



**OWASP**

The Open Web Application Security Project  
<http://www.owasp.org>

---

---

# **Las diez vulnerabilidades de seguridad más críticas en aplicaciones Web**

## **Actualización de 2004**

**27 de enero de 2004**

Copyright © 2004. The Open Web Application Security Project (OWASP). Derechos Reservados.

Permiso otorgado para copiar, distribuir y/o modificar este documento en caso de que este anuncio de derechos reservados y la atribución a OWASP sea mantenida.



## Comentarios

---

*"Con vulnerabilidades anunciadas casi semanalmente, muchos negocios pueden sentirse agobiados tratando de mantenerse actualizados. Sin embargo, existe ayuda en la forma de listas concienzudas de vulnerabilidades y defensas.*

*"El Proyecto Abierto de Seguridad de Aplicación Web (OWASP por sus siglas en inglés) ha producido una lista similar de las 10 principales vulnerabilidades de aplicaciones Web y bases de datos más críticas, así como la forma más efectiva de resolverlas. Las vulnerabilidades de aplicación son a veces desatendidas, pero son tan importantes de resolver como los problemas de redes. Si cada compañía eliminara estas vulnerabilidades comunes, su trabajo no estaría del todo hecho, pero ellos y la Internet, serían significativamente más seguros."*

- **J. Howard Beales, III, Director de la Federal Trade Commission's Bureau of Consumer Protection, antes la Information Technology Association of America's Internet Policy Committee, Viernes, Diciembre 12, 2003**

---

*"Configuraciones mal hechas, falta de atención y software con errores pueden crear desastres en la Internet. Una de las áreas primarias de vulnerabilidad es a través de conexiones WWW. Por diseño, los servicios WWW pretenden ser abiertos y receptivos y usualmente actúan como una interfaz de recursos de valor. De este modo, es crítico que estos servicios sean seguros. Pero con miles de vulnerabilidades potenciales puede ser muy abrumador decidir en dónde iniciar la aplicación de medidas defensivas. Las "Top Ten" de OWASP proveen una vista concienzuda de las vulnerabilidades más significativas y probables en aplicaciones personalizadas WWW. Las organizaciones deben usarla para enfocar sus esfuerzos, con la confianza de que están atendiendo esas áreas que tendrán el mayor impacto al asegurar sus aplicaciones."*

- **Eugene H. Spafford, Profesor de Ciencias Computacionales, Purdue University y Director ejecutivo de la Purdue University Center for Education and Research in Information Assurance and Security (CERIAS)**

---

*"Esta lista de 'Las diez más requeridas' es apenas la punta de un enorme iceberg. La realidad de fondo es vergonzosa: la mayoría de los sistemas y software de aplicaciones Web están escritos sin conciencia de los principios de seguridad, ingeniería de software, implicaciones operacionales y, ciertamente, de sentido común."*

- **Dr. Peter G. Neumann, Principal Scientist, SRI International Computer Science Lab, Moderador de la ACM Risks Forum, Autor de "Computer-Related Risks"**

---

*"Esta lista es un desarrollo importante tanto para los consumidores como los vendedores. Esto educará a los vendedores para evitar los mismos errores que han sido repetidos infinidad de veces en otras aplicaciones Web. También brinda a los consumidores una vía para solicitar a los vendedores que sigan un conjunto mínimo de expectativas para la seguridad de aplicaciones Web, e igualmente importante, identificar cuáles vendedores no están cumpliendo con las expectativas"*

- **Steven M. Christey, Ingeniero de Seguridad de Información Principal y Editor de CVE, Mitre**

---

*"Las "Top Ten" de OWASP brillan como un reflector sobre uno de los riesgos más serios, y comúnmente ignorado, que enfrentan los gobiernos y organizaciones comerciales. La principal causa de estos riesgos no es software con fallas, sino los procesos de desarrollo de software que le ponen poca o ninguna atención a la seguridad. El primer paso efectivo para la creación de una cultura consciente de la seguridad en sus organizaciones es adoptar inmediatamente las "Top Ten" como un estándar mínimo de seguridad de aplicaciones Web."*

- **Jeffrey R. Williams, Aspect Security CEO y Líder del Proyecto OWASP Top Ten**
-



*"A pesar de lo que algunas empresas de tecnología nos quieren hacer creer, no existe un arma infalible para garantizar la seguridad de la Web. Resolver este complejo y desafiante problema implica tener excelente personal, gran conocimiento, excelente educación, excelentes procesos de negocios y excelente uso de tecnología. Las "Top Ten" de OWASP crea una lista aceptablemente bien organizada y bien pensada donde uno puede empezar a comprender su postura de seguridad (o la de tus proveedores de servicios) y, en consecuencia, planear el uso de sus recursos de valor."*

- **Mark Curphey, Fundador de OWASP y Director de Consultoría de Seguridad de Aplicaciones, Foundstone Strategic Security.**
- 

*"Desde páginas Web hasta la captación de datos, casi todas las organizaciones adquieren código de aplicaciones, muchas personas las escriben, y todos las utilizan. Sin embargo, aún después de casi 50 años de experiencia en desarrollo, las aplicaciones siguen teniendo fallas. Peor aún, el mismo tipo de fallas aparece una y otra vez. Este fracaso, en no sólo no aprender de nuestros errores sino tampoco de los de la generación de nuestros padres, crea demasiadas vulnerabilidades para un ataque potencial. No es extraño que los ataques contra aplicaciones estén a la alza.*

*Al compilar esta lista con las diez más importantes fallas de código en aplicaciones, OWASP ha prestado un gran servicio, tanto a los desarrolladores como a los usuarios, al enfocar su atención en las debilidades más comunes y lo que puede hacerse al respecto. Ahora depende de cada organización de desarrollo de software, programadores, y usuarios el aplicar la bien pensada guía presentada aquí."*

- **Dr. Charles P. Pfleeger, CISSP, Master Security Architect, Cable & Wireless, Autor de "Security in Computing"**
- 

*"El nuevo Retorno de Inversión (ROI por sus siglas en inglés) es seguridad. Las empresas tienen que ser capaces de proveer aplicaciones Web y servicios Web seguros para sus aliados comerciales. Éstos requieren tanto tecnología de seguridad como signos tradicionales de seguridad financiera, como por ejemplo, seguros. Sin embargo, esto no es todo; una seguridad deficiente pone en riesgo los datos y aplicaciones de una empresa, por ende, su viabilidad como entidad comercial. Decepcionar a los clientes es una cosa, tratar de enmendar una pérdida o corrupción de sus propios sistemas es otra muy diferente."*

- **Robert A. Parisi, Jr., Senior VP y Chief Underwriting Officer, AIG eBusiness Risk Solutions**
- 

*"Los desarrolladores Web necesitan saber que el grado en que sus aplicaciones de negocios y los datos de clientes estén protegidos de la hostil Internet es directamente determinado por qué tan seguro han escrito su código. La lista de las "Top Ten" de OWASP es una excelente forma de comprender cómo codificar defensivamente y cómo evitar las trampas de seguridad que plagan las aplicaciones Web."*

- **Chris Wysopal, Director de Investigación y Desarrollo, @stake, Inc.**
- 

*"La industria de la salud tiene una necesidad crítica para proveer aplicaciones Web seguras que protejan la privacidad del usuario. Las "Top Ten" de OWASP ayudarán a que las organizaciones de salud evalúen la seguridad de los productos y soluciones de aplicaciones Web. Cualquier organización de salud que utilice aplicaciones que contengan estas fallas puede tener dificultades para cumplir con las regulaciones de la HIPAA."*

- **Lisa Gallagher, Senior VP, Information and Technology Accreditations, URAC**
- 

*"Conforme más y más compañías eligen "offshore outsourcing" para el desarrollo de aplicaciones Web, el reforzamiento de código seguro tiene que saltar al tope de la lista de prioridades ejecutivas. El Proyecto de las "Top Ten" de OWASP expresa fielmente las principales vulnerabilidades de aplicaciones, proveyendo a los ejecutivos de nivel C con una mina de oro de información útil para la configuración de los lineamientos de políticas de seguridad para aplicaciones Web."*

- **Stuart McClure, Presidente y CTO de Foundstone Inc.**
-



*“Las consideraciones en seguridad de aplicaciones son frecuentemente tratadas como un dominio de los especialistas, para ser aplicadas después de que la codificación está hecha y todos los demás requisitos de negocio han sido satisfechos. En Oracle hemos encontrado que la tarea de asegurar aplicaciones es más probable que tenga éxito si esta construida desde el inicio. Los desarrolladores debe tener un conocimiento básico de las vulnerabilidades de seguridad y cómo evitarlas; los directores de liberación deben explícitamente verificar que los requerimientos de seguridad han sido satisfechos antes de que el producto se embarque. La lista de las “Top Ten” de OWASP en un excelente punto de inicio para que las compañías hagan conciencia de seguridad entre sus desarrolladores, y para preparar criterios de seguridad para el despliegue de aplicaciones.”*

■ **John Heimann, Director de Security at Oracle Corp.**

---

*“Mucha gente ha visto las listas que detallan los “grandes riesgos de seguridad en redes”. Estas listas sólo mencionan fallas en programas de terceros, que pueden o no estar corriendo en su red. Aplicando la información en esta lista mantendrá el software que usted desarrollará fuera de estas otras listas.”*

■ **John Viega, Chief Scientist, Secure Software Inc., co-autor de “Building Secure Software”**

---



## Tabla de Contenido

Introducción.....	1
Créditos .....	2
Antecedente .....	3
Novedades .....	4
Lista de las “Top Ten” .....	5
A1 Entrada no validada .....	6
A2 Control de Acceso Interrumpido.....	8
A3 Administración de Autenticación y Sesión Interrumpida .....	11
A4 Fallas de Cross-Site Scripting (XSS) .....	14
A5 Desbordamiento del Búfer .....	17
A6 Fallas de Inyección.....	19
A7 Manejo Inadecuado de Errores.....	21
A8 Almacenamiento Inseguro .....	23
A9 Negación de Servicio .....	25
A10 Administración de Configuración Insegura .....	27
Conclusiones.....	29



## Introducción

El Proyecto Abierto de Seguridad de Aplicaciones Web (OWASP: The Open Web Application Security Project) está dedicado a ayudar a las organizaciones a comprender y mejorar la seguridad de sus aplicaciones Web y servicios Web. Esta lista fue creada con el objetivo de enfocar a las empresas y agencias gubernamentales en las vulnerabilidades más serias. La seguridad de aplicaciones Web ha sido un tema importante, en tanto muchas compañías compiten por hacer accesible contenidos y servicios a través de la Web. Al mismo tiempo, los “piratas informáticos” centran su atención en las debilidades más comunes creadas por los desarrolladores de aplicaciones.

Cuando una organización pone una aplicación Web, invita al mundo a enviar llamadas HTTP (HTTP Requests). Los ataques dentro de estas llamadas navegan a través de los cortafuegos (firewalls), filtros, endurecimiento de plataforma, y sistemas de detección de intrusos sin aviso alguno porque están dentro de llamadas HTTP legales. También, sitios Web “seguros” que utilizan SSL aceptan las llamadas recibidas a través del túnel seguro sin escrutinio alguno. **Esto quiere decir que su código de aplicación Web es parte de su perímetro de seguridad.** Mientras el número, tamaño y complejidad de su aplicación Web se incrementa, así mismo será su exposición del perímetro.

Los puntos de seguridad explicados aquí no son nuevos. En realidad, han sido comprendidos por décadas. Sin embargo, por una variedad de razones, la mayoría de proyectos de desarrollo todavía cometen estas equivocaciones y ponen en peligro no solamente la seguridad de sus clientes, sino también la seguridad de toda la Internet. No existe ninguna solución única para resolver estos problemas. La tecnología de protección y asesoramiento actual está mejorando, pero solamente puede lidiar con un subconjunto limitado de los puntos en el mejor de los casos. Para atender los puntos descritos en este documento, las organizaciones tendrán que cambiar su cultura de desarrollo, entrenar a los desarrolladores, actualizar sus procesos de desarrollo de software y usar la tecnología donde sea apropiado.

**La lista de las “Top Ten” de OWASP es una lista de vulnerabilidades que requiere de atención inmediata.** El código existente para estas vulnerabilidades necesita ser revisado inmediatamente ya que estas fallas están siendo el blanco actual de los atacantes. Los proyectos de desarrollo deben tratar estas vulnerabilidades en sus documentos de requerimiento y diseño, y creación, y probar sus aplicaciones para asegurar que no hayan sido introducidas. Los administradores de proyectos deben incluir tiempo y presupuesto para las actividades de seguridad de aplicaciones, incluyendo entrenamiento para desarrolladores, política de seguridad para desarrollo de aplicaciones, desarrollo y diseño de mecanismos de seguridad, pruebas de penetración y revisión de código fuente.

Animamos a las organizaciones a unirse a la lista creciente de compañías que han **adoptado las “Top Ten” de OWASP como un estándar mínimo** y están comprometidas a producir aplicaciones Web que no contengan estas vulnerabilidades.

Hemos elegido presentar esta lista en un formato similar a la tan exitosa lista de las “Top Twenty” de SANS/FBI con el fin de facilitar su uso y su comprensión. La lista de SANS está enfocada a las fallas de productos particularmente usados en redes e infraestructura. Ya que cada sitio Web es único, las “Top Ten” de OWASP están organizadas alrededor de tipos y categorías especiales de vulnerabilidades que frecuentemente ocurren en aplicaciones Web. Estas categorías están siendo estandarizadas en el proyecto XML de Seguridad de Aplicaciones Web (WAS por sus siglas en inglés) OASIS.

Esta lista representa el conocimiento combinado de los expertos de OWASP cuya experiencia incluye muchos años de trabajo en seguridad de aplicaciones para gobiernos, servicios financieros, empresas farmacéuticas y manufacturación, así como también el desarrollo de herramientas y tecnología. Este documento está diseñado para presentar las vulnerabilidades de aplicaciones Web más serias. Existen muchos libros y pautas que describen estas vulnerabilidades con más detalle y proveen una orientación de cómo eliminarlas. Una de esas pautas es la guía de OWASP disponible en <http://www.owasp.org>.

El documento de las “Top Ten” de OWASP es un documento vivo. Incluye instrucciones y enlaces a información adicional, útil para corregir estos tipos de fallas de seguridad. Actualizamos la lista y las instrucciones conforme se identifican más amenazas críticas y métodos más actuales o adecuados, y agradecemos sus sugerencias durante este proceso. Este es un documento comunitariamente consensuado – su experiencia en combatir atacantes y eliminar estas vulnerabilidades puede ayudar a quienes vengan después de usted. Por favor, envíe sus sugerencias vía correo electrónico a [topten@owasp.org](mailto:topten@owasp.org) con el asunto “OWASP Top Ten Comments” o escribe a la lista para los lectores de habla hispana [owasp-spanish@lists.sourceforge.net](mailto:owasp-spanish@lists.sourceforge.net).



## Créditos

Este documento fue traducido a su versión en español por las siguientes personas (en orden alfabético):

- Juan Carlos Calderon Rojas
- Pedro Del Real López
- Rogelio Miguel Morrell Caballero

Adicionalmente, agradecemos el apoyo como editores a las siguientes personas:

- Alejandra María Ancira Cárdenas
- Fernando Javier Múzquiz Ramírez

OWASP reconoce con agradecimiento la contribución especial de **Aspect Security** por su rol de investigación y preparación de este documento.



<http://www.aspectsecurity.com>



## Antecedente

El desafío de identificar las “principales” vulnerabilidades de aplicaciones Web es una tarea virtualmente imposible. No hay ni siquiera un acuerdo generalizado de qué exactamente está incluido dentro del término “seguridad de aplicaciones Web.” Algunos han argumentado que sólo deberíamos enfocarnos en los puntos de seguridad que afectan a los desarrolladores que escriben código de aplicaciones Web a la medida. Otros abogan por una definición más amplia que abarque la capa de aplicación entera, incluyendo bibliotecas, configuración de servidores y protocolos de la capa de aplicación. Con la esperanza de crear conciencia sobre los riesgos más serios que enfrentan las organizaciones, hemos decido dar una interpretación relativamente amplia al concepto de seguridad de aplicaciones Web, mientras seguimos manteniendo claros los puntos de seguridad de red e infraestructura.

Otro desafío a este esfuerzo es que cada vulnerabilidad específica es única para un sitio Web de una organización en particular. Sería poco práctico detallar vulnerabilidades específicas en las aplicaciones Web de organizaciones particulares, puesto que éstas probablemente serían arregladas con prontitud después del conocimiento general de su existencia. De este modo, hemos elegidos enfocarnos en las principales clases, tipos y categorías de vulnerabilidades de aplicaciones Web.

En la primera versión de este documento decidimos clasificar en categorías significativas un amplio rango de problemas de aplicaciones Web. Estudiamos una variedad de esquemas de clasificación de vulnerabilidades y produjimos un conjunto de categorías. Los factores que caracterizan una buena categoría de vulnerabilidad incluyen si las fallas son estrechamente relacionadas, si pueden ser aplicadas con contramedidas similares y si ocurren frecuentemente en arquitecturas típicas de aplicaciones Web. En esta versión estamos introduciendo un esquema refinado. Éste ha sido desarrollado a partir de nuestro trabajo continuo en el comité técnico de OASIS WAS, y en él describiremos un glosario de asuntos desde donde los investigadores de seguridad puedan describir firmas en formato XML.

Elegir las “Top Ten” de una larga lista de opciones tiene su propio conjunto de dificultades. No hay simplemente ninguna fuente de estadísticas sobre problemas de seguridad de aplicaciones Web. En el futuro, nos gustaría coleccionar estadísticas sobre la frecuencia de ciertas fallas en el código de aplicaciones Web y usar estas métricas para ayudar a priorizar las “Top Ten”. Sin embargo, por un número de razones, este tipo de medición no es probable que ocurra en un futuro cercano.

Reconocemos que no existe ninguna respuesta “correcta” para las categorías de vulnerabilidades que deben estar en las “Top Ten”. Cada organización necesitará pensar en el riesgo para sí misma, basándose en la probabilidad de tener unas de estas fallas y en las consecuencias específicas para su empresa. Por lo pronto, hemos propuesto esta lista como un conjunto de problemas que representan una significativa cantidad de riesgos para una amplia gama de organizaciones. Las “Top Ten” en sí no tienen ningún orden particular, ya que sería casi imposible determinar cuál de ellas representan el mayor riesgo agregado.

El proyecto de la “Top Ten” de OWASP es un esfuerzo, en proceso de desarrollo, para divulgar a un público más extenso la información sobre vulnerabilidades de aplicaciones Web clave. Esperamos actualizar este documento anualmente basados en discusiones de las listas de correo de OWASP y por medio de las sugerencias a [top10@owasp.org](mailto:top10@owasp.org).



## Novedades

OWASP ha recorrido un largo camino desde la última vez que las “Ten Top” fueron lanzadas en enero de 2003. Esta actualización incorpora todos los temas y puntos de vistas, opiniones y debates de la comunidad de OWASP a lo largo de los últimos 12 meses. En general, han habido mejoras pequeñas a todas las partes de las “Top Ten” y sólo algunos cambios grandes:

- **Alineación al WAS-XML**– Uno de los nuevos proyectos iniciados en el 2003 es el Comité Técnico de Seguridad de Aplicaciones Web (WAS TC, por sus siglas en inglés) en [OASIS](#). El propósito del WAS TC es producir un esquema de clasificación para vulnerabilidades de seguridad Web, un modelo que provea una guía inicial para amenazas, impacto y por lo tanto calificación de riesgos, así como también un esquema XML para describir las condiciones de seguridad Web que puedan ser usadas tanto para herramientas de asesoramiento como de protección. El proyecto de las “Top Ten” de OWASP ha usado el WAS TC como una referencia para re perfilar las “Top Ten” y proveer un alcance estandarizado para la clasificación de las vulnerabilidades de seguridad de aplicaciones Web. El glosario WAS define el lenguaje estándar para discutir la seguridad de aplicaciones Web y nosotros hemos adoptado este vocabulario aquí.
- **Adición de Negación de Servicio** – El único nivel de categoría principal que cambió fue la adición de la categoría A9 Negación de Servicio a la lista. Nuestra investigación mostró que una gran gama de organizaciones es susceptible a este tipo de ataque. Basados en la probabilidad de un ataque de negación de servicio y en las consecuencias si un ataque sucede, hemos determinado que se requiere su inclusión en las “Top Ten”. Para acomodar esta nueva entrada hemos combinado la categoría del año pasado A9 Fallas de Administración Remota con la categoría A2 Control de Acceso Interrumpido, ya que ésta es un caso especial de aquella categoría. Nosotros creemos que es apropiado, ya que los tipos de fallas en A2 son típicamente las mismas que las de A9 y requieren del mismo tipo de solución.

La tabla abajo muestra la relación entre las nuevas “Top Ten”, las “Top Ten” del año pasado y el glosario de WAS TC.

Las nuevas “Top Ten” 2004	Las “Top Ten” 2003	Nuevo glosario WAS
A1 Entrada no validada	A1 Parámetros Inválidados	Validación de Entradas
A2 Control de Acceso Interrumpido	A2 Control de Acceso Interrumpido (A9 Fallas de Administración Remota)	Control de Acceso
A3 Administración de Autenticación y Sesión Interrumpida	A3 Administración de Cuentas y Sesión Interrumpida	Administración de Autenticación y Sesión
A4 Fallas de Cross Site Scripting (XSS)	A4 Fallas de Cross Site Scripting (XSS)	Validación de Entradas ->Cross Site Scripting
A5 Desbordamiento de Búfer	A5 Desbordamiento de Búfer	Desbordamiento de Búfer
A6 Fallas de Inyección	A6 Fallas de Inyección de Comandos	Validación de Entradas ->Inyección
A7 Manejo Inadecuado de Errores	A7 Problemas de Manejo de Errores	Manejo de Errores
A8 Almacenamiento Inseguro	A8 Uso Inseguro de Criptografía	Protección de Datos
A9 Negación de Servicio	No aplicable	Disponibilidad
A10 Administración de Configuración Insegura	A10 Configuración Indebida de Servidor Web y de Aplicación	Administración de Configuración de Aplicación Administración de Configuración de Infraestructura



## Lista de las “Top Ten”

El siguiente es un breve resumen de las vulnerabilidades más significantes de la seguridad de aplicaciones Web.

Vulnerabilidades principales en aplicaciones Web		
A1	<b>Entrada no validada</b>	La información de llamadas Web no es validada antes de ser usadas por la aplicación Web. Los agresores pueden usar estas fallas para atacar los componentes internos a través de la aplicación Web.
A2	<b>Control de Acceso Interrumpido</b>	Las restricciones de aquello que tienen permitido hacer los usuarios autenticados no se cumplen correctamente. Los agresores pueden explotar estas fallas para acceder a otras cuentas de usuarios, ver archivos sensitivos o usar funciones no autorizadas.
A3	<b>Administración de Autenticación y Sesión Interrumpida</b>	Las credenciales de la cuenta y los tokens de sesiones no están propiamente protegidos. Los agresores que pueden comprometer las contraseñas, claves, cookies de sesiones u otro token, pueden vencer las restricciones de autenticación y asumir la identidad de otros usuarios.
A4	<b>Fallas de Cross Site Scripting (XSS)</b>	La aplicación Web puede ser usada como un mecanismo para transportar un ataque al navegador del usuario final. Un ataque exitoso puede comprometer el token de sesión del usuario final, atacar la maquina local o enmascarar contenido para engañar al usuario.
A5	<b>Desbordamiento del Búfer</b>	Los componentes de aplicaciones Web en ciertos lenguajes que no validan adecuadamente las entradas de datos pueden ser derribados y, en algunos casos, usados para tomar control de un proceso. Estos componentes pueden incluir CGI, bibliotecas, rutinas y componentes del servidor de aplicación Web.
A6	<b>Fallas de Inyección</b>	La aplicación Web puede pasar parámetros cuando accede a sistemas externos o al sistema operativo local. Si un agresor puede incrustar comandos maliciosos en estos parámetros, el sistema externo puede ejecutar estos comandos por parte de la aplicación Web.
A7	<b>Manejo Inadecuado de Errores</b>	Condiciones de error que ocurren durante la operación normal que no son manejadas adecuadamente. Si un agresor puede causar que ocurran errores que la aplicación Web no maneja, éste puede obtener información detallada del sistema, denegar servicios, causar que mecanismos de seguridad fallen o tumbar el servidor.
A8	<b>Almacenamiento Inseguro</b>	Las aplicaciones Web frecuentemente utilizan funciones de criptografía para proteger información y credenciales. Estas funciones y el código que integran a ellas han sido difíciles de codificar adecuadamente, lo cual frecuentemente redundante en una protección débil.
A9	<b>Negación de Servicio</b>	Los agresores pueden consumir los recursos de la aplicación Web al punto de que otros usuarios legítimos no puedan ya acceder o usar la aplicación. Los agresores también pueden dejar a los usuarios fuera de sus cuentas y hasta causar que falle una aplicación entera.
A10	<b>Administración de Configuración Insegura</b>	Tener una configuración de servidor estándar es crítico para asegurar una aplicación Web. Estos servidores tienen muchas opciones de configuración que afectan la seguridad y no son seguro desde la instalación original del software.



## A1 Entrada no validada

### A1.1 Descripción

Las aplicaciones Web usan entradas de llamadas HTTP (y ocasionalmente archivos) para determinar cómo responder. Los atacantes pueden entrometerse con cualquier parte del llamado HTTP, incluyendo el URL, cadena de consulta (querystring), encabezados, cookies, campos de formularios (normales o escondidos), para tratar de pasar por encima de los mecanismos de seguridad del sitio. Nombres comunes para ataques de manipulación de entradas incluyen: navegación forzada, inserción de comandos, cross site scripting, desbordamiento de búfer, ataques de formato de cadena de caracteres, inyección de SQL, manipulación de cookies y campos escondidos. Cada uno de estos ataques es descrito detalladamente más adelante en este documento.

- A4 – Fallas de Cross Site Scripting trata sobre las entradas que contiene pequeños programas para ser ejecutados en el navegador de otros usuarios
- A5 – Desbordamiento de búfer trata sobre las entradas que ha sido diseñadas para sobrescribir el espacio de ejecución de un programa
- A6 – Fallas de inyección trata sobre las entradas que son modificadas para contener comandos ejecutables

Algunos sitios tratan de protegerse ellos mismos por medio del filtrado de entradas maliciosas. El problema es que hay muchas formas diferentes de codificar información. Estos formatos de codificación no son como la encriptación, ya que éstos son triviales para decodificar. Aun así, los desarrolladores a veces olvidan decodificar todos los parámetros a su forma más simple antes de usarlos. Los parámetros tienen que ser convertidos a su forma más simple antes de ser validados, de otro modo, las entradas maliciosas puede ser enmascaradas y pueden pasar a través de los filtros. El proceso de simplificar estas codificaciones es llamado “canonicalization”. Ya que casi todas las entradas HTTP pueden ser representadas en múltiples formatos, esta técnica puede ser usada para obscurecer cualquier ataque utilizando las vulnerabilidades descritas en este documento. Esto hace el filtrado muy difícil.

Un número sorprendente de aplicaciones Web utilizan sólo mecanismos de validación de entradas a nivel del cliente. Los mecanismos de validación a nivel del cliente son fácilmente sobrepasados, dejando la aplicación Web sin ninguna protección contra parámetros maliciosos. Los atacantes pueden generar sus propias llamadas HTTP usando herramientas tan simples como Telnet. Estos no necesitan poner atención a las cosas que el desarrollador tenía la intención de que ocurriesen a nivel del cliente. Hay que notar que la validación a nivel del cliente es una buena idea para el rendimiento y la usabilidad, pero no tiene ningún beneficio de seguridad. Las verificaciones a nivel del servidor son requeridas para defender contra los ataques de manipulación de parámetros. Ya estando éstas en funcionamiento, la verificación a nivel del cliente puede también ser incluida para mejorar la experiencia del usuario para usuarios legítimos y/o reducir la cantidad de tráfico inválido hacia el servidor.

Estos ataques están siendo cada vez más comunes, a la vez que el número de herramientas que soportan generación de parámetros (fuzzing), corrupción y fuerza bruta continúan creciendo. El impacto al usar datos de entrada no validados no debería ser subestimado. Una gran cantidad de ataques serían difíciles o imposibles si los desarrolladores simplemente pudieran validar la entrada de datos antes de usarla. A menos que una aplicación Web tenga un mecanismo fuerte y centralizado para validar todas las entradas para las llamadas HTTP (o cualquier otra fuente), las vulnerabilidades basadas en entradas maliciosas seguirán existiendo.

### A1.2 Ambientes afectados

Todos los servidores Web, servidores de aplicaciones, y ambientes de aplicación Web son susceptibles a la manipulación de parámetros.



### A1.3 Ejemplos y referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web, Capítulo 8: Validación de Datos <http://www.owasp.org/documentation/guide/>
- Proyecto Modsecurity (Modulo Apache para validación HTTP) <http://www.modsecurity.org>
- Cómo construir un motor de validación para llamadas HTTP (validación J2EE con Stinger) <http://www.owasp.org/columns/jeffwilliams/jeffwilliams2>
- Tener tu pastel y comértelo también (Have Your Cake and Eat it Too) (Validación .NET) <http://www.owasp.org/columns/jpoteet/jpoteet2>

### A1.4 ¿Cómo determinar si usted es vulnerable?

Cualquier parte de una llamada HTTP usada por una aplicación Web sin ser cuidadosamente validada se conoce como parámetro "sucio". La manera más simple de encontrar el uso de un parámetro "sucio" es tener una revisión de código detallado, buscando todas las llamadas donde la información es extraída de una llamada HTTP. Por ejemplo, en una aplicación J2EE, estos son los métodos en la clase HttpServletRequest. Después se puede seguir el código para ver donde se ha utilizado esa variable. Si la variable no se revisa antes de ser usada, seguramente habrá un problema. En Perl, debe considerar el uso de la opción "sucio" (-T).

También es posible encontrar el uso de parámetros sucios empleando herramientas como OWASP's WebScarab. Al aceptar valores inesperados en llamadas HTTP y observando las respuestas de las aplicaciones Web, se pueden identificar los lugares donde se usan los parámetros sucios.

### A1.5 ¿Cómo auto protegerse?

La mejor forma de prevenir la manipulación de parámetros es asegurando que todos los parámetros sean validados antes de que sean usados. Un componente centralizado o librería sería una de las formas más efectivas ya que el código que aplica la verificación debería estar en un sólo lugar. Cada parámetro debe ser verificado contra un formato estricto que especifica exactamente qué entrada de datos debe ser permitida. Es improbable que alcances "negativos" que involucran filtrado de entradas maliciosas o alcances basados en firmas (signatures) sean efectivos, además pueden ser difíciles de mantener.

Los parámetros deben ser validados contra una especificación "positiva" que define:

- Tipo de datos (string, integer, real, etc...)
- Conjunto de caracteres permitidos
- Longitud mínima y máxima
- Si nulo es permitido
- Si el parámetro es requerido o no
- Si los duplicados son permitidos
- El rango numérico
- Valores específicos permitidos (enumeración)
- Patrones específicos (expresiones regulares)

Una nueva clase de aparatos de seguridad conocidos como cortafuegos (firewalls) de aplicaciones Web pueden proveer algunos servicios de validación de parámetros. Sin embargo, para que éstos sean efectivos, el aparato debe estar configurado con una estricta definición de qué es válido para cada parámetro del sitio. Esto incluye proteger adecuadamente todos los tipos de entrada de las llamadas HTTP, incluyendo URLs, formularios, cookies, querystrings, campos escondidos y parámetros.

El proyecto OWASP Filters está produciendo componentes reutilizables en diferentes lenguajes para ayudar a evitar diferentes formas de manipulación de parámetros. El motor de validación de llamadas HTTP, Stinger ([stinger.sourceforge.net](http://stinger.sourceforge.net)), fue también desarrollado por OWASP para ambientes J2EE.



## A2 Control de Acceso Interrumpido

### A2.1 Descripción

El control de acceso, algunas veces llamado autorización, es el cómo una aplicación Web permite el acceso a contenido y funciones a algunos usuarios y a otros no. Estas verificaciones son desarrolladas después de la autenticación, y gobiernan lo que pueden hacer los usuarios “autorizados”. El control de acceso suena como un problema simple pero es insidiosamente difícil de implementar correctamente. El modelo de control de acceso de una aplicación Web tiene un lazo estrecho con el contenido y funciones que el sitio provee. Además, los usuarios pueden caer dentro de un número de grupos o roles con diferentes capacidades o privilegios.

Los desarrolladores frecuentemente menosprecian la dificultad de implementar un mecanismo de control de acceso confiable. Muchos de estos esquemas no fueron diseñados deliberadamente, sino simplemente han evolucionado junto con el sitio Web. En estos casos, las reglas de control de acceso son insertadas en varias locaciones en todo el código. Al acercarse al despliegado del sitio, la colección ad hoc de reglas se convierte tan difícil de manejar que es casi imposible de comprender.

Muchas de estas fallas en los esquemas de control de acceso no son difíciles de encontrar y explotar. Frecuentemente, todo lo que se requiere es hacer una petición por funciones o contenido que no debe ser concedido. Una vez que se descubre una falla, las consecuencias de un esquema de control de acceso con fallas pueden ser devastadoras. Además de visualizar contenido no autorizado, un atacante podría alterar o borrar contenido, realizar funciones no autorizadas, o aún tomar el control de la administración del sitio.

Un tipo específico de problema de control de acceso es la interfase administrativa que permite a los administradores del sitio manejar un sitio mediante Internet. Tales características son usadas frecuentemente para permitir a los administradores del sitio manejar eficientemente los usuarios, datos, y contenido en su sitio. En muchos casos, los sitios soportan una variedad de roles administrativos para permitir una granulación más fina de la administración del sitio. Debido a su poder, estas interfaces son frecuentemente los objetivos primarios de un ataque ya sea por personas del exterior o del interior.

### A2.2 Ambientes afectados

Todos los servidores Web conocidos, servidores de aplicación, y aplicaciones Web son susceptibles a por lo menos alguna de estas cuestiones. Aún si un sitio es completamente estático, si no está apropiadamente configurado, los piratas informáticos pueden obtener acceso a archivos sensibles y manipularlos o realizar otro tipo de daños.

### A2.3 Ejemplos y referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros, Capítulo 8: Control de Acceso: <http://www.owasp.org/guide/>
- Control de Acceso (también conocido como Autorización) en Su Aplicación J2EE <http://www.owasp.org/columns/jeffwilliams/jeffwilliams3>
- <http://www.infosecuritymag.com/2002/jun/insecurity.shtml>

### A2.4 ¿Cómo determinar si usted es vulnerable?

Virtualmente todos los sitios tienen algunos requisitos de control de acceso. Por lo tanto, una política de control de acceso debe ser claramente documentada. También la documentación de diseño debe capturar una propuesta para reforzar esta política. Si esta documentación no existe, entonces es muy probable que el sitio sea vulnerable.

El código que implementa la política de control de acceso debe ser verificado. Tal código debe estar bien estructurado, ser modular y preferiblemente centralizado. Una verificación detallada del código debe ser realizada para validar la implementación correcta del control de acceso. Además, las pruebas de penetración pueden ser muy útiles para determinar si hay problemas en el esquema de control de acceso.



Investigue cómo es administrado su sitio Web. Querrá descubrir cómo son hechos los cambios a las páginas Web, dónde son probados y cómo son transportados al servidor de producción. Si los administradores pueden hacer cambios de forma remota, deseará saber cómo son protegidos esos canales de comunicación. Revise cuidadosamente cada interfase para asegurarse que solamente los administradores autorizados tienen el acceso permitido. También, si existen diferentes tipos o grupos de datos a los que se puede acceder mediante la interfase, asegúrese que sólo se puede acceder a los datos autorizados. Si tales interfaces emplean comandos externos, verifique el uso de tales comandos para asegurar que no están sujetos a ninguna falla de inyección de comandos descritos en este documento

## A2.5 ¿Cómo auto protegerse?

El paso más importante es pensar a partir de los requisitos de control de acceso de una aplicación y capturar este pensamiento en una política de seguridad para aplicaciones Web. Recomendamos encarecidamente el uso de una matriz de control de acceso para definir las reglas de control de acceso. Sin documentar la política de seguridad, no existe una definición de lo que significa estar seguro para ese sitio. La política debe documentar qué tipos de usuarios pueden acceder al sistema y a cuáles funciones y contenidos se les debería permitir el acceso a cada uno de estos tipos de usuarios. El mecanismo de control de acceso debe ser probado extensivamente para asegurar que no hay forma de evitarla. Estas pruebas requieren de una variedad de cuentas e intentos exhaustivos para acceder contenidos o funciones no autorizadas.

Algunos asuntos de control de acceso específico incluyen:

- Id's inseguros – la mayoría de los sitios Web utilizan alguna forma de id, llave o índice como una manera de referenciar usuarios, roles, contenido, objetos o funciones. Si un atacante puede adivinar estos id's y los valores provistos no son validados para asegurar que son autorizados para el usuario actual, el atacante puede ejercer libremente el esquema de control de acceso para ver a qué es lo que puede acceder. Las aplicaciones Web no deben depender del secreto de ningún id para su protección.
- Navegación Forzada Saltando la Verificación del Control de Acceso – muchos sitios requieren que los usuarios pasen por ciertas verificaciones antes de que se les garantice el acceso a ciertos URLs que están típicamente colocados 'más profundamente' en el sitio. Estas verificaciones no deben ser evitables por un usuario que simplemente se salta la página que contiene la verificación de seguridad.
- Cruce de Dirección de Fichero ("path traversal") – este ataque involucra proveer información relativa a la dirección de fichero (Ej., `../../../../target_dir/target_file`) como parte de una petición por información. Tales ataques tratan de acceder archivos que normalmente no son accesibles para nadie o que podrían ser negados si se solicitan directamente. Tales ataques pueden ser enviados en URLs así como en cualquier otra entrada que en última instancia acceda a un archivo (Ej., llamadas a sistema y la interfase de comando "shell command").
- Permisos de Archivos – muchos servidores y aplicaciones Web se basan en listas de control de acceso provistas por el sistema de archivos de la plataforma subyacente. Aun si casi todos los datos son almacenados en servidores de soporte DBMS ("backend servers"), siempre hay archivos almacenados localmente en el servidor y aplicación Web que no deben ser públicamente accesibles, particularmente archivos de configuración, archivos por defecto y scripts que son instalados en la mayoría de los servidores y aplicaciones Web. Solamente archivos que son específicamente diseñados para ser presentados a los usuarios Web deben ser marcados como legibles usando los mecanismos de permiso de los SO's, la mayoría de los directorios no deben ser legibles y muy pocos archivos, si acaso existen, deben ser marcados como ejecutables.
- Cache del Lado Cliente ("client side caching") – muchos usuarios acceden a aplicaciones Web de computadoras compartidas localizadas en bibliotecas, escuelas, aeropuertos y otros puntos de acceso público. Los navegadores frecuentemente ponen en cache las páginas Web que pueden ser accedidas por atacantes para ganar acceso a partes o sitios de otra forma inaccesibles. Los desarrolladores deben usar múltiples mecanismos, incluyendo cabeceras HTTP y meta etiquetas ("meta tags"), para asegurarse que las páginas que contienen información sensible no sean capturadas en cache por el navegador del usuario.

Existen algunos componentes de seguridad a nivel aplicación que ayudan en el refuerzo apropiado de algunos aspectos de su esquema de control de acceso. Nuevamente, como con la validación de parámetro, para que sea efectiva, el componente debe ser configurado con una estricta definición de qué peticiones de acceso son válidas para su sitio. Cuando se utilice tal componente, debe ser cuidadoso para comprender exactamente qué ayuda puede proveer para usted el



componente de control dada la política de seguridad de su sitio y qué parte de su política de control de acceso no puede manejar el componente y por lo tanto debe ser manejado apropiadamente en su propio código.

Para funciones administrativas, la recomendación primordial, si es del todo posible, es nunca permitir el acceso del administrador a través de la puerta principal de su sitio. Dado el poder de estas interfaces, la mayoría de las organizaciones no deben aceptar el riesgo de que estas interfaces estén disponibles para ataques externos. Si es absolutamente requerido el acceso del administrador remoto, esto puede ser logrado sin tener que abrir la puerta principal del sitio. El uso de la tecnología VPN (“Virtual Private Network”) puede ser utilizado para proveer el acceso del administrador externo al interior de la red de la compañía (o sitio) desde la cual un administrador puede entonces acceder al sitio a través de una conexión de soporte DBMS (“backend”) protegida.



## A3 Administración de Autenticación y Sesión Interrumpida

### A3.1 Descripción

La administración de autenticación y sesión incluye todos los aspectos del manejo de la autenticación de usuario y la administración de sesiones activas. La autenticación es un aspecto crítico de este proceso, pero inclusive mecanismos de autenticación sólidos pueden ser minados por funciones de administración de credenciales defectuosas, incluyendo el cambio de contraseña, olvidé mi contraseña, recordar mi contraseña, actualización de cuenta y otras funciones relacionadas. Porque los ataques de paso (“walk by”) son probablemente dirigidos a muchas aplicaciones Web, todas las funciones de administración de cuentas deben requerir re-autenticación aun si el usuario tiene un id de sesión válido.

La autenticación de usuario en Web típicamente involucra el uso de un id de usuario y una contraseña. Métodos más sólidos de autenticación tales como software y hardware basados en tokens criptográficos o biometría están comercialmente disponibles, pero tales mecanismos son de un costo prohibitivo para la mayoría de las aplicaciones Web. Una amplia colección de defectos en la administración de cuentas y sesiones puede traer como consecuencia el comprometer las cuentas de usuario o administración del sistema. Los equipos de desarrollo frecuentemente menosprecian la complejidad de diseñar un esquema de administración de autenticación y sesión que proteja adecuadamente las credenciales en todos los aspectos del sitio.

Las aplicaciones Web deben establecer sesiones para mantener el rastro del flujo de peticiones de cada usuario. HTTP no provee esta capacidad, así que las aplicaciones Web deben crearlas por sí mismas. Frecuentemente, los ambientes de aplicación Web proveen una capacidad de sesión, pero muchos desarrolladores prefieren crear sus propios tokens de sesión. En cualquier caso, si los tokens de sesión no están apropiadamente protegidos, un atacante puede apropiarse de una sesión activa y asumir la identidad del usuario. Diseñar un esquema para crear tokens sólidos de sesión y protegerlos a través de su ciclo de vida ha probado su efectividad para muchos desarrolladores.

A menos que todas las credenciales de autenticación e identificadores de sesión sean protegidos con SSL todo el tiempo y protegidas contra su revelación por causa de otras fallas, tales como “cross site scripting”, un atacante puede apropiarse la sesión del usuario y asumir su identidad.

### A3.2 Ambientes Afectados

Todos los ambientes de servidores Web conocidos, servidores de aplicación y aplicaciones Web son susceptibles a los problemas de administración de autenticación y sesión interrumpida.

### A3.3 Ejemplos y Referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros, Capítulo 6: Autenticación y Capítulo 7: Administración de Sesión: <http://www.owasp.org/guide/>
- Documento (“white paper”) sobre “Vulnerabilidad en la Fijación de Sesión en Aplicaciones Basadas en Web”: [http://www.acros.si/papers/session\\_fixation.pdf](http://www.acros.si/papers/session_fixation.pdf)
- Documento (“white paper”) sobre Recuperación de Contraseñas para Aplicaciones Basadas en Web: <http://fishbowl.pastiche.org/archives/docs/PasswordRecovery.pdf>

### A3.4 ¿Cómo determinar si usted es vulnerable?

Ambas pruebas de verificación de código y penetración pueden ser usadas para diagnosticar problemas de administración de autenticación y sesión. Cuidadosamente verifique cada aspecto de su mecanismo de autenticación para asegurar que las credenciales de usuario están protegidas en todo momento, mientras se encuentran detenidas (Ej., en disco) y mientras están en tránsito (Ej., durante la conexión inicial “login”). Revisa cada mecanismo disponible para cambiar las credenciales de usuario con el fin de asegurar que sólo un usuario autorizado pueda cambiarlas. Revise su mecanismo de administración de sesión para asegurar que los identificadores de sesión siempre estén protegidos y sean utilizados de tal forma que se minimice la probabilidad de exposición accidental u hostil.



### A3.5 ¿Cómo auto protegerse?

El uso apropiado y cuidadoso de mecanismos de administración de autenticación y sesión hechos a la medida o sacados del estante (“off the shelf”) deben reducir significativamente la probabilidad de algún problema en ésta área. Definiendo y documentando la política de su sitio con respecto a la administración de forma segura de las credenciales de usuario es un buen primer paso. Asegurar que su implementación refuerza consistentemente esta política es la clave para tener un mecanismo de administración de autenticación y sesión seguro y robusto. Algunas áreas críticas incluyen:

- **Contraseñas Resistentes** – las contraseñas deben tener restricciones que impongan un tamaño y complejidad mínima . La complejidad típicamente requiere el uso de combinaciones mínimas de caracteres alfabéticos, numéricos, y/o no-alfanuméricos en la contraseña de usuario (Ej., por lo menos uno de cada uno). A los usuarios se les debe requerir cambiar su contraseña periódicamente. A los usuarios se les debe impedir re-utilizar contraseñas anteriores.
- **Uso de Contraseña** – se debe restringir a los usuarios a un número definido de intentos de conexión por unidad de tiempo y los repetidos intentos de conexión fallidos deben ser registrados. Las contraseñas provistas durante los intentos fallidos de conexión no deben ser almacenados ya que esto puede exponer una contraseña de usuario a cualquiera que logre acceder a este registro. El sistema no debe indicar si fue el nombre de usuario o la contraseña lo que estaba equivocado si falla un intento de conexión. Los usuarios deben ser informados de la fecha/hora de su última conexión exitosa y el número de intentos de acceso fallidos a su cuenta desde esa hora.
- **Control de Cambios de Contraseña** – un solo mecanismo de cambio de contraseña debe ser usado donde sea que se permita cambiar a los usuarios su contraseña, independientemente de la situación. A los usuarios siempre se les debe solicitar el ingresar ambas contraseñas, la anterior y la actual, cuando cambian su contraseña (como todas las cuentas informáticas). Si las contraseñas olvidadas son enviadas por correo electrónico a los usuarios, el sistema debe requerir del usuario que se re-autentifique cuantas veces el usuario cambie su dirección de correo electrónico, de lo contrario un atacante que ha tenido acceso temporalmente a su sesión (Ej., acercándose a su computadora mientras está conectado) puede simplemente cambiar su dirección de correo electrónico y solicitar que una contraseña ‘olvidada’ les sea enviada vía correo electrónico.
- **Almacenamiento de Contraseña** – todas las contraseñas deben ser almacenadas ya sea de forma encriptada (“encrypted”) o como hashes para protegerlas de ser expuestas, independientemente de donde son almacenados. La forma de hash es preferible ya que no es reversible. La encriptación debe ser usada cuando se necesita la contraseña en texto llano, tal como cuando se utiliza la contraseña para conectarse a otro sistema. Las contraseñas nunca deben ser insertadas directamente en el código fuente. Las llaves de descifrado deben ser protegidas fuertemente para asegurar que no puedan ser arrebatadas y usadas para descifrar el archivo de contraseñas.
- **Protegiendo Credenciales en Tránsito** – la única técnica efectiva es encriptar completamente la transacción de acceso al sistema es usando algo como SSL. Las transformaciones sencillas de las contraseñas tales como aplicar un hash en el cliente antes de la transmisión provee poca protección ya que la versión cifrada puede simplemente ser interceptada y retransmitida aunque la contraseña en texto llano no pueda ser conocida.
- **Protección del Id de Sesión** – idealmente, toda la sesión de usuario debe ser protegida mediante SSL. Si esto es hecho, entonces el id de sesión (Ej., cookie de sesión) no puede ser tomado de la red, lo cual es el mayor riesgo de exposición de un id de sesión. Si no es viable el uso de SSL por el desempeño u otras razones, entonces los id’s de sesión deben ser protegidos de alguna otra forma. En primera instancia, éstos nunca deben ser incluidos en el URL ya que pueden ser capturados en memoria rápida (“cached”) por el navegador, mandados en la cabecera referenciada o accidentalmente re-enviados a un ‘amigo’. Los id’s de sesión deben ser números largos, complicados y aleatorios que no puedan ser fácilmente adivinados. Los id’s de sesión pueden también ser cambiados frecuentemente durante una sesión para reducir el tiempo en que es válido un id de sesión. Los id’s de sesión deben ser cambiados cuando se cambia a SSL, se autentifica u ocurre otra transacción importante. Los id’s de sesión escogidos por un usuario nunca deben ser aceptados.
- **Listas de cuentas** –los sistemas deben ser diseñados para evitar que los usuarios logren acceso a una lista de los nombres de cuentas del sitio. Si las listas de usuarios deben ser mostradas, es recomendable que alguna forma de pseudónimo (“screen name”) que corresponda a la cuenta actual sea listada en su lugar. De esa forma el pseudónimo no puede ser usado durante un intento de conexión o algún otro intento de obtener la cuenta de usuario.
- **Memoria Rápida (“Cache”) del Navegador** – los datos de autenticación y sesión nunca deben ser suministrados como parte de un GET, en su lugar siempre debe ser utilizado POST. Las páginas de autenticación deben ser marcadas con todas las diversidades de etiquetas de no uso de memoria rápida para prevenir que alguien use el botón de retorno en el navegador del usuario para copiar la página de conexión y re-transmitir las credenciales



previamente tecleadas. Muchos navegadores de hoy soportan la bandera (“flag”) de ‘autocomplete=false’ para prevenir el almacenamiento de credenciales en la memoria rápida (“cache”) de auto completar.

- Relación de Confianza – la arquitectura de su sitio debe evitar la confianza implícita entre componentes siempre que sea posible. Cada componente debe autenticarse por sí mismo con cualquier otro componente con el que esté interactuando, a menos que exista una fuerte razón para no hacerlo (tal como el desempeño o la falta de un mecanismo utilizable). Si las relaciones de confianza son requeridas, procedimientos sólidos y mecanismos arquitectónicos deben ser usados para asegurar que no se puede abusar de tal confianza al evolucionar la arquitectura del sitio con el tiempo.



## A4 Fallas de Cross-Site Scripting (XSS)

### A4.1 Descripción

Las vulnerabilidades de cross-site scripting (algunas veces referido como XSS) ocurren cuando un atacante utiliza una aplicación Web para mandar código malicioso, generalmente en la forma de un *script*, a un usuario diferente. Estas fallas están muy generalizadas y ocurren donde quiera que una aplicación Web utilice entradas de datos de un usuario sin validar la salida que ésta genera.

Un atacante puede usar 'cross site scripting' para mandar un *script* malicioso a un usuario ingenuo. El navegador del usuario final no tiene forma de saber que el *script* no debe ser confiable y lo ejecutará, porque piensa que vino de una fuente confiable, el *script* malicioso puede acceder a cualquier cookie, token de sesión u otra información sensible retenida por su navegador y utilizada con ese sitio. Estos pequeños programas pueden hasta re-escribir el contenido de una página HTML.

Los ataques XSS pueden generalmente ser clasificados en dos categorías: almacenados y reflejados. Los ataques almacenados son aquellos en donde el código inyectado es permanentemente almacenado en el servidor objetivo, tales como bases de datos, en un foro de mensajes, un archivo de visitante, campo comentado, etc. Entonces la víctima recupera el *script* malicioso del servidor cuando solicita la información almacenada. Los ataques reflejados son aquellos donde el código inyectado es reflejado desde el servidor, tal como un mensaje de error, resultado de una búsqueda o cualquier otra respuesta que incluya alguna o todas las entradas mandadas al servidor como parte de una petición. Los ataques reflejados son entregados a las víctimas mediante otra ruta, tal como un mensaje de correo electrónico o algún otro servidor Web. Cuando un usuario es engañado para hacer clic a un enlace malicioso o ante la presentación artesanal de una forma, el código inyectado viaja al servidor Web vulnerable, el cual refleja el ataque de regreso al navegador del usuario. El navegador ejecuta el código porque viene de un servidor 'confiable'.

La consecuencia de un ataque XSS son las mismas, independientemente de si es almacenado o reflejado. La diferencia es cómo llega el efecto destructivo al servidor. No se engañe al pensar que un sitio de "solo lectura" o de "presencia literaria corporativa en línea sin interacción" ("*brochureware*") no es vulnerable a ataques reflejados XSS serios. XSS puede causar una variedad de problemas al usuario final que varían en la severidad, desde una molestia hasta el comprometer completamente una cuenta. Los ataques XSS más severos involucran la divulgación de una cookie de sesión de usuario, permitiendo a un atacante secuestrar la sesión de usuario y apoderarse de la cuenta. Otros ataques dañinos incluyen la divulgación de los archivos del usuario final, instalación de programas de caballos de Troya, re-dirigiendo al usuario a alguna otra página o sitio y modificando la presentación del contenido. Una vulnerabilidad XSS que permite a un atacante modificar un comunicado de prensa o un artículo de noticia puede afectar el precio de mercado de una compañía o disminuir la confianza del consumidor. Una vulnerabilidad XSS en un sitio farmacéutico podría permitir a un atacante modificar la información de dosificación resultando en una sobredosis.

Los atacantes usan frecuentemente una variedad de métodos para codificar la porción maliciosa de la etiqueta, tal como el utilizar Unicode (*código de 16 bits para exponer signos en el ordenador, parecido al ASCII pero con un número mayor de signos, lo que permite el uso de todos los idiomas mundiales*), de forma que la solicitud parezca menos sospechosa al usuario. Hay cientos de variantes de estos ataques, incluyendo versiones que ni siquiera requieren ningún símbolo < >. Por esta razón el intentar "filtrar" estos *scripts* probablemente no tendrá éxito. En lugar de ello recomendamos validar las entradas contra una especificación positiva rigurosa de lo que es esperado. Los ataques XSS usualmente vienen en la forma de JavaScript incrustado. Sin embargo, cualquier contenido activo incrustado es una fuente potencial de daño, incluyendo: ActiveX (OLE), Vbscript, Shockwave, Flash y más.

Las cuestiones sobre XSS también pueden estar presentes en los servidores de aplicación y Web subyacentes. La mayoría de los servidores de aplicación y Web generan páginas Web simples para mostrar en caso de varios errores, tal como 404 'página no encontrada' o 500 'error interno del servidor'. Si estas páginas muestran como respuesta cualquier información de la solicitud del usuario, tal como la URL a la que trataban de acceder, pueden ser vulnerables a un ataque reflejado XSS.

La probabilidad de que un sitio contenga vulnerabilidades XSS es extremadamente alta. Hay una gran variedad de formas de engañar a las aplicaciones Web al transmitir *scripts* maliciosos. Los desarrolladores que intentan filtrar las partes maliciosas de estas solicitudes es muy probable que pasen por alto posibles ataques o codificaciones. Encontrar estas fallas no es enormemente difícil para los atacantes, ya que todo lo que necesitan es un navegador y algo de tiempo. Hay



numerosas herramientas gratuitas disponibles que ayudan a los piratas informáticos a encontrar estas fallas así como a crear cuidadosamente ataques XSS e inyectarlos en los sitios objetivo.

## A4.2 Ambientes Afectados

Todos los servidores Web, servidores de aplicación y ambientes de aplicación Web son susceptibles de cross site scripting.

## A4.3 Ejemplos y Referencias

- Preguntas y Respuestas (“FAQ”) sobre Cross Site Scripting FAQ: <http://www.cgisecurity.com/articles/xss-faq.shtml>
- Asesor CERT Sobre Etiquetas (“tags”) HTML Maliciosas: <http://www.cert.org/advisories/CA-2000-02.html>
- CERT “Comprendiendo la Mitigación de Contenido Malicioso”:  
[http://www.cert.org/tech\\_tips/malicious\\_code\\_mitigation.html](http://www.cert.org/tech_tips/malicious_code_mitigation.html)
- Resumen Ejecutivo sobre la Exposición de Seguridad Cross-Site Scripting:  
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/ExSumCS.asp>
- Comprendiendo la Causa y Efecto de las Vulnerabilidades XSS: <http://www.technicalinfo.net/papers/CSS.html>
- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros, Capítulo 8: Validación de Datos  
<http://www.owasp.org/documentation/guide/>
- Como Construir un Motor de Validación de Petición HTTP (J2EE validación con Stinger)  
<http://www.owasp.org/columns/jeffwilliams/jeffwilliams2>
- Ten Tu Pastel y Cómetelo También (validación .NET) <http://www.owasp.org/columns/jpoteet/jpoteet2>

## A4.4 ¿Cómo determinar si usted es vulnerable?

Las fallas de XSS pueden ser difíciles de identificar y remover de una aplicación Web. La mejor manera de encontrar fallas es realizar una revisión de seguridad del código y buscar por todos aquellos lugares donde una entrada de una solicitud HTTP podría posiblemente hacerse camino hacia la salida de HTML. Note que una diversidad de diferentes etiquetas HTML puede ser usada para transmitir un JavaScript malicioso. Nessus, Nikto y algunas otras herramientas disponibles pueden ayudar a buscar estas fallas en un sitio Web, pero sólo pueden escarbar en la superficie. Si una parte del sitio Web es vulnerable, existe una alta probabilidad de que también existan otros problemas.

## A4.5 ¿Cómo auto protegerse?

La mejor forma de proteger una aplicación Web de ataques XSS es asegurar que su aplicación desarrolla una validación de todas las cabeceras, *cookies*, cadenas de petición, campos de formularios y campos escondidos (Ej., todos los parámetros) contra una especificación rigurosa de lo que debe ser permitido. La validación no debe tratar de identificar contenido activo y removerlo, filtrarlo o desinfectarlo. Hay muchos tipos de contenido activo y también muchas formas de codificarlo para evitar los filtros para tal contenido. Recomendamos contundentemente una política de seguridad ‘positiva’ que especifique lo que es permitido. Políticas basadas en identificación de ataques o ‘negativas’ son difíciles de mantener y probablemente serán incompletas.

La salida codificada de usuario suministrada también puede abatir las vulnerabilidades XSS evitando que *scripts* insertados sean transmitidos a los usuarios en una forma ejecutable. Las aplicaciones pueden ganar una protección significativa contra los ataques basados en JavaScript al convertir, en todas las salidas generadas, los caracteres siguientes a la entidad de codificación HTML apropiada::

De:	A:
<	&lt;
>	&gt;
(	&#40;
)	&#41;



#	&#35;
&	&#38;

El proyecto de Filtros de OWASP está produciendo componentes re-utilizables en varios lenguajes para ayudar a prevenir muchas formas de falsificación de parámetros, incluyendo la inyección de ataques XSS. OWASP también ha liberado CodeSeeker, una aplicación a nivel cortafuego. Además, el programa de entrenamiento WebGoat de OWASP tiene lecciones sobre Cross Site Scripting y codificación de datos.



## A5 Desbordamiento del Búfer

### A5.1 Descripción

Los atacantes utilizan el desbordamiento del búfer para corromper la ejecución de la pila (“*stack*”) de una aplicación Web. Mediante el envío de entradas hábilmente urdidas hacia una aplicación Web, un atacante puede causar que una aplicación Web ejecute código arbitrario – tomando el control efectivo de una máquina. Los desbordamientos del búfer no son fáciles de descubrir y aun cuando uno es descubierto, es generalmente extremadamente difícil de explotar. No obstante, los atacantes se las han ingeniado para identificar los desbordamientos del búfer en una asombrosa matriz de productos y componentes. Otra clase similar de fallas es conocida como ataques de cadena de formato

Las fallas de desbordamiento del búfer pueden estar presentes ya sea en los productos del servidor Web o en el servidor de aplicación que prestan el servicio del sitio para los aspectos estáticos y dinámicos o la aplicación Web en sí misma. Las fallas de desbordamiento del búfer encontradas en productos ampliamente utilizados en servidores probablemente serán extensamente conocidas y pueden exponer un riesgo significativo a los usuarios de estos productos. Cuando las aplicaciones Web usan bibliotecas, tales como bibliotecas gráficas para generar imágenes, éstas se abren por sí mismas a ataques potenciales de desbordamiento del búfer.

El desbordamiento del búfer también puede ser encontrado en el código de aplicaciones Web hechas a la medida e incluso pueden tener una mayor probabilidad a ataques dada la falta de escrutinio por las cuales pasa típicamente una aplicación Web. Las fallas de desbordamiento del búfer en las aplicaciones Web hechas a la medida tienen menor probabilidad de ser detectadas porque normalmente habrá muchos menos piratas informáticos tratando de encontrar y explotar tales fallas en una aplicación específica. Si se descubre en una aplicación hecha a la medida, la habilidad de explotar las fallas (más que hacer que la aplicación deje de funcionar) es significativamente reducida por el hecho de que el código fuente y los mensajes de error detallados para la aplicación normalmente no están disponibles para el pirata informático.

### A5.2 Ambientes Afectados

Casi todos los servidores Web, servidores de aplicación y ambientes de aplicación Web conocidos son susceptibles de desbordamiento de memoria intermedia, la excepción notable son los ambientes Java y J2EE, los cuales son inmunes a estos ataques (excepto por los desbordamientos propios de JVM).

### A5.3 Ejemplos y Referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros, Capítulo 8: Validación de Datos: <http://www.owasp.org/documentation/guide/>
- Aleph One, “Destrozando la Pila por Diversión y Beneficio”, <http://www.phrack.com/show.php?p=49&a=14>
- Mark Donaldson, “Dentro del Ataque de Desbordamiento de Memoria Intermedia: Mecanismo, Método y Prevención”, [http://rr.sans.org/code/inside\\_buffer.php](http://rr.sans.org/code/inside_buffer.php)

### A5.4 ¿Cómo determinar si usted es vulnerable?

Para productos de servidor y bibliotecas, manténgase al día con los últimos reportes sobre errores para los productos que está utilizando. Para las aplicaciones de software hechas a la medida, todo código que acepte entradas de usuario mediante peticiones HTTP debe ser revisado para asegurar que puede manejar apropiadamente entradas arbitrariamente grandes.

¿Cómo auto protegerse?

Manténgase al día con los últimos reportes de errores para los productos de servidores Web y de aplicación y otros productos en su infraestructura de Internet. Aplique los últimos parches para estos productos. Periódicamente examine su sitio Web con uno o más de los buscadores(Scanners) comúnmente disponibles que buscan fallas de desbordamiento de memoria intermedia en sus productos para servidores y en sus aplicaciones Web hechas a la medida.



Para el código de sus aplicaciones hechas a la medida, necesitarás revisar todo código que acepte entradas de los usuarios mediante solicitudes HTTP y asegurar que se provee el tamaño apropiado para verificar todas esas entradas. Esto debe ser hecho aun para ambientes que no son susceptibles a tales ataques ya que entradas demasiado largas que no son capturadas podrían causar negación de servicio u otros problemas operacionales.



## A6 Fallas de Inyección

### A6.1 Descripción

Las fallas de inyección permiten a los atacantes transmitir código malicioso a otro sistema a través de una aplicación Web. Estos ataques incluyen llamadas al sistema operativo mediante llamadas al sistema, el uso de programas externos vía interfaces de comando y también llamadas a servidores de soporte de bases de datos mediante SQL (Ej., Inyección SQL). *scripts* escritos en Perl, Python, y otros lenguajes pueden ser inyectados en aplicaciones Web pobremente diseñadas y ejecutadas. En cualquier momento en que una aplicación Web utiliza un intérprete de cualquier tipo hay peligro de un ataque por inyección.

Muchas aplicaciones Web utilizan características del sistema operativo y programas externos para realizar sus funciones. "Sendmail" es probablemente el programa externo más frecuentemente utilizado, pero muchos otros programas también son usados. Cuando una aplicación Web pasa información de una petición HTTP como parte de una solicitud externa, debe ser cuidadosamente examinada. De lo contrario, el atacante puede inyectar (meta) caracteres, comandos maliciosos o modificadores de comando en la información y la aplicación Web, sin dudar, pasa éstos al sistema externo para su ejecución.

La inyección SQL es una forma de inyección peligrosa y está particularmente generalizada. Para explotar una falla de inyección SQL, el atacante debe encontrar un parámetro que la aplicación Web pase hacia la base de datos. Mediante la inserción cuidadosa de comandos SQL en el contenido del parámetro, el atacante puede engañar a la aplicación Web para que envíe una solicitud maliciosa a la base de datos. Estos ataques no son difíciles de intentar y están surgiendo muchas herramientas que buscan estas fallas. Las consecuencias son particularmente dañinas ya que un atacante puede obtener, corromper o destruir el contenido de una base de datos.

Los ataques por inyección pueden ser fáciles de descubrir y explotar, pero también pueden ser extremadamente oscuros. Las consecuencias también pueden ser de una variedad completa, desde triviales hasta comprometer completamente el sistema o destruirlo. En cualquier caso, el uso de llamadas externas está ampliamente extendido, así que la probabilidad de que una aplicación Web tenga fallas de inyección de comandos debe ser considerablemente alta.

### A6.2 Ambientes Afectados

Todo ambiente de aplicación Web permite la ejecución de comandos externos tales como llamadas al sistema, interfaces de comando y solicitudes SQL. La susceptibilidad de una llamada externa a una inyección de comando depende de cómo es hecha la llamada y el componente específico que está siendo llamado, pero casi todas las llamadas externas pueden ser atacadas si la aplicación Web no está apropiadamente codificada.

### A6.3 Ejemplos y Referencias

- Ejemplos: un parámetro malicioso podría modificar las acciones tomadas por una llamada al sistema, que normalmente recupera el archivo actual de usuario, para acceder a otro archivo de usuario (Ej., incluyendo los caracteres de cruce de dirección de fichero `../` como parte de una petición de un nombre de archivo). Comandos adicionales podrían ser añadidos al final de un parámetro que es pasado a un archivo de comandos para ejecutar un comando adicional junto con el comando deseado (Ej., `rm -r *`). Las solicitudes de SQL pudieran ser modificadas agregando "restricciones" adicionales a una cláusula WHERE (Ej., `"OR 1=1"`) para lograr modificar o acceder a datos no autorizados.
- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros, Capítulo 8: Validación de Datos <http://www.owasp.org/documentation/guide/>
- Como Construir un Motor de Validación de Petición HTTP (J2EE validación con Stinger) <http://www.owasp.org/columns/jeffwilliams/jeffwilliams2>
- Ten Tu Pastel y Cómetelo También (validación .NET) <http://www.owasp.org/columns/jpoteet/jpoteet2>
- Documento ("*white paper*") sobre inyección SQL: <http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>



## A6.4 ¿Cómo determinar si usted es vulnerable?

La mejor forma de determinar si es vulnerable a los ataques de inyección de comandos es buscar en el código fuente todas las llamadas a los recursos externos (Ej., system, exec, fork, Runtime.exec, solicitudes SQL o cualquiera sea la sintaxis para realizar solicitudes a los intérpretes de su ambiente). Note que muchos lenguajes tienen múltiples maneras de ejecutar comandos externos. Los desarrolladores deben revisar su código y buscar por todos los lugares donde la entrada de una petición HTTP podría posiblemente hacerse camino a través de cualquiera de estas llamadas. Debe examinar cuidadosamente cada una de estas llamadas para asegurarse que los pasos de protección delineados anteriormente son seguidos.

## A6.5 ¿Cómo auto protegerse?

La manera más simple de protegerse contra las inyecciones es evitar acceder a intérpretes externos siempre que sea posible. Para muchos comandos y llamadas al sistema, hay bibliotecas específicas de lenguajes que realizan las mismas funciones. Usar tales bibliotecas no involucra al intérprete de comandos y por lo tanto evita un gran número de problemas con comandos de sistema.

Para aquellas llamadas que forzosamente debe hacer, como llamadas a bases de datos, debe validar cuidadosamente los datos proveídos para asegurar que no almacenan algún contenido malicioso. También puede estructurar muchas peticiones de manera que aseguren que todos los parámetros proveídos son tratados como datos y no como código ejecutable potencial. El uso de procedimientos almacenados y sentencias preparadas (“prepared statements”) proveerán protección significativa, asegurando que las entradas proveídas sean tratadas como datos. Estas medidas reducirán, pero no eliminarán completamente el riesgo relacionado a estas llamadas externas. De todas maneras debe validar siempre tales entradas para asegurarte que cumplen con las expectativas de la aplicación en cuestión.

Otro tipo de protección fuerte contra la inyección de comandos es asegurarse que la aplicación corre sólo con los privilegios indispensablemente necesarios para realizar su función. Así que no debería correr el servidor Web como administrador o acceder a la base de datos como DBADMIN, de otra manera un atacante puede abusar de los privilegios administrativos otorgados a la aplicación Web. Algunos de los ambientes J2EE permiten el uso de cajas de arena Java (“Java Sand Box”) que pueden evitar la ejecución de comandos del sistema.

Si un comando externo debe ser usado, cualquier información del usuario que sea insertada dentro del comando debe ser rigurosamente revisada. Deben de usarse mecanismos para manejar cualquier error posible, tiempo de espera agotado o bloqueos durante las llamadas.

Todas las salidas, códigos de retorno y códigos de error de la llamada deben ser revisadas para asegurar que el procesamiento esperado sea realmente el que ocurrió. Al menos, esto permitirá determinar que algo salió mal. De otra manera el ataque puede ocurrir y nunca sería detectado.

El proyecto de Filtros OWASP está produciendo componentes reutilizables en diversos lenguajes para ayudar a evitar muchas formas de inyección. OWASP también ha liberado CodeSeeker, un cortafuegos a nivel de aplicación.



## A7 Manejo Inadecuado de Errores

### A7.1 Descripción

El manejo inadecuado de errores puede introducir variados problemas de seguridad a un sitio Web. El error mas común es cuando información detallada de mensajes error, como rastreos de pila, volcados de BD y códigos de error son mostrados al usuario (un pirata informático). Estos mensajes revelan detalles de la implementación que nunca deberían ser revelados. Tales detalles pueden proveer a los piratas informáticos de pistas importantes sobre potenciales fallas en el sitio y tales mensajes de error son también perturbadores para los usuarios normales.

Las aplicaciones Web frecuentemente generan condiciones de error durante su operación normal. Falta de memoria, excepciones por punteros nulos, llamadas fallidas a sistema, BD no disponible, tiempo de espera de red agotado y cientos de otras condiciones comunes pueden causar que los errores sean generados. Estos errores deben ser manejados de acuerdo a un esquema bien pensado que provea al usuario de un mensaje de error con sentido, información de diagnostico para quienes mantienen el sitio, y ninguna información útil para un atacante.

Incluso cuando los mensajes de error no proveen muchos detalles, las inconsistencias en tales mensajes pueden revelar pistas importantes de cómo funciona un sitio y qué información esta presente bajo la cubierta. Por ejemplo, cuando un usuario trata de acceder a un archivo que no existe, el mensaje de error tradicionalmente indica “archivo no encontrado”. Cuando se accede a un archivo al que el usuario no está autorizado, se indica “acceso negado”. Se supone que el usuario no debe saber siquiera si existe el archivo, pero tales inconsistencias claramente revelan la presencia o ausencia de archivos inaccesibles o la estructura de directorios del sitio.

Un problema común de seguridad causado por el manejo inadecuado de errores es la prueba de seguridad de apertura de archivos. Todos los mecanismos de seguridad deben negar el acceso hasta que sea específicamente otorgado, y no otorgar acceso hasta que sea negado, lo cual es una razón común de porque ocurren los errores de apertura de archivos. Otros errores pueden causar que el sistema se caiga o consuma recursos significativos, negando o reduciendo efectivamente el servicio a usuarios legítimos.

Un buen mecanismo de manejo de errores debe ser capaz de manejar cualquier conjunto de entradas posible, mientras fomenta una seguridad apropiada. Mensajes de error simples deben ser producidos y registrados de tal manera que su causa, ya sea un error en el sitio o un intento de ataque, pueda ser revisada. El manejo de errores debe no sólo enfocarse en las entradas proveídas por el usuario, sino que deben también incluir cualquier error que pueda ser generado por componentes internos tales como llamadas al sistema, consultas de BD o cualquier otra función interna.

### A7.2 Ambientes Afectados

Todos los servidores Web, Servidores de aplicación y ambientes de aplicaciones Web son susceptibles a problemas de manejo de errores.

### A7.3 Ejemplos y Referencias

- Discusión sobre generación de códigos de error de OWASP: <http://www.owasp.org/documentation/guide/>

### A7.4 ¿Cómo determinar si usted es vulnerable?

Tradicionalmente una revisión simple puede determinar cómo responde su sitio a varios tipos de errores de entrada. Pruebas más profundas son usualmente requeridas para hacer que ocurran errores internos y ver cómo se comporta el sitio.

Otra estrategia importante es tener una revisión detallada del código que busque en éste la lógica en el manejo de errores. El manejo de errores debe ser consistente a través de todo el sitio y cada pieza debe ser parte de un esquema bien diseñado. Una revisión de código revelará cómo pretende el sistema manejar varios tipos de errores. Si encuentra que no



hay organización en el esquema de manejo de errores o que parece que hay muchos esquemas diferentes, muy posiblemente hay un problema.

### **A7.5 ¿Cómo auto protegerse?**

Una política específica de cómo manejar errores debería ser documentada, incluyendo los tipos de errores a ser manejados y, para cada uno de ellos, qué información debe ser reportada al usuario y qué información será registrada. Todos los desarrolladores necesitan entender la política y asegurarse que su código la siga.

En la implementación, asegúrese de que el sitio está construido para manejar elegantemente todos los posibles errores. Cuando los errores ocurran, el sitio debe responder con un resultado específicamente diseñado que sea de ayuda al usuario, sin que revele detalles internos innecesarios. Ciertas clases de errores deben ser registrados para ayudar a detectar errores de implementación en el sitio y/o intentos de ataque.

Muy pocos sitios tienen alguna habilidad de detección de intrusos en su aplicación Web, pero es ciertamente concebible que una aplicación Web pueda rastrear repetidos intentos fallidos y generar alertas. Note que la vasta mayoría de ataques a aplicaciones Web no son detectados nunca porque muy pocos sitios tienen la capacidad para hacerlo. Por lo tanto, el éxito de los ataques contra la seguridad de aplicaciones Web parece ser seriamente subestimado.

El proyecto de Filtros OWASP está produciendo componentes reutilizables en diversos lenguajes, los cuales ayudan a evitar revelar códigos de error dentro de la páginas Web de los usuarios ya que filtran las páginas cuando están construidas dinámicamente por la aplicación.



## A8 Almacenamiento Inseguro

### A8.1 Descripción

Muchas de las aplicaciones Web necesitan guardar información sensible, ya sea en una base de datos, en un sistema de archivos o en algún lugar. La información podría ser contraseñas, números de tarjetas de crédito, registros contables o información propietaria. Frecuentemente las técnicas de encriptación (“encryption”) son usadas para proteger esta información sensible. Mientras que la encriptación se ha vuelto relativamente fácil de implementar y usar, los desarrolladores frecuentemente cometen errores aun cuando la integran dentro de una aplicación Web. Los desarrolladores pueden sobrestimar la protección ganada por el uso de la encriptación y no ser cuidadosos en asegurar otros aspectos del sitio. Unas pocas áreas donde los errores son comúnmente cometidos incluyen:

- Fallar al encriptar información crucial.
- Almacenamiento inseguro de llaves, certificados y contraseñas.
- Almacenamiento incorrecto de secretos en memoria.
- Fuentes pobres de aleatorización.
- Elección pobre de algoritmo.
- Intentar inventar el nuevo algoritmo de encriptación.
- Fallar al incluir soporte para cambios en las llaves de encriptación y otros procedimientos requeridos de mantenimiento.

El impacto de estas debilidades puede ser devastador para la seguridad del sitio Web. La encriptación es generalmente usada para proteger los activos más sensibles del sitio, los cuales pueden ser totalmente comprometidos por una debilidad.

### A8.2 Ambientes afectados

La mayoría de los ambientes de aplicación Web incluyen alguna forma de soporte criptográfico. En el raro caso que tal soporte no esté disponible aún, hay una gran variedad de productos de terceros que pueden ser agregados. Sólo los sitios Web que usan encriptación para proteger información almacenada o en tránsito son susceptibles a estos ataques. Note que esta sección no cubre el uso de SSL, el cual es cubierto en la sección A10 Manejo de Configuración Segura. Esta sección menciona sólo la encriptación programática de datos en la capa de aplicación.

### A8.3 Ejemplos y referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros  
<http://www.owasp.org/documentation/guide/>
- Bruce Schneier, “Criptografía Aplicada”, 2ª edición, John Wiley & Sons, 1995

### A8.4 ¿Cómo determinar si usted es vulnerable?

Descubrir fallas criptográficas sin acceso al código fuente puede tomar muchísimo tiempo. Sin embargo, es posible examinar tokens, id’s de sesión, cookies y otras credenciales para ver si obviamente no son aleatorias. Todas las estrategias cripto-analíticas pueden ser usadas para intentar descubrir cómo es que un sitio Web usa las funciones criptográficas.

Por mucho, la estrategia más fácil es revisar el código para ver cómo están implementadas las funciones criptográficas. Una revisión cuidadosa de la estructura, calidad e implementación de los módulos criptográficos debe ser hecha. El revisor debe tener un fuerte conocimiento previo en el uso de la criptografía y las fallas comunes. La revisión debe cubrir también cómo las llaves, contraseñas y otros secretos son almacenados, protegidos, cargados, procesados y limpiados de la memoria.



### **A8.5 ¿Cómo auto protegerse?**

La manera más fácil de protegerse contra las fallas criptográficas es minimizar el uso de la encriptación y sólo mantener la información que es absolutamente necesaria. Por ejemplo, más que encriptar los números de tarjetas de crédito y guardarlos, simplemente pida a los usuarios que los introduzcan nuevamente. También, en lugar de guardar las contraseñas encriptadas con un algoritmo que utilice llaves, use una función de una vía como SHA-1 para encriptar las contraseñas.

Si la criptografía debe ser usada, escoja una biblioteca que ha sido expuesta al escrutinio público y asegúrese de que no hay vulnerabilidades abiertas. Encapsule las funciones criptográficas que son usadas y revise el código cuidadosamente. Asegúrese de que los secretos, como las llaves, certificados y contraseñas son almacenados de forma segura. Para hacerlo más difícil para un atacante, un secreto maestro puede ser dividirla en al menos dos ubicaciones y ensamblarla en tiempo de ejecución. Tales ubicaciones pueden incluir archivos de configuración, un servidor externo o dentro del código mismo.



## A9 Negación de Servicio

### A9.1 Descripción

Las aplicaciones Web son especialmente susceptibles a ataques de negación de servicio. Toma nota de que los ataques de negación de servicio al nivel de red, como las inundaciones SYN (SYN Floods), son un problema independiente que está fuera del alcance de este documento

Las aplicaciones Web no pueden fácilmente identificar la diferencia entre un ataque y el tráfico ordinario. Hay muchos factores que contribuyen a esta dificultad, pero uno de los más importantes es que, por un número de razones, las direcciones IP no son útiles como una credencial de identificación. Porque no hay manera confiable para decir de dónde proviene una petición HTTP, es muy difícil filtrar tráfico malicioso. Para ataques distribuidos ¿Cómo una aplicación podría identificar la diferencia entre un ataque real, múltiples usuarios dando al mismo tiempo un clic en el botón recargar (lo cual puede suceder si hay un problema temporal con el sitio), o está siendo “derribado” (“slashdotted”)?

La mayoría de los servidores Web pueden manejar varios cientos de usuarios concurrentes bajo condiciones normales de uso. Sin embargo, un solo atacante puede generar suficiente tráfico desde una sola computadora para desplazar muchas aplicaciones. El balanceo de carga puede hacer estos ataques más difíciles, pero no imposibles, especialmente si las sesiones están ligadas a un servidor en particular. Esta es una buena razón para hacer los datos de sesión de la aplicación lo más pequeños posible y hacer un tanto difícil, de alguna manera, el iniciar una nueva sesión.

Cuando un atacante puede consumir completamente algún recurso requerido, puede impedir a los usuarios legítimos usar el sistema. Algunos recursos limitados incluyen el ancho de banda, las conexiones a BD, espacio en disco, CPU, memoria, hilos o recursos específicos de las aplicaciones. Todos estos recursos pueden ser consumidos por ataques que hagan blanco en ellos. Por ejemplo, un sitio que permite a los usuarios sin autenticar solicitar tráfico de un tablero de mensajes, puede iniciar muchas consultas a la base de datos por cada petición HTTP que recibe. Un atacante puede fácilmente enviar tantas peticiones que la fila de conexiones de la base de datos las use todas y no quedará ninguna para los usuarios legítimos del servicio.

Otras variantes de estos ataques hacen blanco en los recursos del sistema relacionados a un usuario en particular. Por ejemplo, un atacante puede ser capaz de bloquear a un usuario legítimo enviando credenciales invalidas hasta que el sistema bloquee la cuenta. O el atacante puede pedir una nueva contraseña para el usuario, forzándolo a acceder a su cuenta de correo para volver a obtener acceso. Alternativamente, si el sistema bloquea alguno de los recursos de un usuario, entonces el atacante podría potencialmente interrumpir el acceso a éstos de manera que otros no pudieran usarlo.

Algunas aplicaciones Web son incluso susceptibles a ataques que las pondrán inmediatamente fuera de línea. Las aplicaciones que no manejan los errores apropiadamente pueden incluso tumbar el contenedor del servidor Web. Estos ataques son particularmente devastadores porque instantáneamente le impiden usar la aplicación a todos los usuarios.

Hay una amplia variedad de estos ataques, muchos de los cuales pueden fácilmente ser lanzados con unas pocas líneas de código Perl desde una computadora de bajo poder. Mientras no haya defensas perfectas para estos ataques, es imposible hacer más difícil que tengan éxito.

### A9.2 Ambientes afectados

Todos los servidores Web, servidores de aplicación y ambientes de aplicación Web son susceptibles a ataques de negación de servicio.

### A9.3 Ejemplos y Referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros  
<http://www.owasp.org/documentation/guide/>



#### **A9.4 ¿Cómo determinar si usted es vulnerable?**

Una de las partes difíciles de los ataques de negación de servicio es determinar si se es vulnerable. Herramientas de pruebas de carga, como el JMeter pueden generar tráfico Web de manera que puedas probar ciertos aspectos de cómo se desempeña su sitio bajo tráfico pesado. Ciertamente una prueba importante es cuantas peticiones por segundo puede manejar su aplicación. Probar desde una sola dirección IP es útil pues le da una idea de cuántas peticiones puede un atacante generar para dañar su sitio.

Para determinar si cualquier recurso puede ser usado para crear un ataque de negación de servicio, debe analizar cada uno para ver si hay alguna manera de agotarlo. Debe enfocarse particularmente en qué puede hacer un usuario autenticado y, a menos que confíe en todos tus usuarios, debe determinar también qué puede hacer un usuario no autenticado.

#### **A9.5 ¿Cómo auto protegerse?**

Defenderse de un ataque de negación de servicio es difícil, ya que no existe una manera perfecta de protegerse de estos ataques.

Como regla general, debe limitar al mínimo posible los recursos asignados a cualquier usuario. Para usuarios autenticados es posible establecer cuotas de manera que puedan limitar la cantidad de carga que un usuario en particular puede poner sobre el sistema. Particularmente puede considerar sólo manejar una petición por usuario cada vez, sincronizando la sesión del usuario. Podría también considerar suspender cualquier petición de un usuario que se esté procesando cuando llegue otra petición de ese mismo usuario.

Para usuarios no autenticados, debe evitar cualquier acceso innecesario a la base de datos u otro recurso caro. Trata de diseñar el flujo de su sitio de manera que un usuario no autenticado esté incapacitado para ejecutar alguna operación costosa. Puede considerar guardar en la memoria rápida el contenido recibido por usuarios no autenticados en lugar de generarlo o acceder a la base de datos para obtenerlo.

Debe también verificar su esquema para manejo de errores a fin de asegurar que un error no puede afectar la operación general de la aplicación.



## A10 Administración de Configuración Insegura

### A10.1 Descripción

El Servidor Web y el servidor de aplicación juegan un papel clave en la seguridad de una aplicación Web. Estos servidores son responsables de servir el contenido y ejecutar aplicaciones que generan contenido. Además, muchos servidores de aplicación proveen un número de servicios que los servidores Web pueden usar, incluyendo almacenamiento de información, servicios de directorio, correo, mensajería y más. Fallar al manejar la configuración apropiada de sus servidores puede llevar a una amplia variedad de problemas de seguridad.

Frecuentemente el grupo de desarrollo Web está separado del grupo que opera el sitio. De hecho, hay comúnmente un amplio trecho entre aquellos que escriben la aplicación y aquellos responsables del ambiente de operaciones. Lo relacionado a seguridad en aplicaciones Web frecuentemente acorta este trecho y requiere que los miembros de ambos lados del proyecto afiancen apropiadamente la seguridad del sitio de la aplicación.

Hay una amplia variedad de problemas de configuración de servidores que pueden plagar la seguridad de un sitio. Estos incluyen:

- Fallas de seguridad no parchadas en el software del servidor.
- Fallas de seguridad en el software del servidor o malas configuraciones que permiten ataques de listado de directorio o atravesamiento de directorio.
- Innecesarios archivos por defecto, de respaldo o de ejemplo, incluyendo Scripts, aplicaciones, archivos de configuración y páginas Web.
- Permisos no adecuados en archivos y directorios.
- Servicios innecesarios habilitados, incluyendo manejo de contenido y administración remota.
- Cuentas por defecto con contraseñas por defecto.
- Funciones administrativas o de depuración que son habilitadas o accesibles.
- Mensajes de error demasiado informativos (más detalles en la sección de manejo de errores)
- Certificados SSL y opciones de encriptación mal configurados.
- Uso de certificados auto firmados para alcanzar la autenticación y protección sobre ataques de hombre-en-el-medio.
- Uso de certificados por defecto.
- Autenticación inadecuada con sistemas externos.

Algunos de estos problemas pueden ser detectados con herramientas de examinación de seguridad (security scanning tools) ampliamente disponibles. Una vez detectados, estos problemas pueden ser fácilmente explotados y resultan en el compromiso total de un sitio Web. Los ataques exitosos pueden también resultar en el compromiso de los sistemas de respaldo incluyendo bases de datos y redes corporativas. Ambos, tener software seguro y una configuración segura, son requeridos para tener un sitio seguro.

### A10.2 Ambientes Afectados

Todos los servidores Web, servidores de aplicación y ambientes de aplicaciones Web son susceptibles a la mala configuración.

### A10.3 Ejemplos y Referencias

- Guía de OWASP para Construir Aplicaciones Web y Servicios Web Seguros  
<http://www.owasp.org/documentation/guide/>



- Mejores prácticas de seguridad para servidores Web: <http://www.pcmag.com/article2/0,4149,11525,00.asp>
- Asegurando servidores Web públicos (del CERT): <http://www.cert.org/security-improvement/modules/m11.html>

#### **A10.4 ¿Cómo determinar si usted es vulnerable?**

Si no ha hecho un esfuerzo concienzudo para asegurar sus servidores Web y de aplicaciones, es muy probable que sea vulnerable. Pocos productos de servidor, sino es que ninguno, son seguros al sacarlos de la caja. Una configuración segura para su plataforma debe ser documentada y actualizada frecuentemente. Una revisión manual de la guía de configuración debe ser hecha regularmente para asegurar que ha sido mantenida actual y consistente. También es recomendable una comparación con los sistemas actualmente desplegados.

Además hay un número de productos de examinación disponibles que externamente buscan vulnerabilidades conocidas en una aplicación o servidor Web, incluyendo Nessus y Nikto. Debe correr estas herramientas regularmente, al menos mensualmente, para encontrar problemas tan pronto como sea posible. Las herramientas debe ser corridas tanto internamente como externamente. Las examinaciones externas deben correr desde una computadora localizada fuera de la red del servidor. Las examinaciones internas deben ser corridas desde la misma red de los servidores objetivo.

#### **A10.5 ¿Cómo auto protegerse?**

El primer paso es crear una guía de fortificación para su configuración particular de servidor Web y servidor de aplicación. Esta configuración debe ser usada en todas las computadoras corriendo la aplicación y también en los ambientes de desarrollo. Recomendamos empezar con cualquier guía existente que pueda obtener de su proveedor o aquellas disponibles por las organizaciones de seguridad como OWASP, CERT y SANS, y después adecuarlas a tus necesidades particulares. La guía de fortificación debe contener los siguientes temas:

- Configuración de todos los mecanismos de seguridad.
- Dar de baja todos los servicios no usados.
- Establecer roles, permisos y cuentas, incluyendo la deshabilitación de todas las cuentas por defecto o cambiando sus contraseñas.
- Registro de eventos y alertas.

Una vez que su guía ha sido establecida, úsela para configurar y mantener sus servidores. Si tiene un número grande de servidores que configurar, considere semi-automatizar o automatizar completamente el proceso de configuración. Use una herramienta de configuración existente o cree la suya propia. Un número de herramientas de este tipo ya existen. Puede también usar herramientas de replicación de discos como Ghost para hacer una imagen de un servidor fortificado existente y después replicar esa imagen a servidores nuevos. Tal proceso puede o no funcionar para su ambiente particular.

Mantener la configuración del servidor segura requiere vigilancia. Debe de asegurarse que la responsabilidad de mantener la configuración del servidor actualizada sea asignada a un individuo o equipo. El proceso de mantenimiento incluye:

- Monitorear las últimas vulnerabilidades publicadas
- Aplicar los parches más recientes.
- Actualizar la guía de configuración de seguridad.
- Búsqueda de vulnerabilidades frecuentes desde ambas perspectivas, interna y externa.
- Revisiones internas frecuentes de la configuración de seguridad del servidor comparada a la guía de configuración.
- Reportes de estado regulares a la gerencia superior, documentando la postura general de seguridad.



## Conclusiones

OWASP ha ensamblado esta lista para incrementar la conciencia sobre la seguridad en aplicaciones Web. Los expertos en OWASP han concluido que estas vulnerabilidades representan un serio riesgo para agencias y compañías que han expuesto su lógica de negocio a la Internet. Los problemas de seguridad en aplicaciones Web son tan serios como los problemas de seguridad de red. Aunque éstos han recibido, por tradición, considerablemente menos atención. Los atacantes han empezado a enfocarse en los problemas de seguridad en aplicaciones Web y están desarrollando activamente herramientas y técnicas para detectarlos y explotarlos.

Esta lista de las “Top Ten” se encuentra aún en su etapa inicial. Creemos que estas vulnerabilidades representan el riesgo más serio para la seguridad en aplicaciones Web. Sin embargo, hay muchas otras áreas críticas que fueron consideradas para la lista y que también representan un riesgo significativo para las organizaciones que despliegan aplicaciones Web. Estas incluyen fallas en las áreas de:

- Código innecesario o malicioso.
- Seguridad de Hilos Interrumpida y Programación Concurrente.
- Obtención de Información no Autorizada
- Problemas o Débil Registro de Eventos.
- Corrupción de datos.
- Interrupción de Guardado en Memoria Rápida, Enfilamiento y Reutilización.

Damos la bienvenida a tus comentarios sobre esta lista de las “Top Ten”. Por favor participa en las listas de correo de OWASP y ayuda a mejorar la seguridad en aplicaciones Web. Visita <http://www.owasp.org> para empezar.