se aplican mecanismos y se producen evidencias durante el proceso que garantizan que el software contiene las propiedades de seguridad que se le requieran.

En particular, y partiendo del axioma “la seguridad absoluta no es alcanzable”, el software seguro es capaz de minimizar el impacto de la mayoría de los ataques, tolerar aquellos que no pueda resistir, y recuperarse rápidamente y con el menor impacto de aquellos otros que no pueda tolerar.

El campo de la investigación en ingeniería de software seguro se centra en desarrollar métodos eficaces y herramientas que permitan al desarrollador, durante el ciclo de vida del desarrollo de software, la implementación segura desde las primeras fases de análisis hasta las etapas de más largo plazo de desarrollo.

El desarrollo seguro es una parte importante de la seguridad informática, englobado dentro del ámbito de la prevención. Podríamos definir que un programa seguro es aquel capaz de seguir realizando las funciones para las que ha sido creado en todo momento, y capaz de evitar que la existencia de errores en él pueda ser utilizada como puente para la realización de acciones que pongan en peligro la integridad, confidencialidad o disponibilidad del resto de elementos del sistema en el que se está ejecutando.

Por tanto, la programación segura engloba el conjunto de técnicas, normas y conocimientos que permiten crear programas que no puedan ser modificados de forma ilegítima con fines maliciosos y que estén carentes de fallos que puedan comprometer la seguridad del resto de elementos del sistema con el que interactúan.

Uno de los principales objetivos radica en ayudar a desarrolladores de software a implementar medidas preventivas que ayuden a mitigar o reducir lo máximo posibles errores comunes de programación y que pueden ser aprovechados por un posible atacante para comprometer una aplicación.

1.1 Propiedades del software seguro

Cuando una aplicación transmite o almacena información confidencial, la aplicación es responsable de garantizar que los datos almacenados y transferidos estén cifrados y no puedan obtenerse, alterarse o divulgarse fácilmente de forma ilícita. En general, se suele decir que el objetivo fundamental de la seguridad informática se basa en preservar los siguientes puntos:

- **Confidencialidad**. El software debe asegurar que cualquiera de sus características, los activos que administra y su contenido son accesibles solo para las entidades autorizadas e inaccesibles para el resto. El acceso a la información está limitado a usuarios autorizados. Los datos deben protegerse de la observación o divulgación no autorizadas tanto en tránsito como almacenadas.
- **Integridad**. El software y los activos del sistema solo pueden ser modificados por usuarios autorizados. Esta propiedad se debe preservar durante el desarrollo del software y su ejecución. Los datos han de protegerse al ser creados, alterados o borrados maliciosamente por atacantes no autorizados.
- **Disponibilidad**. El software debe estar operativo y ser accesible a sus usuarios autorizados siempre que se requiera, así como ha de desempeñarse con una performance adecuada para que los usuarios puedan realizar sus tareas de forma correcta y dar cumplimiento a los objetivos de la organización que lo utiliza. El acceso a los activos en un tiempo razonable está garantizado para usuarios autorizados. Los datos deben encontrarse disponibles para los usuarios autorizados, según sea necesario.

Para las entidades que actúan como usuarios, se requieren dos propiedades adicionales:

- **Trazabilidad**. Todas las acciones relevantes relacionadas con la seguridad de una entidad que actúa como usuario se deben registrar y trazar con el objetivo de disponer de datos para la realización de auditorías; de esta forma, la trazabilidad debe ser posible tanto durante la ocurrencia de las acciones registradas como a posteriori.
- **No repudio**. Constituye la habilidad de prevenir que una entidad que actúa como usuario desmienta o niegue la responsabilidad de acciones que han sido ejecutadas.

Estas propiedades básicas son las más utilizadas normalmente para describir la seguridad de los sistemas y aplicaciones. Un ataque con éxito de inyección de SQL en una aplicación para extraer información de identificación personal de su base de datos sería una violación de la propiedad de confidencialidad. El éxito de un ataque cross-site scripting (XSS) en contra de una aplicación web podría dar lugar a una violación tanto en su integridad como en su disponibilidad . Un ataque con éxito de desbordamiento de búfer que inyecta código con el objetivo de obtener y modificar la información de usuarios sería una violación de las cinco propiedades básicas de seguridad.

1.2 Principios de diseño seguro de aplicaciones

A continuación, se sugieren una serie de principios orientados al diseño seguro de aplicaciones:

1.2.1 Minimizar el área de la superficie de ataque

Cada característica que se añade a una aplicación incrementa su complejidad y aumenta también el riesgo de aplicación en conjunto. Una nueva característica implica un nuevo punto de ataque. Uno de los factores clave para reducir el riesgo de una aplicación recae en la reducción de la superficie de ataque. Pueden eliminarse posibles puntos de ataque si se deshabilitan módulos o componentes innecesarios para la aplicación; por ejemplo, si la aplicación no utiliza el almacenamiento en caché de resultados, sería recomendable deshabilitar dicho módulo. De esta manera, si se detecta una vulnerabilidad de seguridad en ese módulo, la aplicación no se verá amenazada.

1.2.2 Seguridad por defecto

Existen muchas maneras de entregar una experiencia out of the box ("lista para usar") a los usuarios. Sin embargo, por defecto, la experiencia debe ser segura, más facilitar, no obstante, la capacidad de reducción del nivel de seguridad si el usuario lo cree necesario. Del lado de los desarrolladores, constituye una práctica habitual utilizar opciones de configuración de seguridad reducidas para evitar que dichas configuraciones compliquen el desarrollo. Si, para su implementación, la aplicación requiere de características que obligan a reducir o cambiar la configuración de seguridad predeterminada, se recomienda estudiar sus efectos y consecuencias, para lo que hay que realizar pruebas específicas de auditoría de seguridad.

1.2.3 Privilegios mínimos

Según el principio del mínimo privilegio, se recomienda que las cuentas de usuario tengan la mínima cantidad de privilegios necesarios. Asimismo, se aconseja que los procesos se ejecuten únicamente con los privilegios necesarios para completar sus tareas. De esta manera, se limitan los posibles daños que podrían producirse si se ve comprometido el proceso. Si un atacante llegase a tomar el control de un proceso en un servidor o comprometiese una cuenta de usuario, los privilegios concedidos determinarán, en gran medida, los tipos de operaciones que podrá llegar a realizar dicho atacante; por ejemplo, si cierto servidor solo requiere acceso de lectura a la tabla de una base de datos, bajo ninguna condición deberían darse privilegios administrativos a los usuarios si no resultan necesarios.

1.2.4 Validación de datos de entrada

La debilidad de seguridad más común en aplicaciones es la falta de validación apropiada de las entradas del usuario o del entorno. Esta debilidad origina casi todas las principales vulnerabilidades en las aplicaciones, tales como inyecciones de código, inserción de

secuencias de comandos, ataques al sistema de archivos o desbordamientos de memoria. Las aplicaciones deben validar todos los datos introducidos por el usuario antes de realizar cualquier operación con ellos. La validación podría incluir el filtrado de caracteres especiales o el control de la longitud de los datos introducidos. Esta medida preventiva protege a la aplicación de usos incorrectos accidentales o de ataques deliberados por parte de usuarios.

1.2.5 Defensa en profundidad

Con "defensa en profundidad", nos referimos a definir una estrategia de seguridad estándar en la que se establezcan varios controles de defensa en cada una de las capas y los subsistemas de la aplicación. Estos puntos de control ayudan a garantizar que solo los usuarios autenticados y autorizados puedan obtener el acceso a la siguiente capa y a sus datos.

1.2.6 Control seguro de errores

Es necesario controlar las respuestas cuando se produce algún error y no mostrar en ellas información que pudiera ayudar al atacante a descubrir datos acerca de cómo ha sido desarrollada la aplicación o cómo funcionan sus procedimientos. La información detallada de los errores producidos no debería ser mostrada al usuario, sino que tendría que ser enviada al fichero de log correspondiente. Una simple página de error 403 indicando un acceso denegado puede decir a un escáner de vulnerabilidades que un directorio existe y puede proporcionar a una atacante información que le permita realizar un mapa de la estructura de directorios de la aplicación.

1.2.7 Separación de funciones

Un punto importante consiste en la separación de funciones entre los distintos perfiles de la aplicación; por ejemplo, en una tienda de subastas online, un usuario vendedor pone en subasta un producto. Si la separación de funciones se encuentra correctamente implementada, este mismo usuario no debe poder participar en la puja por el artículo. Por otra parte, determinados roles deben contar con un nivel de confianza más elevado que los usuarios normales, como en el caso del administrador, que posee privilegios avanzados, como administrar al resto de usuarios del sistema o configurar políticas de contraseñas, así como definir los diferentes parámetros de la aplicación.

1.2.8 Evitar la seguridad por oscuridad

La seguridad basada en la oscuridad es un control de seguridad débil, especialmente si se trata del único control. Esto no significa que mantener secretos constituya una mala idea; significa que la seguridad de los sistemas clave no debería basarse, exclusivamente, en mantener detalles ocultos. Por ejemplo, la seguridad de una aplicación no debería basarse en mantener en secreto el conocimiento del código fuente. La seguridad ha de fundamentarse en muchos otros factores, incluyendo políticas de contraseñas, diseño de una arquitectura sólida tanto a nivel de aplicación como a nivel de comunicaciones de red

y realización periódica de controles de auditoría. Un ejemplo práctico es Linux, cuyo código fuente es open source y está disponible para todo el mundo y, aun así, se trata de un sistema operativo seguro y robusto.

1.3 Análisis de requisitos de seguridad

La seguridad debería ser un aspecto de alta prioridad cuando estamos desarrollando y probando aplicaciones que manejan datos confidenciales a nivel de autenticación y autorización. El análisis de los requisitos de seguridad debe ser parte de la primera fase de análisis de los requisitos de cada proyecto de aplicaciones móviles. La siguiente lista puede ayudar con el análisis de los requisitos de seguridad:

- Identificar los posibles roles de usuario, así como sus limitaciones y permisos dentro de la arquitectura (app y backend).
- Identificar qué tipo de enfoques y herramientas de pruebas de seguridad se requieren para lograr un buen nivel de seguridad.
- Identificar el impacto que tienen las funcionalidades a nivel de usuario en la seguridad a nivel frontend y backend

La mayoría de los fallos de seguridad de aplicaciones se pueden prevenir mediante la integración de los procesos y buenas prácticas de seguridad desde las primeras etapas de desarrollo. La planificación de una estrategia inicial de diseño de aplicaciones y mantenimiento de la seguridad en mente todo el tiempo permite reducir las posibilidades de los riesgos de seguridad que surgen durante las últimas etapas de desarrollo de aplicaciones

ORTEGA CANDEL, José Manuel (2020). DESARROLLO SEGURO EN INGENIERÍA DEL SOFTWARE - Aplicaciones seguras con Android, NodeJS, Python y C++. (1 ed. p. 22) Alfaomega, Marcombo. https://www.alfaomega.com.mx/read/9786075386195/index