Seguridad en aplicaciones web en el ámbito de la e-Administración

Problemas más comunes y principales contramedidas

Seguridad en aplicaciones web en el ámbito de la e-Administración

Problemas más comunes y principales contramedidas

Fernando de la Villa
Gerente de Soluciones
Area Seguridad
Mnemo Evolution and Integration Services

Palabras clave
Seguridad, información, aplicaciones web, contramedidas, controles, seguridad en desarrollo

Resumen de la Comunicación

La entrada en vigor de la Ley de Acceso Electrónico de los Ciudadanos a los Servicios Públicos implica la progresiva incorporación de multitud de nuevos servicios públicos electrónicos. El éxito de estos nuevos servicios y también de los ya existentes pasa por lograr un alto nivel de e-Confianza en ciudadanos y empresas para que los utilicen de forma preferente.

En este escenario resulta fundamental cuidar al máximo la seguridad que afecta a las aplicaciones web de forma más directa y que es la más visible para los usuarios finales. La presente comunicación ofrece una breve recopilación de los problemas que actualmente se encuentran con más frecuencia en las aplicaciones web y algunas recomendaciones y buenas prácticas de seguridad dirigidas a mitigarlos.

1. INTRODUCCIÓN

La entrada en vigor de la Ley de Acceso Electrónico de los Ciudadanos a los Servicios Públicos implica la progresiva incorporación de multitud de nuevos servicios públicos electrónicos. El éxito de estos nuevos servicios y también de los ya existentes pasa por lograr un alto nivel de e-Confianza en ciudadanos y empresas para que los utilicen de forma preferente.

La propia Ley, indica en su exposición de motivos que el principal reto en la implantación de las TIC, tanto en la sociedad en general como en la propia Administración, es la generación de confianza que elimine los riesgos asociados a su uso. Para conseguir un alto nivel de confianza es imprescindible lograr el más alto nivel de seguridad en los servicios públicos electrónicos, lo que redunda en un mayor nivel de calidad percibido por ciudadanos y empresas acerca de los mismos.

En este escenario resulta fundamental cuidar al máximo la seguridad que afecta a las aplicaciones web de forma más directa y que es la más visible para los usuarios finales. Esto no significa en absoluto que se puedan descuidar otros aspectos. En la provisión de un servicio a través de la web intervienen diferentes combinaciones de hardware y software estructurados en diferentes capas. La seguridad de cada uno de estos elementos es igualmente importante para proporcionar un alto nivel de seguridad final.

Dada la complejidad y extensión del tema solamente se ha realizado aquí una breve recopilación de los problemas que actualmente se encuentran con más frecuencia en las aplicaciones web y algunas recomendaciones de seguridad para mitigarlos. La recopilación se basa en el resultado de diversos análisis de referencia y en nuestra experiencia en auditorías de seguridad en general y en actividades de *hacking* ético en particular.

En cualquier caso, se recomienda ampliar la información aquí contenida, acudiendo a los estándares y buenas prácticas internacionales cuya aplicación siempre ayuda a elevar el nivel de seguridad no solo de cualquier aplicación web sino de toda la infraestructura informática y de comunicaciones que se encuentra por debajo.

2. PROBLEMAS DE SEGURIDAD MÁS COMUNES

Si bien son muchos los problemas de seguridad que pueden afectar a una aplicación web, también es cierto que existe una serie de problemas comunes a la mayoría de ellas y que son las aprovechadas con mayor frecuencia por atacantes expertos y noveles. Reflejo de ello son los resultados que muestran los análisis más conocidos como son OWASP *top* 10, MITRE *Common Weakness Enumeration*, SANS *top* 20 *Attack Targets* y WASC *Web Application Security Statistics Project*.

Tomando como referencia la lista del *Open Web Application Security Project* (OWASP *top* 10 2007), los problemas de seguridad de aplicaciones web más frecuentes son los siguientes:

1. *Cross Site Scripting* (XSS)
2. Problemas de inyección de código
3. Ejecución maliciosa de código
4. Referencias directas inseguras a objetos
5. *Cross Site Request Forgery* (CSRF)
6. Fuga de información y gestión de errores inapropiada
7. Problemas de autenticación y gestión inapropiada de sesiones
8. Almacenamiento criptográfico inseguro

9. Comunicaciones inseguras
10. Fallos en la restricción de acceso mediante URL

Comparando esta lista con los problemas identificados por los otros tres análisis mencionados se observan claramente cuatro comunes a todos ellos y que figuran, además, en los primeros puestos. Todos ellos, exceptuando el *Cross Site Request Forgery*, aparecen en ediciones anteriores de estos análisis, por lo que siguen siendo problemas importantes que afectan a la seguridad de aplicaciones web y que deben ser tenidos en cuenta como prioritarios a la hora de afrontar el desarrollo y la explotación de este tipo de aplicaciones.

A continuación se tratan brevemente los problemas más destacados, sus causas y algunas de las contramedidas aplicables en cada caso.

Cross Site Scripting (XSS)

Se trata de una vulnerabilidad muy extendida en las aplicaciones web que, en realidad, es un subtipo de inyección de código. Normalmente toma la forma de un hiperenlace que contiene código malicioso cuyo contenido se suele ofuscar para no dar pistas evidentes de sus intenciones al usuario.

El funcionamiento de este tipo de ataque es realmente simple. Basta con lograr que el usuario haga clic en uno de estos hiperenlaces que se le puede hacer llegar fácilmente mediante correo electrónico, por mensajería instantánea o mediante su inserción en los comentarios de algún *blog* o foro.

Cuando el usuario hace clic en el enlace se ejecuta en su propio cliente web el código en cuestión que puede llevar al secuestro de su sesión, redirigir su navegación a una web maliciosa con aspecto legítimo (el tristemente famoso *phising*), incluso tomar el control del cliente web aprovechando alguna vulnerabilidad del mismo. Habitualmente se inyecta código JavaScript aunque podría ser utilizado código en cualquier otro lenguaje que sepa interpretar el cliente web del usuario afectado.

La mejor manera de proteger las aplicaciones web contra este tipo de vulnerabilidad consiste en el filtrado de los metacaracteres clásicos correspondientes a la sintaxis de los lenguajes que se utilizan para la inyección, como los caracteres “<”, “>”, “#”, “&” y las comillas simples y dobles, por citar algunos.

Inyección de código

Además del tipo de inyección que supone el Cross Site Scripting y que merece mención aparte, existen otro tipo de inyecciones de código no menos importantes. En especial, la inyección de código SQL en aplicaciones que sean vulnerables a este vector de ataque. Normalmente el objetivo final de un atacante es el robo de información sensible que, en la mayoría de los casos, se encuentra almacenada en una base de datos. Por esta razón, si la aplicación web es vulnerable a un ataque de inyección SQL, el atacante no necesita buscar otros puntos débiles que pueda tener el software instalado en los servidores para lograr su objetivo.

La inyección de código SQL viene causada por la utilización de sentencias SQL embebidas en la aplicación y el incorrecto filtrado de la entrada de datos del usuario. Todavía es habitual ver cómo las sentencias SQL se construyen en tiempo de ejecución de la aplicación, insertando directamente los datos introducidos por el usuario en las mismas para finalmente ordenar su ejecución al sistema gestor de bases de datos. Si no se filtra correctamente la información introducida por el usuario en un formulario web, se posibilita la introducción de comandos SQL que interrumpen el flujo normal de ejecución de sentencias SQL de la aplicación y permite

realizar acciones malintencionadas como, por ejemplo, el salto de una pantalla de identificación sin necesidad de introducir la contraseña, el borrado de datos de la base de datos y la obtención de información almacenada en la misma.

Para prevenir la inyección de código SQL también se puede aplicar una sencilla contramedida como es el uso de procedimientos almacenados. El código de estos procedimientos ha sido preprocesado por el sistema gestor de base de datos por lo que los datos recibidos se tratan como tales y no se utilizan para construir y posteriormente analizar, optimizar y ejecutar la sentencia SQL correspondiente. En el peor de los casos podría llegar a almacenarse en la base de datos información inadecuada si no se ha realizado un filtrado previo, pero no llega a romperse el flujo normal de ejecución, eludiendo así el problema de seguridad que supone el intento de ataque. La precaución que hay que tomar, por razones obvias, es no utilizar nunca procedimientos que ejecuten una sentencia SQL pasada como parámetro que soportan algunos sistemas gestores de bases de datos.

Por otro lado, la utilización de procedimientos almacenados presenta ventajas adicionales. Al no necesitar ser analizada la sintaxis de cada sentencia y optimizar el camino de ejecución de la misma por parte del sistema gestor de base de datos se obtiene un ahorro de tiempo considerable, especialmente en aquellas aplicaciones que hacen uso intensivo de la base de datos. De igual modo, el código SQL se mantiene en la capa de datos en lugar de estar entremezclado con código en las demás capas que componen la aplicación, facilitando así el mantenimiento de la misma.

Pero no solamente es posible la inyección de código SQL en una aplicación web. En ocasiones también es factible la inyección de comandos del sistema operativo. Para evitarlo se debe impedir siempre que sea posible la invocación directa de comandos de sistema operativo desde la aplicación y, en todo caso, se debe evitar la invocación con parámetros que de un modo u otro hayan sido introducidos por un usuario. Asimismo se deben deshabilitar los procedimientos del sistema gestor de base de datos que permitan la ejecución de comandos del sistema operativo.

Además de las medidas a tomar en el desarrollo del código para evitar estos ataques no se debe olvidar una adecuada configuración del sistema gestor de base de datos. Por un lado es necesario realizar una correcta gestión de los usuarios de base de datos que utilizará la aplicación web para realizar su trabajo, así como de los permisos que tiene cada uno de ellos. En caso de no ser esto posible se debe poner especial cuidado en que sea utilizado solamente por un usuario específico para ello y en los puntos de la aplicación en los que sea estrictamente necesario.

Cross Site Request Forgery (CSRF)

Esta sencilla técnica, conocida también por el nombre *Session Riding*, cotiza al alza y se perfila como uno de los vectores de ataque más habituales en el futuro próximo. De hecho se especula con la posibilidad de que llegue a superar al *Cross Site Scripting* en número de intentos de ataque, motivo por el cual la mayoría de los *rankings* de vulnerabilidades antes mencionados han decidido introducirla recientemente en los primeros puestos.

Cross Site Request Forgery se basa en forzar que el cliente web de un usuario efectúe una petición HTTP al servidor web elegido por el atacante, distinto del dominio visitado por el usuario en un principio. La petición se realiza de modo que el usuario no sea consciente de ello. Para que el ataque funcione con éxito el usuario debe haberse identificado correctamente en la web seleccionada por el atacante, de lo contrario no sería efectiva. De este modo, al llevarse a cabo desde el propio cliente web de la víctima y utilizando una sesión abierta por el usuario, la petición parece completamente legítima al servidor web destinatario y como tal

podría ser ejecutada y quedar reflejada en sus registros. Esto supone un gran peligro para la víctima que tendría muy difícil demostrar que dicha petición no la realizó por voluntad propia.

El ataque se puede materializar de múltiples maneras. Por ejemplo, se puede visitar un foro en el que aparece un post que contiene una supuesta imagen que en realidad es una petición GET al web elegido por el atacante. El cliente web, al encontrar el *tag* HTML de la imagen realiza una petición de la URL maliciosa que es el objetivo perseguido. El usuario no notará nada extraño y casi con toda seguridad no se percatará de lo que realmente ha sucedido.

Las posibilidades que brinda CSRF son muy amplias. Puede facilitar la realización de transacciones no deseadas como pujas en subastas electrónicas y transferencias dinerarias, así como la publicación de contenidos comprometedores en *blogs* y foros. Además se trata de un ataque aplicable no solamente al entorno Internet, sino también dentro de una red interna. Por ejemplo, abre la posibilidad de modificar la configuración de dispositivos de red y de seguridad a través de sus interfaces web de configuración si la víctima es un administrador de sistemas que mantenga abierta alguna sesión administrativa con estos dispositivos.

Una dificultad añadida radica en la dificultad de detectar esta vulnerabilidad de forma automática. La mera utilización de herramientas automáticas de detección de vulnerabilidades por parte de los propietarios de las aplicaciones web no garantiza en absoluto su detección. Normalmente será necesario comprobar de forma manual que la vulnerabilidad no existe aunque se hayan tomado precauciones para evitarla.

Por si esto fuera poco no es una vulnerabilidad fácil de prevenir. El objetivo básico de las contramedidas que se apliquen debe consistir en validar las peticiones de un usuario para discernir si se han realizado de forma automática o las ha realizado el usuario con la mejor de las intenciones. Para ello se pueden aplicar en los formularios web técnicas como los gráficos CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) que si se aplican correctamente ayudan a reducir considerablemente el éxito de las peticiones automáticas. Otra posibilidad es la utilización de *tokens* de sesión adicionales a los identificadores de sesión utilizados ampliamente por las aplicaciones web. Son números aleatorios proporcionados por el servidor web al usuario, que no pueden ser conocidos por el atacante de antemano y que deben ser enviados (nunca mediante una *cookie*) de nuevo al servidor junto a la petición legítima.

En el ámbito de la administración electrónica, con la extensión paulatina de mecanismos de firma electrónica como el proporcionado por la FNMT y la contenida en el DNI electrónico, en aquellos casos que no sea imprescindible su uso se podría estudiar la posibilidad de firmar una petición por el usuario, que deberá introducir el correspondiente PIN, para impedir la realización de peticiones automáticas maliciosas.

En cualquier caso, atendiendo al principio de proporcionalidad, no se trata de utilizar estas contramedidas en todos y cada uno de los formularios disponibles en una aplicación web. Solamente en aquellos que puedan suponer un riesgo para el ciudadano o la empresa que esté utilizando la aplicación web en cuestión.

Fuga de información

Habitualmente se habla de fuga de información en aplicaciones web haciendo referencia a la posibilidad de obtener datos sobre la propia aplicación. En definitiva, se busca información sensible como la configuración de la aplicación, pistas sobre el funcionamiento interno, incluso la posibilidad de obtener el código fuente en su totalidad o en parte. Esta información puede no constituir un problema en sí porque de forma aislada no suele ser útil. Sin embargo, en muchas

ocasiones es esta información la que permite intentar otros vectores de ataque con posibilidades de éxito y que finalmente el atacante acabe alcanzando su objetivo.

Una forma de obtener información sobre una aplicación web es a través de los mensajes de error generados, que pueden llegar a mostrar secuencias de llamadas que dan una idea del funcionamiento interno de la aplicación y que, en ocasiones, muestran detalles internos muy útiles para el atacante. Para evitarlo se debe asegurar que la aplicación realiza una correcta gestión de errores y deshabilitar en el entorno de producción los mensajes de error al usuario que contengan información técnica.

Posiblemente la posibilidad más peligrosa es la obtención del código fuente de la aplicación. Disponer de este código permite buscar puntos débiles, especialmente en lo referente al tratamiento de datos introducidos por el usuario y en las peticiones que se realizan a la base de datos. La forma más apropiada de evitar este problema es disponer de procesos adecuados de gestión del cambio que, entre otras cosas, impidan la llegada del código fuente junto con el código objeto correspondiente al entorno de producción y que asegure la inexistencia de archivos innecesarios, como por ejemplo las habituales copias de seguridad de archivos con extensiones conocidas (.old, .bkp, .bak, .tmp, etcétera).

Autorización insuficiente

Si bien OWASP top 10 lo ubica en séptima posición y MITRE CWE en la decimoquinta, según nuestra experiencia en la realización de auditorias de aplicaciones web es uno de los problemas de seguridad más habituales. En un sentido amplio, consiste en acceder a información o funcionalidades restringidas que, además de estar correctamente identificado, exigen que el usuario tenga los permisos necesarios.

Sin embargo, en ocasiones se producen olvidos en la introducción de mecanismos de autorización en las aplicaciones que abren la puerta a un posible atacante a ciertas funcionalidades que en condiciones normales tendría vetadas. De igual modo, en ocasiones ni siquiera se introducen mecanismos de autorización en determinadas páginas, sino que se confía en que solamente accederán aquellas personas cuyo acceso les aparece en forma de opción en el correspondiente menú. Si no se protege el canal mediante cifrado, un simple *sniffer* permitirá a un atacante conocer la forma de acceder estas páginas restringidas, algunas veces incluso sin estar identificado en la aplicación.

El mismo problema se da en el caso de la incorrecta exposición de interfaces administrativas de cara a Internet. En ocasiones se publica una interfaz administrativa basando su seguridad únicamente en la supuesta dificultad de conocer la URL apropiada para acceder a la misma ("*security by obscurity*"). Existen herramientas automáticas que prueban miles de posibles URLs con valores utilizados frecuentemente para acceder a este tipo de interfaces y que en más de una ocasión encuentran una interfaz de este tipo. A veces incluso se ha encontrado una interfaz completamente desprotegida confiando de nuevo en la falsa sensación de seguridad que proporciona el no conocer la URL adecuada. Obviamente, resulta fundamental proteger adecuadamente todas y cada una de las páginas que componen una interfaz administrativa. En cualquier caso lo mejor es no exponer a una red pública ninguna interfaz administrativa y, en todo caso, utilizar cifrado en el acceso a la interfaz.

Desde el punto de vista del desarrollo se deben aprovechar los mecanismos de seguridad que proporcionan los distintos *frameworks* utilizados para asegurar una aplicación coherente y uniforme de autenticación y autorización a lo largo y ancho de toda la aplicación web. Asimismo se debe asegurar una correcta gestión de usuarios de modo que solamente tengan acceso a las opciones y funcionalidades restringidas aquellos usuarios que deban tenerlo, y durante el tiempo que deban tenerlo.

3. PRINCIPALES CONTRAMEDIDAS Y BUENAS PRÁCTICAS

Tal y como se indicaba al principio, la seguridad de una aplicación no solamente depende de su código, sino de toda una serie de mecanismos de protección, detección y prevención aplicados a las diferentes capas que conforman su arquitectura que finalmente hacen que la aplicación web funcione y atienda las peticiones de los usuarios. En este sentido referencias como la guía de buenas prácticas que proporciona el estándar internacional ISO/IEC 27002:2005 son fundamentales. Contempla no sólo controles a nivel técnico sino también los controles organizativos necesarios para asegurar un nivel de protección adecuado y proporcional a la relevancia de los activos que se desean proteger.

A continuación se destacan algunas de las contramedidas y buenas prácticas recomendadas para el desarrollo y la explotación de una aplicación web, que ayudan a mitigar los problemas de seguridad antes mencionados y dificultan adicionalmente otro tipo de ataques.

Validación de datos de entrada

Un aspecto fundamental para la seguridad de una aplicación es la correcta validación de los datos de entrada. No solamente para asegurar que los datos introducidos por el usuario presentan el formato esperado por la aplicación, sino también para frustrar cualquier intento de inyección de código sea del tipo que sea. En todo caso es importante recordar que la validación siempre se debe hacer en el servidor, nunca sólo en cliente. La validación en cliente puede ser útil para ayudar a los usuarios legítimos a proporcionar los datos con el formato adecuado y descargar así al servidor de algunas peticiones que generarían errores por este motivo. Sin embargo, un atacante siempre va a poder saltarse estas comprobaciones en cliente utilizando cualquiera de las herramientas especializadas disponibles para manipular las peticiones web o, incluso, generar sus peticiones de forma manual o con la ayuda de algún lenguaje de *scripting*. Por este motivo, la única validación realmente efectiva es la realizada en la parte servidora de la aplicación.

Con respecto a la forma de validar los datos desde el punto de vista de seguridad, la opción más habitual consiste en filtrar aquellos caracteres que forman parte de la sintaxis de los lenguajes habitualmente utilizados para inyección de código. Es más, no sólo basta con filtrar los caracteres como tales, sino que es necesario filtrar sus distintas codificaciones en notaciones utilizadas habitualmente en entornos web (formato hexadecimal, URL *encoded*, etcétera). Por ejemplo, caracteres como el “mayor qué” y el “menor qué”, salvo ciertos casos, no tienen porqué formar parte de la entrada de datos de un usuario y su filtrado no supone ningún trastorno para el usuario legítimo. Otra posibilidad consiste en admitir solamente un conjunto de caracteres concreto en el que por supuesto no figuran los caracteres mencionados, rechazando cualquier dato que contenga un carácter fuera del conjunto.

Independientemente de la postura que se adopte en la validación de datos resulta fundamental que se aplique a todos y cada uno de los campos disponibles en un formulario, incluidos los campos ocultos y sin olvidar los datos pasados como parámetro en una petición GET. En más de una ocasión se consigue realizar una inyección con éxito porque no se aplicó el filtrado en un simple campo de un formulario, estando correctamente filtrados todos los demás. La mejor manera de evitar este problema es utilizar un mecanismo de validación de datos uniforme para toda la aplicación que actúe como una capa intermedia por la que deben pasar todos los datos de entrada antes de ser utilizados internamente por el resto de la aplicación.

Otro factor a tener en cuenta en verificar la longitud de los datos para evitar los tradicionales ataques de desbordamiento de buffer. En función de los lenguajes y las tecnologías utilizadas

para el desarrollo de la aplicación web este aspecto puede tener mayor o menor importancia, pero en cualquier caso resulta conveniente realizar esta comprobación.

Por último, las validaciones no sólo se deben aplicar a los datos introducidos por el usuario en un cliente web. Es igualmente importante aplicarlas a los datos que provienen de otros sistemas independientemente del formato en que se reciban, si bien se debe prestar especial atención a los formatos en texto plano porque son más fáciles de manipular por un atacante. Además, no solamente hay que contemplar recepciones puntuales de información, por ejemplo, a través de *web services*. También es necesario contemplar importaciones masivas de datos, archivos descargados por la aplicación que posteriormente son procesados para su almacenamiento en base de datos, etcétera.

Gestión de errores y gestión de logs

Un atacante, salvo en los casos en que cuente con información previa y detallada sobre la aplicación web, necesita realizar múltiples intentos infructuosos de ataque hasta alcanzar o no el éxito. Si se realiza una correcta gestión de errores y excepciones, y se aplica una gestión de logs adecuada, se podrán detectar los ataques y actuar a tiempo.

Las plataformas actuales proporcionan diversas facilidades de gestión de errores y excepciones, así como funcionalidades para reflejar en archivos de log y/o base de datos todos los sucesos que se consideren relevantes. Conviene aprovechar estos mecanismos para aplicar una gestión uniforme en toda la aplicación y obtener registros útiles que permitan reconstruir en caso necesario la secuencia de hechos acaecidos en una aplicación.

Es importante que los errores mostrados al usuario no sean de carácter técnico. Por un lado este tipo de errores no aportan nada al usuario y sí podría hacerlo, y de forma decisiva, a un atacante. Por otro lado, la experiencia del usuario se ve perjudicada si el usuario observa un mensaje que no entiende, en un formato técnico, en lugar de un sencillo texto indicando que se ha producido un error y con un aspecto coherente con el diseño de la aplicación web. No obstante, los mensajes de tipo técnico sí deben ser almacenados en log para que el personal de soporte y el equipo de mantenimiento puedan determinar las causas y dar los pasos necesarios para subsanar el problema.

Monitorización y gestión de eventos de seguridad

De poco serviría una exhaustiva gestión de logs por parte todas las capas que conforman una aplicación web si esos logs no se revisan adecuadamente. Casi tan importante como aplicar contramedidas y controles es la monitorización de la aplicación en su conjunto, y no solamente desde el tradicional punto de vista de sistemas y comunicaciones, sino también desde el punto de vista de seguridad.

Un aspecto absolutamente fundamental para llevar a cabo una correcta monitorización es la sincronización horaria de todos los servidores y dispositivos. Esto ayuda a la resolución de problemas técnicos, permite reconstruir con precisión los hechos acaecidos en el sistema de cara a evaluar el alcance de un incidente de seguridad y permite realizar una correcta correlación de eventos.

Lo ideal aquí es disponer de un sistema de gestión de la seguridad que recopile toda la información relativa a eventos de seguridad generados por los distintos servidores y dispositivos y que permita aplicar correlación a dicha información para generar alertas de seguridad útiles. Un sistema que facilite una completa gestión del ciclo de vida de las alertas generadas y que disponga de un cuadro de mandos que permita reflejar las métricas de seguridad que se definan y permita comprobar su evolución a lo largo del tiempo.

Otros aspectos a tener en cuenta

En el plan de pruebas de la aplicación no puede faltar una exhaustiva batería de pruebas de seguridad que compruebe el correcto comportamiento ante intentos de ataque. En este sentido se debe hacer especial hincapié en la verificación de los datos de entrada que recibe la aplicación, la identificación y la autorización de usuarios y la integridad de sesiones, sin descuidar otros aspectos como la gestión de errores, la generación de logs y la resistencia ante ataques concretos como el Cross Site Request Forgery. Los resultados de las pruebas realizadas deben ser correctamente documentados para garantizar que no se ha producido la omisión accidental de ninguna prueba y cómo se ha comportado la aplicación en cada uno de los casos.

Un solo descuido en una tarea de mantenimiento puede introducir una vulnerabilidad fatal para la seguridad de la aplicación. Por ello se deben realizar las pruebas pertinentes de seguridad, además de las puramente funcionales, antes de poner en producción cualquier cambio derivado de una tarea de mantenimiento de la aplicación, haya implicado cambios o no en el código de la misma.

Una forma de descubrir posibles problemas de seguridad es la auditoría de código fuente focalizada en la seguridad. Estas auditorías pueden ser especialmente útiles en el caso de aplicaciones muy grandes en cuyo desarrollo han intervenido diferentes grupos de trabajo y personal tanto interno como externo. Pero también son útiles en otros casos porque permiten encontrar problemas de seguridad que no son fáciles de localizar con otras técnicas, como la existencia de puertas traseras y bombas lógicas.

No hay que olvidar que continuamente están apareciendo nuevas vulnerabilidades y técnicas de ataque. Por esta razón es conveniente realizar auditorías periódicas de la seguridad de la aplicación y de toda su infraestructura. De no hacerlo de forma repetida se corre el riesgo de tener la sensación de falsa de seguridad por haber pasado con éxito una primera auditoría. En cualquier momento puede aparecer un problema por una nueva vulnerabilidad descubierta en software de terceros o una vulnerabilidad introducida en una tarea de mantenimiento que no fue descubierta antes de su paso a producción. Es importante que estas auditorías se realicen siguiendo metodologías como OSSTMM (Open Source Security Testing Methodology Manual), OWASP o ISSAF (Information Systems Security Assessment Framework), o alguna metodología propia basada en todo o en parte en estas metodologías. De este modo se proporciona el rigor necesario a los trabajos realizados y se asegura el desarrollo de una auditoría completa que contempla todos los aspectos necesarios.

De forma análoga hay que poner especial cuidado en la aplicación de tecnologías más recientes como AJAX. Estas tecnologías abren nuevas posibilidades de ataque al cambiar la forma de interactuar con las aplicaciones web y por tanto son desarrolladas y deben ser probadas de manera distinta a la habitual.

Por último, tal como indica la Ley de Acceso Electrónico de los Ciudadanos a los Servicios Públicos en su artículo 4.g, se debe aplicar el principio de proporcionalidad en la aplicación de las garantías y medidas de seguridad. Para hacerlo se debe realizar un análisis de riesgos que, mediante la valoración de los activos de información que participan en la provisión de un servicio público electrónico, ofrezca una aplicación de la seguridad en su justa medida en función de las necesidades y la criticidad del servicio.

4. CONCLUSIONES

La seguridad de una aplicación web y las infraestructuras que la soportan no es un asunto trivial. Si se pretende realizar con las mayores garantías se debe incorporar en todos y cada uno de los pasos del ciclo de vida de la aplicación, desde su concepción inicial hasta su retirada de explotación.

Las tendencias actuales dejan entrever que las vulnerabilidades de plataforma, utilizadas tradicionalmente para la realización exitosa de ataques, están perdiendo peso para dejar paso a los vectores de ataque dirigidos contra las aplicaciones web. Esto se debe en gran medida a los tradicionales problemas de seguridad que se derivan del proceso de desarrollo y es el principal motivo por el que se debe prestar la mayor atención a la seguridad en este proceso.

La seguridad es un aspecto más que configura el nivel de calidad de los servicios prestados electrónicamente a ciudadanos y empresas. Solamente mediante la inclusión a todos los niveles de los controles y buenas prácticas de seguridad proporcionados por los estándares y las prácticas comúnmente aceptadas será posible ganar la confianza de ciudadanos y empresas para que utilicen de forma masiva los servicios públicos ofrecidos de forma electrónica.

5. REFERENCIAS

- OWASP top 10 2007
  http://www.owasp.org/index.php/Top_10_2007

- MITRE Common Weakness Enumeration – Vulnerability Trends
  http://cwe.mitre.org/documents/vuln-trends/index.html

- SANS Top 20 Internet Security Attack Targets
  https://www2.sans.org/top20/

- WASC Web Application Security Statistics Project.
  http://www.webappsec.org/projects/statistics/

- OWASP Guide
  http://www.owasp.org/index.php/OWASP_Guide_Project

- ISECOM Open Source Security Testing Methodology Manual (OSSTMM)
  http://www.isecom.org/osstmm/

- OISSG Information Systems Security Assessment Framework (ISSAF)
  http://www.oissg.org/issaf

