AndalucíaCERT
CENTRO DE SEGURIDAD TIC

Informe de divulgación
Criterios para el desarrollo de aplicaciones seguras

Tipo de documento:	*Informe*
Autor del documento:	*AndalucíaCERT*
Código del Documento:	*CERT-IF-1588-060412*
Edición:	*0*
Categoría	*Público*
Fecha de elaboración:	*06/04/2012*
N° de Páginas	*1 de 9*

1 TABLA DE CONTENIDOS

2 OBJETO

El objeto de este documento es proporcionar a todo el personal implicado en el desarrollo y en la explotación del software, más allá de las pautas y directrices que determina MADEJA como Marco de Desarrollo de la Junta de Andalucía, una referencia clara de cuáles son los criterios que han de guiar esta actividad para conseguir un buen nivel de seguridad en la aplicación desde la fase de desarrollo. Describiremos principios generales para el diseño seguro de aplicaciones y comentaremos puntos a tener en cuenta para detectar y corregir posibles fallos de seguridad y debilidades presentes en nuestras aplicaciones.

3 ALCANCE

El documento va destinado al personal de la Junta de Andalucía y público en general.

4 INTRODUCCIÓN

La necesidad de desarrollar aplicaciones seguras y, por tanto, tener en cuenta la seguridad en las metodologías de desarrollo es tan importante como tenerla en cuenta para la construcción de edificios o barcos. Desarrollar aplicaciones sin tener en cuenta la seguridad es como construir un barco sin botes salvavidas.

Si creamos aplicaciones, debemos de tener en cuenta que somos los responsables tanto de su construcción como de que se mantenga siempre en pie ante cualquier situación posible. Un ingeniero de barcos debe pensar en que el barco que ha diseñado tiene que ser capaz de mantenerse a flote, ser veloz, cómodo de usar, pero también debe pensar en que debe soportar fuertes vientos, fuegos, accidentes, etc.

El **desarrollo seguro** es una parte importante de la seguridad informática, englobado dentro del ámbito de la **prevención**. No hay una definición exacta, pero podemos decir que un programa seguro será aquel que sea capaz de seguir realizando las funciones para las que ha sido creado inalterables en todo momento, y capaz de evitar que la existencia de errores en el mismo puedan ser utilizados como puente para la realización de actos que pongan el 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 subvertidos o alterados de forma ilegítima con fines maliciosos, y carente de fallos que puedan comprometer la seguridad del resto de elementos del sistema con el que interactúa.** Al igual que en otros ámbitos de la seguridad informática, el desarrollo seguro constituye un compromiso entre cuan seguro queremos que sea nuestro programa y cuanto esfuerzo estamos dispuestos a invertir para conseguirlo.

Si bien las metodologías y estándares relacionados con el desarrollo y construcción de software buscan mantener altos niveles de confiabilidad y control de la solución informática, la seguridad informática y

sus principios de diseño seguro **no suelen ser** parte formal u obligatoria de dichos estándares. En el caso de MADEJA, marco de desarrollo usado en la Junta de Andalucía, la seguridad se considera como algo clave.

En general se suele decir que el objetivo fundamental de la seguridad informática se basa en preservar los siguientes puntos:

- **Integridad:** Los activos del sistema sólo pueden ser borrados o modificados por usuarios autorizados.
- **Confidencialidad:** El acceso a la información está limitado a usuarios autorizados.
- **Disponibilidad:** El acceso a los activos en un tiempo razonable está garantizado para usuarios autorizados.

Los principios y recomendaciones descritos en este documento están relacionados con el mantenimiento de estos tres pilares.

5 PRINCIPIOS DE DISEÑO SEGURO DE APLICACIONES

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

5.1 Minimizar el área de la superficie de ataque

"Si no lo utiliza, deshabilítelo."

Cada característica que se añade a una aplicación incrementa la complejidad de la misma e incrementa el riesgo de aplicación en conjunto. Una nueva característica implica un nuevo punto de ataque.

Uno de los factores claves para reducir el riesgo de una aplicación recae en la reducción de la superficie de ataque.

Es posible eliminar posibles puntos de ataque si se deshabilitan módulos y/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.

5.2 Seguridad por defecto

Hay muchas maneras de entregar una experiencia "out of the box" ("listo para usar") a los usuarios. Sin embargo, **por defecto, la experiencia debe ser segura, facilitando, no obstante, la capacidad de reducción del nivel de seguridad si el usuario lo cree necesario.**

Por ejemplo, por defecto, deberían habilitarse las restricciones de complejidad mínima en las contraseña y su tiempo máximo de validez. Sin embargo, debe existir la opción para permitir al usuario deshabilitar estas dos características para simplificar el uso de la aplicación e incrementando, consecuentemente, el riesgo de que su contraseña pueda ser adivinada o robada.

Bajo este enfoque, la responsabilidad del nivel de seguridad definido y de la aceptación del riesgo derivado es delegada al usuario, no es impuesta por la propia aplicación.

Del lado de los desarrolladores, es 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, es recomendable estudiar sus efectos y consecuencias sometiéndola a pruebas de auditoría de seguridad.

5.3 Privilegios mínimos

El principio del mínimo privilegio recomienda que las cuentas tengan la mínima cantidad de privilegios necesarios para realizar sus actividades. Esto abarca a los derechos de usuario, permisos de recursos tales como límites de CPU, memoria, red y permisos del sistema de ficheros.

Asimismo, se recomienda que los procesos se ejecuten únicamente con los privilegios necesarios para completar sus tareas, ni mas, ni menos. De esta manera se limitan los posibles daños que podrían producirse si se ve comprometido el proceso.

Si un usuario malintencionado tomar el control de un proceso en un servidor o comprometer una cuenta de usuario, los privilegios concedidos determinarán en gran medida los tipos de operaciones que podrá llegar a realizar.

Un código que requiera confianza adicional (y privilegios elevados) deberá aislarse en procesos independientes. Igualmente, en una aplicación o sistema, una cuenta de usuario de administrador que cuente con privilegios elevados no debe usarse nunca para labores de operación rutinaria que pudieran ser realizadas usando una cuenta con menos privilegios.

Por ejemplo, si cierto servidor sólo requiere acceso de lectura a la tabla de una base de datos y la habilidad para escribir en un fichero log, exclusivamente se deben conceder los permisos para realizar estas dos operaciones. Bajo ninguna circunstancia deberían darse privilegios administrativos si no son necesarios.

5.4 Validación de datos

Garantizar que la aplicación sea robusta ante todas las formas posibles de ingreso, modificación o salida de datos, ya sean proporcionados por el usuario, por la infraestructura, por entidades externas o de bases de datos.

5.4.1 Validación de datos de entrada

Una premisa fundamental es **no confiar en los datos que el usuario pueda introducir**, ya que éste tiene todas las posibilidades manipularlos. La debilidad de seguridad más común en aplicaciones es la falta de validación apropiada de las entradas del usuario o del entorno. Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones, tales como inyecciones de código, la 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, control de la longitud de los datos introducidos, etc. Esta medida preventiva protege a la aplicación de usos incorrectos accidentales o ataques deliberados por parte de usuarios.

5.4.2 Modificación de datos

La modificación de los datos también produce un buen número de errores de seguridad. Esta modificación puede ir desde la concatenación de parámetros al incremento en el valor de un entero, por poner algunos ejemplos.

Verificar que la seguridad de la aplicación se mantiene consistente tras una modificación en los datos también debe ser obligatorio en toda aplicación segura.

5.4.3 Validación datos de salida

Por último, **los datos generados por la aplicación pueden ser objeto de numerosos problemas de seguridad**.

Pongamos por ejemplo el sistema de autenticación de un sitio web. Al hacer un inicio de sesión con un usuario no existente y contraseña errónea se devuelve un mensaje tipo “No existe el usuario”, y cuando se hace con un usuario existente, pero contraseña errónea, se devuelve un mensaje como “Contraseña incorrecta”. Estos mensajes de salida nos permiten conocer cuando un usuario existe o no en el sistema. Un atacante podría enumerar los usuarios existentes en el sistema con el fin de conseguir un diccionario válido de usuarios, para realizar un posterior ataque de fuerza bruta.

5.5 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 subsistemas de la aplicación**. Estos puntos de control ayudan a garantizar que sólo los usuarios autenticados y autorizados puedan obtener el acceso a la siguiente capa y a sus datos.

5.6 Control seguro de errores

Es necesario **controlar las respuestas de los errores** y no mostrar en ellas información que pudiera ayudar al atacante a descubrir datos acerca de cómo ha sido desarrollada o cómo funcionan los procedimientos de la aplicación. La información detallada de los errores producidos no debería ser mostrada al usuario, sino que debería ser enviada al fichero de log correspondiente.

Una simple página de error 403 (acceso denegado) puede indicar a un escáner de vulnerabilidades que un directorio existe, y permitirle realizar un mapa aproximado de la estructura de directorios mediante pruebas de ensayo y error.

5.7 Los sistemas externos son inseguros por defecto

"Si no es de su propiedad, no presuponga que alguien se esté ocupando de la seguridad por usted."

En la actualidad el número de organizaciones que optan por **contratar servicios a terceros**, como alojamientos o servicios "en la nube", va en aumento. El uso de sistemas externos tiene una serie de ventajas, pero también **es necesario tener en cuenta los riesgos derivados**. Los servicios en la nube son una "caja negra" en la que se deja en manos del proveedor del servicio la responsabilidad del almacenamiento de datos y su control. Es muy probable que los proveedores del servicio externo tengan diferentes políticas de seguridad y posturas a la suya. De ahí que la confianza implícita de ejecutar sistemas externos no esté garantizada.

Para un mayor detalle sobre seguridad del uso de servicios, software o infraestructuras en la nube, consulte el informe de AndalucíaCERT: **CERT-IF-1379-120120-Seguridad_en_la_nube**.

5.8 Separación de funciones

Un control clave contra posibles fraudes es 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 ordenador. 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, ciertos roles deben tener un nivel de confianza más elevado que los usuarios normales, como el caso del administrador, que posee privilegios avanzados como apagar y encender el sistema, configurar políticas de contraseñas así como los diferentes parámetros de la aplicación. Debido a estos privilegios, un administrador no debería ser capaz, siguiendo el ejemplo anterior, de hacer login en la aplicación como super-usuario con la capacidad de pujar por objetos en nombre de otros usuarios.

5.9 Evitar la seguridad por oscuridad

La seguridad de un mecanismo o aplicación no debería depender del secreto o confidencialidad de su diseño o implementación. Si se intentan ocultar secretos mediante el uso de nombres de variables engañosos o de ubicaciones de archivos no habituales, no estará mejorando la seguridad.

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 sea 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 debería basarse en muchos otros factores, incluyendo políticas razonables de contraseñas, defensa en profundidad, límites en las transacciones de negocios, arquitectura de red sólida, y controles de auditoria y fraude.

Un ejemplo práctico es Linux. El código fuente de Linux está ampliamente disponible, y aún así es un sistema operativo resistente, seguro y robusto.

5.10 Simplificar mecanismos de seguridad

"Haga todo tan simple como sea posible, pero no más simple."

El área de la superficie de ataque y la simplicidad van de la mano. **Los mecanismos de seguridad establecidos deben ser tan sencillos como sea posible.** Los desarrolladores deben evitar el uso de complejas arquitecturas destinadas a asegurar la aplicación, allí donde un enfoque simple sería más rápido y simple.

5.11 Gestionar correctamente los incidentes de seguridad

Una vez que un fallo de seguridad ha sido identificado, es importante desarrollar pruebas para su explotación, comprender la raíz del problema y determinar la forma de solucionarlo de una manera ágil.

Es muy recomendable publicar de forma periódica actualizaciones y parches de seguridad que resuelvan las vulnerabilidades descubiertas.

6 MADEJA

MADEJA es el Marco de Desarrollo de la Junta de Andalucía. Su misión es proporcionar un entorno que permita a todos los implicados en el desarrollo y en la explotación del software tener una referencia clara de cuáles son las directrices que han de guiar esta actividad, así como dar a conocer los recursos y herramientas que están a su disposición.

Dentro de este marco se contemplan algunas directrices de carácter obligatorio de seguridad para el desarrollo de aplicaciones en el entorno de la Junta de Andalucía.

- MADEJA: Marco de Desarrollo de la Junta de Andalucía.

7 CONCLUSIONES

Como hemos visto en este breve documento, la programación segura responde al deseo permanente del ser humano por avanzar en medio de la dificultad y confrontar el reto de la perfección. Sabemos que el software libre de errores es una meta prácticamente inalcanzable dado que los buenos hábitos y prácticas de programación no se han desarrollado suficientemente para avanzar en sólidos desarrollos de sistemas de información vistos tanto desde la perspectiva técnica como desde la óptica de administración de proyectos.

En este sentido, es menester que los profesionales más experimentados en el desarrollo de software, orienten y comuniquen su práctica en esta disciplina, ofreciendo un contexto práctico para las nuevas generaciones de desarrolladores, para que desde sus inicios fomenten acciones formales y buenos hábitos al enfrentarse al reto de un desarrollo.

8 REFERENCIAS

- Marco de desarrollo de la JdA (MADEJA)
- OWASP. Seguridad en el desarrollo de aplicaciones web