Sociedad Andaluza para el Desarrollo de las Telecomunicaciones S.A.
CONSEJERÍA DE ECONOMÍA, INNOVACIÓN, CIENCIA Y EMPLEO

AndalucíaCERT
CENTRO DE SEGURIDAD TIC

Informe de divulgación

OWASP Top 10 - 2013
Los 10 riesgos de seguridad más importantes en aplicaciones web

Tipo de documento:	*Informe*
Autor del documento:	*AndalucíaCERT*
Código del Documento:	*CERT-IF-4062-130827*
Edición:	*0*
Categoría	*Público*
Fecha de elaboración:	*27/08/2013*
N° de Páginas	*1 de 25*

1 TABLA DE CONTENIDOS

2 OBJETO

En este documento se identifican y describen los diez riesgos de seguridad en aplicaciones web considerados como más importantes en la actualidad según OWASP. Esta información está basada en el proyecto "OWASP Top 10 – 2013" y tiene por objetivo crear conciencia sobre la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan organizaciones de todo tipo.

3 ALCANCE

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

4 INTRODUCCIÓN

La presencia de software inseguro debilita nuestra infraestructura. A medida que ésta se vuelve más compleja y está más interconectada, aumenta la dificultad de conseguir que las aplicaciones sean seguras.

Un punto importante para empezar a mejorar la seguridad de nuestras aplicaciones es identificar los posibles riesgos a los que se exponen, para seguidamente poder llevar a acabo un análisis del riesgo donde valorar el impacto que pueden tener en la organización si llegase a tener lugar.

En este sentido y con el fin de mejorar la seguridad de su infraestructura, este documento presenta un listado con los 10 riesgos de seguridad más comunes en aplicaciones web basado en información publicada por OWASP, una de los proyectos orientados a la mejora de la seguridad en aplicaciones web más importantes del mundo.

5 OWASP – Proyecto libre de seguridad en aplicaciones web

OWASP (The Open Web Application Security Project) es un proyecto libre dedicado a trabajar por y para la seguridad en aplicaciones web. Creado en 2001, es soportado y coordinado por un organismo sin ánimo de lucro denominado "Fundación OWASP". Consiste en una comunidad a nivel mundial enfocada en la mejora de la seguridad en las aplicaciones de software. Esta comunidad está compuesta por empresas, organizaciones educativas y particulares de todo el mundo, estando abierta a la participación de cualquier persona y/o patrocinador.

Está organizado en proyectos y capítulos locales repartidos por todo el mundo dedicados al desarrollo de documentación, herramientas y estándares de código libre.

Su trabajo podría separarse en 2 direcciones:

- Proyectos de documentación
- Proyectos de desarrollo

5.1 Proyectos de documentación

Existen un gran número de proyectos de documentación impulsados por OWASP. Entre ellos destacaremos los siguientes:

- Guía OWASP – Guía detallada sobre la seguridad de las aplicaciones web.
- OWASP Top 10 – Documento de alto nivel que se centra sobre los riesgos más críticos y comunes de las aplicaciones web.
- Métricas – Un proyecto para definir métricas de seguridad aplicables a aplicaciones web.
- Legal – Un proyecto para ayudar a los vendedores y compradores de software a negociar adecuadamente los aspectos de seguridad en sus contratos.
- Guía de desarrollo - Documento que cubre todos los aspectos de la seguridad en aplicaciones y servicios web para su aplicación en las fases de desarrollo.
- Guía de pruebas – Una guía de pruebas efectiva para auditar la seguridad de aplicaciones web.
- ASVS (Application Security Verification Standard) - Propuesta de estándar o marco de referencia a través del cual se deben implementar los análisis de seguridad a aplicaciones web.
- AppSec FAQ – Preguntas y respuestas frecuentes sobre seguridad de aplicaciones web.
- Policy frameworks – Documentos de apoyo para organizaciones que realicen revisiones de seguridad según diferentes normativas (ISO 27002, COBIT, ...).
- Especificación de requisitos legales para aplicaciones web – Con objetivo de extraer aquellos aspectos de la **legislación española** que tienen repercusión sobre la especificación de requisitos de las aplicaciones web, y construir una especificación de requisitos legales para aplicaciones web.

5.2 Proyectos de desarrollo

Existen un gran número de proyectos de desarrollo impulsados por OWASP. Entre ellos destacaremos los siguientes:

- ESAPI (Enterprise Security API) – Colección gratuita y abierta de un gran número de métodos de seguridad que un desarrollador puede necesitar para construir una aplicación web segura. Se encuentra disponible para lenguajes como PHP, JAVA, Python, .NET, ASP, Ruby, entre otros, además de ofrecer documentación para su implementación a través del uso de patrones de software.
- WebScarab – Un aplicación de chequeo de vulnerabilidades de aplicaciones web.
- Filtros de validación – Filtros genéricos de seguridad perimetral que los desarrolladores pueden usar en sus propias aplicaciones.
- WebGoat – Una herramienta interactiva de formación y benchmarking para que los usuarios aprendan sobre seguridad de aplicaciones web de forma segura y legal.
- DotNet – Un conjunto de herramientas para securizar los entornos .NET.

- OWASP DNIe - Evaluar y mejorar la seguridad de las aplicaciones web que hacen uso del DNI electrónico (DNIe).
- AntiSammy - API que ayuda a asegurar que los datos de entrada que proporcionan los usuarios estén en cumplimiento con las reglas de la aplicación, evitando la inclusión de código malicioso.

6 OWASP TOP 10 - 2013

Este documento está centrado en la información aportada por uno de los proyectos impulsados por OWASP. Concretamente el "OWASP Top 10".

Se trata de una lista de los 10 problemas de seguridad más críticos y frecuentes detectados en aplicaciones web. Nace en 2004 del consenso y debate entre expertos en la materia de la seguridad en aplicaciones web.

Su **misión** es concienciar y educar a desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad y los riesgos más importantes en el ámbito de las aplicaciones web, identificando y remarcando las diez amenazas más serias a las que se expone. El Top 10 provee técnicas básicas sobre cómo protegerse en estas áreas de alto riesgo y también orientación sobre los pasos a seguir para su prevención.

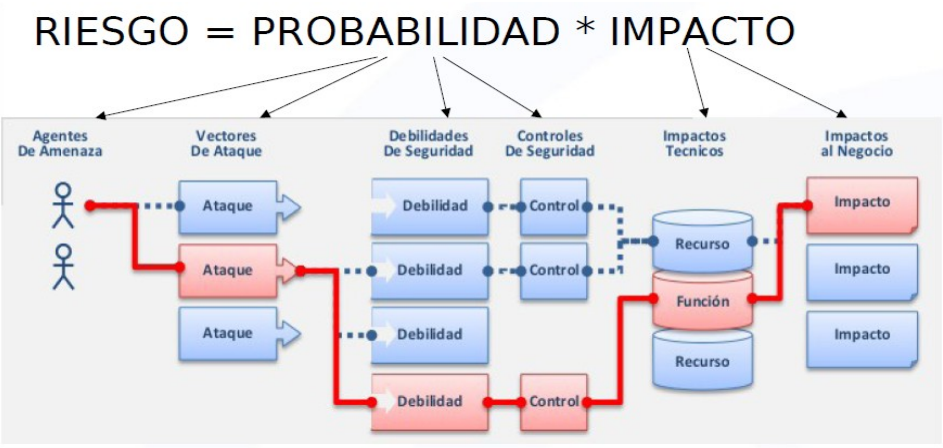
Actualmente se han publicado 4 ediciones, siendo el año 2013 cuando se ha dado a conocer su última versión, conocida como "OWASP Top 10 – 2013".

7 CLASIFICACIÓN DEL RIESGO

Para conseguir provocar un daño en su aplicación, y por consiguiente en su negocio, los atacantes pueden usar un gran número de posibles rutas. Cada uno de estos posibles caminos representa un riesgo que dependiendo de cada escenario concreto, deberá ser tenido en cuenta, porque de llegar a aprovecharse podría provocar un fuerte impacto en nuestra organización.

A veces, estas rutas son triviales de encontrar y explotar y a veces son extremadamente difíciles. De manera similar, el daño causado podrá ir de ninguno hasta ser extremadamente severo.

En el modelo propuesto por OWASP, para determinar el riesgo asociado a su organización, puede evaluar la probabilidad asociada con cada **agente de amenaza**, **vector de ataque** y **debilidad de seguridad** y combinarla con una estimación del **impacto técnico** y **de negocios** en su organización.



Juntos, estos factores nos permitirán determinar el valor total del riesgo.

El OWASP Top 10 se enfoca en la identificación de los riesgos más serios para un amplio espectro de organizaciones. Al final del documento, para cada uno de estos riesgos, se provee una tabla con información genérica acerca de la probabilidad y el impacto técnico asociado a estos riesgos basado en la Metodología de Evaluación de Riesgos OWASP.

8 TOP 10

8.1 Inyección (A1)

La mayoría de las aplicaciones están diseñadas para su interacción con el usuario. Para ello se ponen a disposición del usuario diferentes medios por los que se puede comunicar con la aplicación, como por ejemplo, pulsar un botón, mover el ratón sobre una determinada zona o introducir un texto en un formulario. Estas acciones envían información a la aplicación para que un intérprete las entienda y le indique al sitio web qué es lo que quiere el usuario hacer.

El problema surge cuando, debido a una mala implementación de la aplicación, se deja abierta la posibilidad de que el usuario interactúe directamente con uno de estos intérpretes. En esta situación, un usuario con los conocimientos suficientes podría modificar el comportamiento normal de la aplicación aprovechándose de su capacidad para comunicarse con ésta a través de un intérprete de comandos

Con inyección nos referimos pues a la inserción de comandos malintencionados entre los datos que se proporcionan a una aplicación, cuando ésta cuenta con un intérprete que los entiende y está accesible a los usuarios. El fin es modificar el comportamiento habitual de la aplicación y facilitarle al intruso sus propósitos. Por ejemplo, conseguir que la aplicación muestre la lista de usuarios y contraseñas dada de alta, las versiones de los aplicativos, subir archivos de forma ilegítima para tomar el control del servidor, etc.

La lista de intérpretes de los que un atacante podría aprovecharse para llevar a cabo una inyección es bastante amplia. Es equivalente a la lista de tecnologías con la que pueda haber sido desarrollada la aplicación (SQL, HTML, Xpath, Hibernate, interpretes de sistema operativo, etc). La más famosa es la inyección SQL.

8.1.1 Impacto

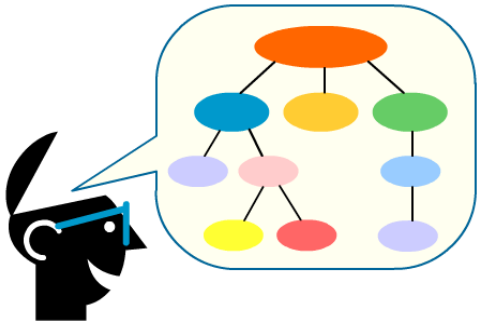
- Severo.
- Violación de la integridad: Pueden permitir crear, borrar o modificar cualquier información arbitraria disponible en la aplicación.
- Violación de la confidencialidad: Los contenidos de una base de datos pueden potencialmente ser leídos.
- Denegación de servicio.
- Acceso a cuentas de usuario.
- Acceso privilegiado al sistema operativo.
- Una falla de inyección puede incluso a llevar a que un intruso se haga con el control del servidor.

8.1.2 Medidas de prevención

- Validación de TODAS las entradas y vías de entrada que se dejan a disposición de los usuarios (empezando por la URL).
- Evitar el uso de interpretes.
- Uso de APIs seguras en el desarrollo.
    - Permiten decodificar y convertir todos los datos ingresados por los usuarios a su forma más simple (escapado) antes de enviarla a un intérprete. El fin es evitar que se ejecuten comandos que pudiese haber inyectado.
    - Uso de variables parametrizadas. Permiten distinguir entre código y datos.
- Realizar siempre una validación “positiva” de las entradas de los usuarios o en base a “listas blancas” donde se contemplen todas las posibles entradas que puede introducir un usuario.
- Seguir el principio de mínimo privilegio en las conexiones con bases de datos para reducir el impacto que pueda provocar que un fallo de inyección sea aprovechado.

8.2 Pérdida de autenticación y gestión de sesiones (A2)

¿Qué son las sesiones web? De una forma sencilla, podría decirse que es "algo" que proporciona la capacidad a un sitio web de "**recordar**" las acciones que ha realizado un usuario. Es como la "memoria" del sitio web. Por ejemplo, gracias a las sesiones web es posible que la página pueda saber:



- Si el usuario se ha identificado.
- Qué elementos ha visitado.
- Los artículos añadidos a una cesta de la compra.
- El tiempo que lleva conectado a la página.
- Etc.

Según esta información, la web va adaptando y modificando sus contenidos o funcionalidades.

El modo en que un sitio web (que usa el protocolo HTTP) implementa esta especie de "memoria" o "recuerdos" es con el uso de los llamados **identificadores de sesión**. Se trata de un **dato único** que es compartido por el navegador que estamos usando (Mozilla, Explorer, Safari, Chrome...) y el servidor web al que nos estamos conectando. El navegador lo guarda en nuestro ordenador en lo que se conoce como una **cookie** y lo incluye en cada petición que va intercambiando con la web en cuestión. Mediante este método una web puede saber si tenemos ya una sesión establecida con ella y recordar quienes somos y todo lo que hemos hecho durante un determinado intervalo de tiempo.

Problema: Los identificadores de sesión son datos que se exponen en la red, en los navegadores, en logs, etc.

Si un usuario malintencionado captura un identificador de sesión válido y realiza peticiones web en las que incluye este identificador, podrá suplantar al usuario al que le pertenece esa sesión. Esto se denomina **secuestro de sesión web**. Todo ello sin necesidad de adivinar su contraseña, pero con el mismo efecto.

8.2.1 Ejemplos

8.2.1.1 Escenario 1

- Una aplicación de reserva de vuelos soporta re-escritura de direcciones URL poniendo los identificadores de sesión en la propia dirección:
  - http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
- Un usuario autenticado en el sitio quiere mostrar la venta a sus amigos.
- Envía por correo electrónico el enlace anterior, sin ser consciente de que está proporcionando su identificador de sesión.

- Cuando sus amigos utilicen el anterior enlace podrán utilizar su sesión (y quizás su tarjeta de crédito) suplantándole.

8.2.1.2 Escenario 2

- El correo electrónico corporativo no establece correctamente los tiempos de desconexión en la aplicación.
- Un usuario utiliza un ordenador público para acceder al correo.
- Cuando termina, en lugar de utilizar la función de "Cerrar sesión", cierra la pestaña del navegador y se marcha.
- Un atacante utiliza el mismo navegador al cabo de una hora, y ese navegador todavía se encuentra autenticado.

8.2.2 Impacto

- Suplantación de identidad.
- Robo de credenciales.
- Cuentas de usuario comprometidas.
- Acceso a información sensible.
- Escalado de privilegios.

8.2.3 Medidas de prevención

- Verificar la arquitectura de la aplicación
  - El sistema de autenticación debe ser:
    - Simple
    - Centralizada
    - Estandarizada
  - Utilizar el gestor de sesiones estándar. No inventar uno propio.
  - Estar seguro que tanto las credenciales como las sesiones de usuario están cifradas durante toda la comunicación.
  - No enviar identificadores de sesión en la URL.

- Verificar la implementación de la web mediante auditorías.
  - Verificar certificados SSL.
  - Examinar todas las funciones relacionadas con autenticación de usuarios.
  - Verificar que la función encargada del "cierre de sesión" destruye la sesión.
  - No utilizar solamente análisis automáticos.

- Facilitar a los desarrolladores web un **único** conjunto de controles fuertes de autenticación y gestión de sesiones.

- Disponer de una interfaz de programación simple para los desarrolladores web
  - Considerar **ESAPI Authenticator** y las APIs de usuario como buenos ejemplos a emular, utilizar o sobre los que partir.
- Reunir todos los requisitos de gestión de sesiones y autenticación definidos en el Application Security Verification Standard (ASVS) de OWASP.

8.3 Secuencia de comandos en sitios cruzados - XSS (A3)

XSS son las siglas de Cross-Site-Scripting (traducido, "secuencia de comandos en sitios cruzados"), y hacen referencia a una modalidad de ataques muy conocidos contra sitios web. Los XSS abarcan cualquier tipo de ataque contra sitios web en los que el objetivo es la inyección de secuencias de comandos que se ejecutan en el lado del usuario que visita el sitio web atacado, en vez de en el propio servidor.

En general, este tipo de ataques puede ser realizado contra cualquier aplicación que tenga por objetivo final presentar información en un navegador web.

Se aprovecha de la incorrecta validación de los datos que los usuarios pueden llegar a ingresar en la aplicación, permitiendo la inserción de código malicioso. Necesitan por tanto un punto de entrada de datos donde poder inyectar código malicioso:

- Formularios
- Mensajes en foros
- Libro de visitas
- Buscadores
- Variables en la URL
- Correo web
- etc.

¿En qué se diferencia entonces de un ataque de inyección como lo comentado en el punto 7.1?

El XSS es una forma especial de ataque de inyección contra la validación de los parámetros de entrada. La diferencia principal entre un ataque XSS y algún otro ataque de inyección estándar (SQL por ejemplo) es que **las víctimas en este caso suelen ser los usuarios de la aplicación**, más que la propia aplicación.

Los ataques XSS **son del lado del cliente**. Están pensados para que impacten directamente contra el usuario que visita el sitio web atacado y para ello usan comandos usando lenguajes que se interpretan en el navegador (principalmente **HTML** y **Javascript**).

8.3.1 Ejemplo

8.3.1.1 Escenario 1

- La página web de un periódico deportivo famoso presenta un fallo que permite la inserción de código Javascript.
- Un atacante detecta el fallo e inserta un código Javascript que captura el identificador de sesión de cualquier usuario que visite la página y lo manda a un sitio propiedad del atacante.
- Una vez obtenido el identificador de sesión el atacante puede acceder a la cuenta de los usuarios del sitio web atacado.

8.3.1.2 Escenario 2

- Un atacante detecta un fallo en un foro muy activo que le permite insertar dentro de un comentario un código Javascript malicioso.
- Cuando un usuario visualiza el foro se ejecuta este código en su navegador y provoca que de forma silenciosa se conecte con un servidor del atacante.
- Desde este servidor el atacante es capaz de obligar al usuario hacer casi cualquier cosa a través del navegador:
  - Habilitar la webcam.
  - Navegar por sitios a merced del atacante.
  - Hacer acciones de todo tipo.
  - Enviar correo fraudulentos a sus contactos.
  - Etc.

8.3.2 Impacto

- Navegación dirigida: Pueden obligar al usuario a que acceda a sitios web o realice peticiones concretas sin que éste se de cuenta.
- Phishing: Pueden conseguir presentar al usuario afectado páginas web falsas que suplantan a servicios existentes (Gmail, portal de un banco, etc) para que ingrese datos susceptibles de ser robados.
- Robo de credenciales y sesiones
- Ejecución de acciones automáticas
- Impacto mediático

8.3.3 Medidas de prevención

Podemos plantear las medidas de prevención desde dos frentes diferentes:

- Diseño de la aplicación
  - Validación de TODAS las entradas y vías de entrada que se dejan a disposición de los usuarios (empezando por la URL).
  - Verificar que el tipo de datos y la longitud de cada campo se corresponda con lo esperado.
  - Filtrar caracteres especiales que puedan resultar peligrosos.
  - Filtrar determinados comandos.
  - En general en los ataques XSS son usadas etiquetas como SCRIPT, OBJECT, APPLET, EMBED y FORM, por lo que se deberían filtrar.
  - Envío de mensajes de alerta al usuario cuando se detecte puede estar siendo víctima de la ejecución de código extraño en su navegador.

- Navegadores
  - Uso de las versiones más recientes de navegadores web.
  - A partir de Internet Explorer 8 se incluye un filtro Anti-XSS
    - Aparición de alerta cuando se intenta acceder a una página web que ha sido víctima de un ataque XSS.

8.4 Referencia directa de objetos insegura (A4)

Este punto se centra en una fallo de seguridad que recae sobre AUTORIZACIÓN. Con autorización nos referimos a los permisos con los que cuentan los usuarios en función del tipo de perfil que tienen. Es decir, qué pueden hacer y qué no, a donde pueden acceder y a donde no, etc. No confundir con autenticación:

- Autenticación: "Sabemos quién eres".
- Autorización: Siendo quién eres, "¿Qué puede y qué no puedes hacer?"

La referencia directa insegura a objetos hace referencia a la posibilidad que tiene un usuario de acceder a objetos a los que no debería.

¿Y qué es un objeto? Consideraremos objetos a los elementos internos de la aplicación, es decir, ficheros, directorios y registros de la base de datos que son referenciados a través de parámetros en la URL, o en formularios. Por esto último se dice que se realiza una **referencia directa** de ellos, porque según la información que contengan estos parámetros, se accederá a uno u otro directamente.

Veámoslo con un ejemplo:

Imaginemos que accedemos a nuestro banco y para acceder a los últimos movimientos de nuestra cuenta pinchamos sobre un enlace que carga la siguiente URL:

- https://www.onlinebank/cuentas.php?cuenta=21010005130002713345

- Usuario del ejemplo: 13579A
- Cuenta del ejemplo: 21010005130002713345

Con esto se cargarían los movimientos de nuestra cuenta propiedad del usuario indicado. Esto se traduce en que se accede a un objeto (por ejemplo, un registro en la base de datos) referenciado por el número de cuenta. Pero ¿y si meto otro número de cuenta que no sea mía? En principio, si todo se hubiera hecho correctamente, debería mostrar un error al estar introduciendo una cuenta ajena, pero si somos vulnerable a la referencia directa de objetos insegura nos mostrará los datos de esta cuenta sin ser nuestra, accediendo de forma directa e insegura al objeto al que referencia el número de cuenta (ajeno, en este caso).

8.4.1 Impacto

- Evasión de permisos.
- Acceso a recursos no permitidos (por ejemplo, de otros usuarios).
- Se puede comprometer toda la información que puede ser accedida en base a los parámetros vulnerables.

8.4.2 Medidas de prevención

- Eliminar las referencias directas. Usar referencias indirectas, por ejemplo, mediante valores aleatorios.
  - ESAPI es una librería que proporciona soporte para mapeos numéricos y aleatorios de objetos.
- Validación de referencias directas.
  - Verificar siempre los permisos del usuario para acceder al objeto solicitado.
  - Verificar el modo de acceso (lectura o escritura).

8.5 Configuración de seguridad defectuosa (A5)

La seguridad de una aplicación depende de la fortaleza y la calidad de los cimientos sobre los que está construida. Este apartado se centra en los riesgos de seguridad que recaen sobre todos aquellos elementos relacionados con el despliegue de una aplicación, así como su entorno. Sistema operativo, permisos, configuraciones, actualizaciones de componentes, cuentas de usuario, manejo de errores, y un largo etcétera de cosas relacionadas con la configuración a la hora de hacer pública una aplicación.

En este punto cobran especial importancia los administradores, los planes de despliegue de aplicaciones, los procesos de instalación y configuración, la planificación y ejecución de actualizaciones, la arquitectura escogida, y una revisión periódica de auditorías y comprobaciones del sistema.

Algunas preguntas que podríamos plantearnos en este sentido:

- ¿Ha fortalecido la seguridad en todos los niveles de la pila de la aplicación?
- ¿Tiene implementados procesos que permitan mantener actualizado el software de TODA su organización?.
- ¿Todo lo innecesario ha sido deshabilitado, eliminado o desinstalado? p.e. puertos, servicios, páginas, cuentas de usuario, privilegios.
- ¿Ha cambiado, o deshabilitado, las contraseñas de las cuentas predeterminadas?
- ¿Ha configurado el sistema de manejo de errores para prevenir que se acceda de forma no autorizada a los mensajes de error?
- ¿Se han comprendido y configurado de forma adecuada las características de seguridad de las bibliotecas y ambientes de desarrollo?
- ¿Está todo documentado?

Respuestas:

- No: MAL!
- Si: BIEN! Pero será necesario un proceso concertado y repetible para desarrollar y mantener una correcta configuración de seguridad de la aplicación.

8.5.1 Impacto

- Los defectos frecuentemente permiten a los atacantes obtener acceso no autorizado a datos o funcionalidades privilegiadas del sistema.
- Estos defectos se traducen como un riesgo para todo el sistema.
- Los datos pueden ser robados o modificados.
- Los costes de recuperación pueden ser altos.

8.5.2 Medidas de prevención

Lo primero que podemos pensar en este punto es que la lista de posibles errores relacionados con el despliegue de la aplicación (así como los riesgos asociados) a prevenir es prácticamente interminable. Por ellos creemos que lo más oportuno es proporcionar una serie de puntos claves a tener en cuenta para evitar configuraciones defectuosas:

- **Arquitectura**: Deberá ser robusta y contar con una buena separación entre los componentes y su seguridad.
- **Permisos**: Éstos deberán ser los justos para que cada cosa pueda hacer aquello para lo que está pensado. Habrá que diferenciar los permisos de la aplicación y los del servidor, y controlar:
  - Permisos de acceso, de lectura y escritura en sistemas de fichero
  - Permisos de acceso y modificación a la BBDD
  - Permisos de utilización de la aplicación y sus partes.

- Puertos de acceso al servidor y a los servicios levantados.

- **Actualizaciones**: Contar con procedimientos de actualización y aplicación de parches, ya sea para la aplicación, como sistemas y plataformas donde está desplegada. Se deberá de contar con un entorno de pruebas donde analizar los efectos de estos cambios antes de pasarlos a producción.
- **Gestión de errores**: Es muy importante tener control y conocimiento sobre todos lo errores que le puedan aparecer a un usuario y no proporcionarles información de los sistemas en estos errores, ya que podría usar esta información para preparar un ataque. Con esto no queremos decir que no se deba informar al usuarios de los errores, sino que éstos deben estar previamente tratados y procesados para que no den más información de la necesaria.
- **Revisiones periódicas** del despliegue que ayuden a detectar fallos o actualizaciones y parches por aplicar.
- **Documentación**: Todos los detalles del despliegue deben estar documentados correctamente.

8.6 Exposición de información sensible (A6)

Algo que en la actualidad puede sonar realmente raro que ocurra, OWASP lo ubica dentro de los 10 riesgos de seguridad más frecuentes en las aplicaciones web.

Los sistemas computacionales guardan hoy en día infinidad de datos. Algunos de ellos irrelevantes, pero otros de vital importancia o de gran sensibilidad. Entre estos últimos podrían mencionarse credenciales de acceso, números de tarjetas de crédito, datos médicos, etc... Toda esta información sensible, aunque inicialmente no esté al alcance de cualquier persona, por unas u otras circunstancias, pueden caer en manos no adecuadas. Por ejemplo, tener una intrusión en nuestros sistemas, que uno de nuestros empleados pierda un portátil, que se extravíe un disco duro con una copia de seguridad de nuestro sistema o cualquier otro tipo de acontecimiento o acción que provoque que está información sensible caiga en manos no autorizadas. Y, ¿qué pasa en esta situación? Todos nuestros datos sensibles estarían al alcance de cualquiera. Esto no deberíamos permitirlo.

¿Como podemos evitarlo? La respuesta es cifrando esta información sensible para que, cuando caiga en manos no autorizadas, resulte totalmente inútil y no se pueda sacar partido de ella. Para esto se inventaron los métodos de cifrado, a través de los cuales podemos evitar que la información sensible sea leída aunque caiga en malas manos.

A la pregunta de **qué deberíamos cifrar**, la respuesta más fácil de recordar y de poner en práctica, es: Toda aquella información que no le darías nunca a un desconocido.

Es muy frecuente encontrar aplicaciones web que no se preocupan por proteger la información sensible, ya sea de los usuarios como propia de la aplicación.

Así pues, este riesgo ocurre cuando:

- No se identifican todos los datos sensibles.
- No se identifican todos los lugares donde estos datos son almacenados.
  - Base de datos, ficheros, carpetas, archivos de log, copias de seguridad, etc.
- No se protege esta información en todas sus ubicaciones.
- No se protege esta información en todos sus estados:
  - En uso.
  - Almacenada.
  - En tránsito.

8.6.1 Impacto

- Atacantes acceden o modifican información privada o confidencial.
- Atacantes extraen información que podrían usar en otros ataques.
- Mala imagen para la compañía, clientes insatisfechos, y perdida de confianza.
- Gastos para corregir el incidente, tales como análisis forense, envío cartas de disculpas, reemisión de tarjetas de crédito, etc.

8.6.2 Medidas de prevención

- Lo primero es identificar son los datos que son suficientemente sensibles y requieren protección adicional.
  - Contraseñas
  - Tarjetas de crédito
  - Datos médicos
  - Información personal
  - etc.
- Identificar por donde se mueven.
- Identificar donde se almacenan.
- Cifrarlos en todos sus estados con algoritmos de cifrado seguro (AES, RSA, SHA-256).
  - No usar algoritmos débiles como MD5 o SHA1.
  - Verificar la implementación de cifrado.
- Verificar que tienen acceso a los datos descifrados sólo los usuarios autorizados.
- No almacenar datos sensibles innecesariamente.

8.7 Falta de controles de acceso (A7)

Al igual que el riesgo A4 ("Referencia directa a objetos de forma insegura"), éste se trata de otro riesgo que recae sobre una mala implementación de la AUTORIZACIÓN.

¿Ha probado en alguna ocasión a poner en el navegador una dirección de una página a la que en principio no se debe poder acceder, para ver si así podía acceder directamente?

Por ejemplo. Imaginemos que navegamos por el sitio www.bancaonline.com/user/getAccounts, y probamos a acceder a www.bancaonline.com/admin/getAccounts. Si tras esto accedemos al perfil del administrador es que estamos ante un fallo de este tipo.

En este caso, más que un fallo en la implementación o la programación del sitio web, se trata de un fallo en la configuración o incluso en la administración del sitio. La mayoría de las páginas web cuentan con 2 espacios, uno visible al público, y otro privado. Además, dentro de la parte privada suelen existir diferentes niveles de acceso dependiendo del usuario. Zonas sólo para el administrador, otras para determinado grupo de usuarios (por ejemplo, trabajadores), otras para usuarios con privilegios mínimos, etc. Si estos niveles de acceso no están correctamente gestionados puede ocurrir que un usuario con pocos privilegios pueda acceder a una parte de la web que no debiera simplemente escribiendo la dirección de acceso a ella en la barra del navegador.

Un error que da lugar a esta situación es mostrar a los usuarios a qué es lo que pueden acceder, y a lo que no se le oculta simplemente. Esto se llama un control de acceso en la capa de presentación y puede ser fácilmente evadido con sólo modificar la URL para acceder a esas partes que "no se ven".

8.7.1 Impacto

- Datos que deberían sólo mostrarse a determinados usuarios quedan expuestos a personas que no deberían tener acceso a ellos.
- Realización de acciones privilegiadas al alcance de cualquiera.
- Acceso a cuentas e información de otros usuarios.

8.7.2 Medidas de prevención

- Autenticación y autorización basada en roles.
  - Minimizan el esfuerzo necesario para mantener las políticas.
- La gestión de políticas debe ser fácil de actualizar y de auditar.
  - No implementar directamente en el código de la aplicación.
- Negar el acceso a todas las páginas por defecto. Habilitarlo de forma explícita a aquellos usuarios. o roles de usuarios que lo requieran.
- Los privilegios de los usuarios deben ser los mínimos requeridos para realizar sus funciones.

- Revisar cada URL de la aplicación. No dejar cabos sueltos.

8.8 Falsificación de peticiones en sitios cruzados - CSRF (A8)

En este punto hablamos de un conocido ataque. Se trata del "Cross Site Request Forgery" o CSFR, traducido al español, "Falsificación de peticiones en sitios cruzados.

El objetivo es aprovecharse de la confianza que delega un sitio web en los usuarios. Veámoslo con un ejemplo.

Un sitio web puede proporcionar al usuario la capacidad de realizar ciertas acciones. Imaginemos el caso de una red social. Nuestro usuario puede publicar comentarios, hacer modificaciones en el perfil, mandar mensajes a otros usuarios, publicar contenidos de todo tipo, etc. Si tuviésemos un usuario con mayores privilegios, por ejemplo un usuario administrador, podríamos incluso hacer más cosas, como crear nuevos usuarios, borrar usuarios existentes, hacer modificaciones en la página, etc.

En principio, cuando un sitio web nos otorga la capacidad de realizar estas acciones confía en que cuando reciba la orden por parte de nuestro usuario de realizar una u otra acción, somos nosotros quienes de forma consciente estamos pidiendo que esa acción se realice. Es esta confianza que deposita la página web sobre nuestras acciones la que puede llevarnos ser víctimas de un ataque de este tipo.

Supongamos que para cerrar la sesión de nuestro usuario en esta página web pulsamos sobre un botón que pone "SALIR", y al pulsar sobre éste se realiza la siguiente petición web:

http://sitioWeb.es/session.html?action=logout

Ahora imaginemos que seguimos dentro del sitio web con nuestra sesión de usuario abierta y nos llega un mensaje privado de un usuario diciendo lo siguiente:

"Hola amigo. Te paso el enlace de la foto que nos hicimos anoche: FOTO.jpg"

Pulsamos sobre este enlace y accedemos a una página web donde aparece una imagen que no reconocemos. Entonces la cerramos como si no hubiera pasado nada, y volvemos a la página donde estábamos. Sin embargo, nos encontramos con que nos hemos salido de la sesión. ¿Cómo ha sido posible esto? Lo que ha ocurrido es que al acceder al sitio web con la imagen que nos han mandado, por detrás se ha cargado la URL **http://sitioWeb.es/session.html?action=logout** sin que nos diésemos cuenta, lo cual ha ordenado al sitio web inicial que cerremos la sesión de nuestro usuario.

Como hemos podido observar, el sitio web no ha sido capaz de determinar si esta orden la hemos enviado nosotros conscientemente o no. El sitio web confía en que si le llega la orden desde nuestro usuario, somos nosotros quienes la hemos mandado, y la lleva a cabo.

Quizás este ejemplo no muestra un caso grave, pero imaginemos que en vez de una orden de cierre de sesión se trata de la petición de una transferencia bancaria, dar de alta o baja a un usuario, enviar un mensaje a un cierto contacto (envío de spam), etc.

8.8.1 Impacto

- El impacto es directamente proporcional a las posibles acciones que pueda generar un usuario en una aplicación en función de sus privilegios.
  - Iniciar transacciones de forma involuntaria.
  - Acceso a datos sensibles.
  - Crear, modificar, eliminar cuentas de usuarios.
  - Modificar datos personales
  - Publicar información en nombre de otro.
  - Comprar productos.
  - Etc.
- ERROR: Pensar que el usuario es el responsable.

8.8.2 Medidas de prevención

- No usar para las acciones de peticiones web fácilmente reconocibles y replicables.
  - Uso de tokens fuertes y aleatorios.
  - No predecibles.
  - Ocultos.
- Requerir reautenticaciones en funciones críticas.
- Uso de captchas.

A continuación se proporcionan proyectos de OWASP dedicados a proporcionar herramientas y recomendaciones para evitar este tipo de fallos:

- OWASP CSRFTester Project
- OWASP CSRFGuard Project
- OWASP CSRF Prevention Cheat Sheet

8.9 Uso de componentes vulnerables (A9)

En la edición del 2010 del Top 10 de OWASP este punto estaba incluido dentro del actual punto A5: "Configuración de seguridad defectuosa". Sin embargo, en la edición de 2013 se ha decidido tratar este riesgo como un caso a parte debido a que es muy común. Por este motivo es posible que lo comentado en este punto nos suene ya.

En este apartado, como bien se indica en el nombre, nos centramos en el riesgo que supone usar en nuestra aplicación componentes vulnerables. Nos referimos a componentes para los que se han publicado fallos de seguridad, e incluso parches para corregirlos o actualizaciones.

Es extremadamente frecuente encontrarse con páginas web que hacen uso de versiones de programas obsoletos. Esto supone un riesgo enorme para nuestra aplicación, e incluso para nuestra infraestructura.

Actualizar el software de nuestras aplicaciones es fundamental para evitar que la seguridad de nuestra aplicación, la de nuestros usuarios y la de nuestra organización se vea comprometida.

En ocasiones algo tan trivial como actualizar puede resultar bastante complicado, sobre todo en los casos en que el software usado sea a medida. Pensemos en el siguiente escenario.

- Se publica una nueva vulnerabilidad crítica en Java cuya explotación permite tomar el control del equipo del usuario.
- El fabricante de Java publica una actualización de Java que corrige este problema.
- Los usuarios de nuestra organización hacen uso de un programa corporativo que requiere el uso de una versión antigua de Java.
- No es posible actualizar.

Un problema típico en este escenario tiene su origen en el desarrollo y uso de aplicaciones corporativas. Muchas aplicaciones corporativas hacen uso en sus desarrollos de software de terceros (librerías de Java, bases de datos MySQL, etc). Los desarrolladores de software deben ser conscientes de este tipo de situaciones a la hora de requerir el uso de software de terceros para sus desarrollos y contar con planes de actualización. En caso de que se publique un fallo de seguridad en un software de tercero usado en una aplicación corporativa se deberá contar con los medios apropiados para poder actualizar y corregir el fallo.

Si este tipo de fallos de seguridad no son gestionados debidamente se traslada el problema a los usuarios, dando lugar a una proliferación del fallo por toda la organización.

8.9.1 Impacto

- El uso de componentes vulnerables conlleva a todo tipo de amenazas y ataques.

8.9.2 Medidas de prevención

- **Actualizaciones**: Contar con procedimientos de actualización y aplicación de parches, ya sea para la aplicación, como sistemas y plataformas donde está desplegada. Se deberá de contar con un entorno de pruebas donde analizar los efectos de estos cambios antes de pasarlos a producción.

- **Revisiones periódicas** del despliegue que ayuden a detectar componentes vulnerables, y actualizaciones y parches por aplicar.

- Los proyectos de software deben tener procedimientos para:
  - Identificar los componentes y dependencias que necesitan y sus versiones.
  - Estar al día de la seguridad de estos componentes .
  - Bases de datos públicas .
  - Listas de correo del proyecto .
  - Listas de correo de seguridad .
  - Desarrollar correctivos para adaptar la aplicación a las nuevas versiones de los componentes.
  - Establecer políticas de seguridad que rijan el uso de componentes .
  - Realizar pruebas de seguridad.

8.10 Redirecciones y reenvíos no validados (A10)

Para ir cerrando este listado haremos una breve parada para comentar el caso de las redirecciones. No es difícil encontrar una aplicación que en determinadas ocasiones, por necesidad de la propia aplicación, se realice una redirección legítima hacia otra web.

El problema surge cuando esas redirecciones pueden ser alteradas por un usuario con malas intenciones para llevarnos a un sitio web a su antojo donde, por ejemplo, haya suplantado a la página original con el objetivo de engañarnos y robar nuestras credenciales de usuario.

8.11 Impacto

- Redirección de usuarios a sitios maliciosos .
- Evadir controles de acceso .
- Acceso a partes de administración de la aplicación.

8.12 Medidas de prevención

- Evitar el uso de reenvíos y redirecciones.
- Si se utilizan, no involucrar parámetros manipulables por el usuario para definir el destino.
- Si los parámetros de destino no pueden evitarse, asegúrese de que el valor facilitado es validado y autorizado.
  - Se recomienda que el valor de cualquier parámetro de destino sea un valor mapeado no interpretable a simple vista, en lugar de la dirección, o parte de la dirección, de la URL
  - En el código del servidor traducir dicho valor a la dirección URL de destino.
- Uso de OWASP Enterprise Security API (ESAPI).

9 RESUMEN

A continuación se muestra un resumen de todos los riesgos vistos en este documento. Se muestra en una tabla con los valores del análisis del riesgo proporcionados por OWASP.

RISK	Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
		Exploitability	Prevalence	Detectability	Impact	
A1-Injection		EASY	COMMON	AVERAGE	SEVERE	
A2-Auth'n		AVERAGE	WIDESPREAD	AVERAGE	SEVERE	
A3-XSS		AVERAGE	VERY WIDESPREAD	EASY	MODERATE	
A4-Insecure DOR		EASY	COMMON	EASY	MODERATE	
A5-Config		EASY	COMMON	EASY	MODERATE	
A6-Sens. Data		DIFFICULT	UNCOMMON	AVERAGE	SEVERE	
A7-Function Acc.		EASY	COMMON	AVERAGE	MODERATE	
A8-CSRF		AVERAGE	COMMON	EASY	MODERATE	
A9-Components		AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	
A10-Redirects		AVERAGE	UNCOMMON	EASY	MODERATE	

Este listado cubre bastantes escenarios, pero existen otros riesgos que se deberán considerar y evaluar. Algunos de éstos son:

- Clickjacking .
- Defectos en concurrencia.
- Denegaciones de servicio.
- Expression Languaje Injections.

- Fuga de información.
- Manejo inapropiado de errores.
- Insuficiente protección contra sistemas automáticos (bots).
- Insuficiente registro y responsabilidad.
- Falta de detección y respuesta a intrusiones.
- Ejecución de archivos maliciosos.
- Privacidad de usuarios.

10 CONCLUSIONES

En definitiva, ¿todo esto para qué?

La principal motivación que nos ha llevado a desarrollar este informe es crear conciencia en los equipos responsables las aplicaciones web que se ponen a disposición de los usuarios (desde la dirección, equipo de desarrollo, de explotación, etc), responsables en gran medida de estas cuestiones.

Hoy en día, la seguridad en las aplicaciones no debe ser una opción. Entre los ataques en aumento y presiones de cumplimiento normativo, las organizaciones deben establecer un mecanismo eficaz para asegurar sus aplicaciones. Dado el asombroso número de solicitudes y líneas de código que ya están en producción, muchas organizaciones están luchando para conseguir gestionar el enorme volumen de vulnerabilidades. Se recomienda establecer un programa de seguridad de las aplicaciones para aumentar el conocimiento y mejorar la seguridad en toda su cartera de aplicaciones.

Conseguir un nivel de seguridad de las aplicaciones requiere que diversas partes diferentes de una organización trabajen juntos de manera eficiente, incluidos los departamentos de seguridad y auditoría, desarrollo de software, y gestión ejecutiva y del negocio. Se requiere que la seguridad sea visible, para que todos los participantes puedan ver y entender la postura de la organización de seguridad en aplicaciones. También es necesario centrarse en las actividades y resultados que realmente ayuden a mejorar la seguridad de la empresa mediante la reducción de riesgo en la forma más rentable posible.

Como última anotación, creemos que el mejor consejo es pensar las cosas bien pensadas y no hacerlas a la ligera, y sobre todo, a la hora de diseñas e implementar un sistema, no solo ponerse en la piel del “desarrollador”, sino también en la del “usuario” y como no, en la del “atacante”.

11 GLOSARIO

- **Riesgo:** Probabilidad de que una amenaza provoque o cause un daño.
- **Login:** Proceso mediante el cual se controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas por el usuario
- **API:** Interfaz de programación de aplicaciones (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.
- **Cookie:** Una cookie (o galleta informática) es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.
- **Captcha:** Son las siglas de Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar máquinas y humanos). Se trata de una prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano.
- **Benchmarking:** El benchmarking es un anglicismo que puede definirse como un proceso sistemático y continuo para evaluar comparativamente los productos, servicios y procesos de trabajo en organizaciones. Consiste en tomar "comparadores" (o benchmarks) a aquellos productos, servicios y procesos de trabajo que pertenezcan a organizaciones que evidencien las mejores prácticas sobre el área de interés, con el propósito de transferir el conocimiento de las mejores prácticas y su aplicación.

12 REFERENCIAS

- OWASP
- OWASP Top 10 – 2013 - RC1